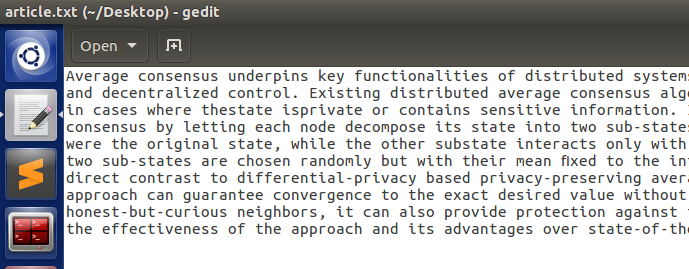
**Lab7-report**

**57118132 吴嘉琪**

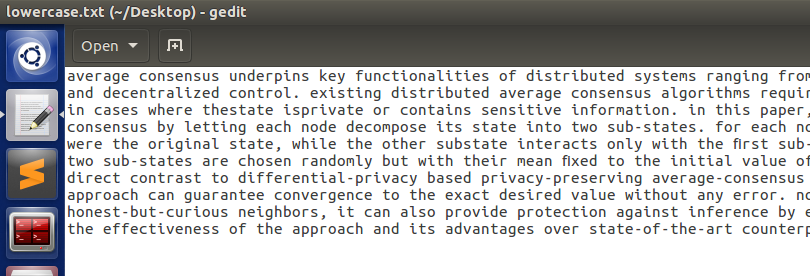
**Task1:** **Frequency Analysis Against Monoalphabetic Substitution Cipher**

首先创建一个article.txt文件，内容为复制的一段英文文章。



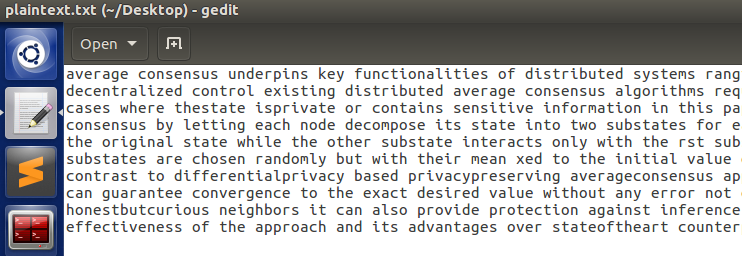
使用tr命令把大写换成小写并将修改后内容存入lowercase.txt文件

$ tr [:upper:] [:lower:] < article.txt > lowercase.txt



使用tr命令把标点符号和数字去掉并将修改后内容存入plaintext.txt文件

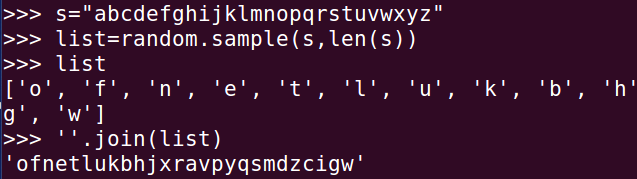
$ tr -cd ’[a-z][\n][:space:]’ < lowercase.txt > plaintext.txt



在python3中生成替换表

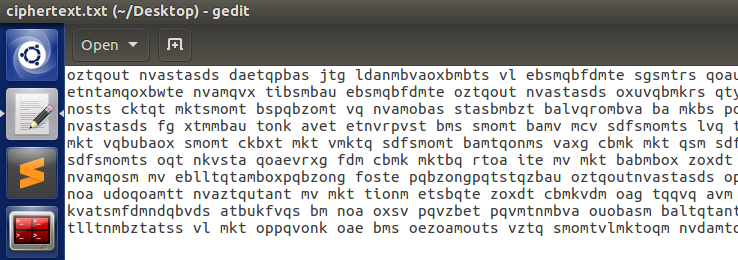
random.sample()方法随机地从指定列表中提取出len（s）个不同的元素。

‘’.join(list) 以‘’内的字符作为分隔符（这里没有分隔符），将list内的所有的元素合并成一个新的字符串。



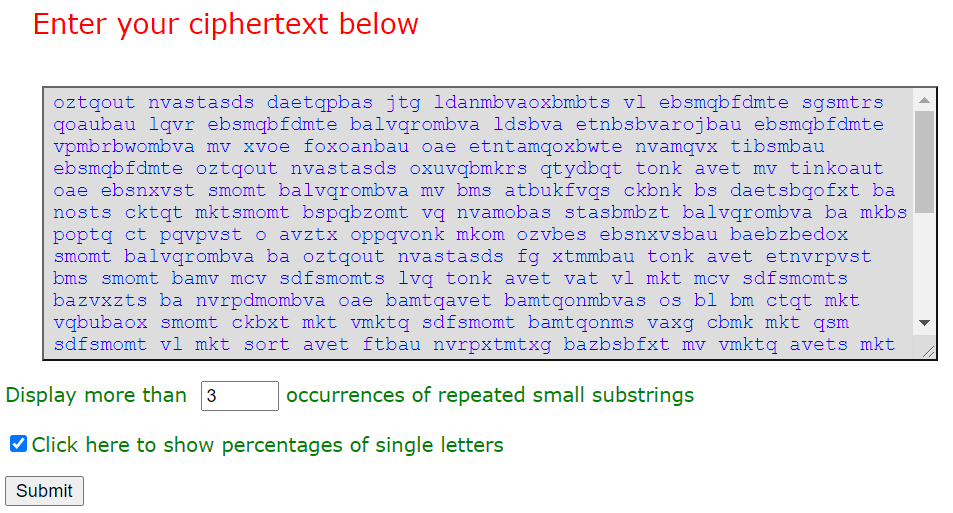
使用tr命令通过替换进行加密

$ tr 'a-z' 'ofnetlukbhjxravpyqsmdzcigw' < plaintext.txt > ciphertext.txt

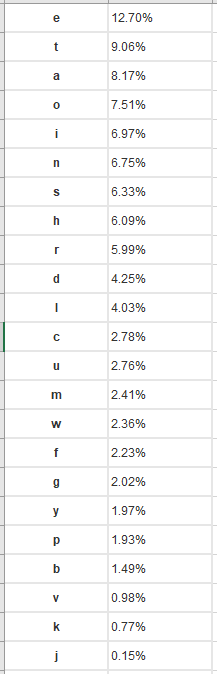
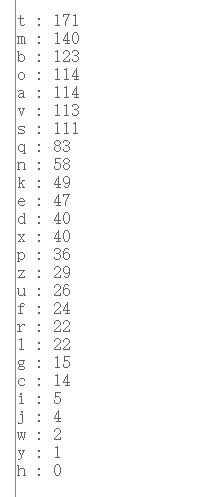
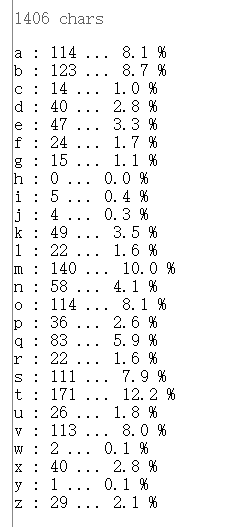


在工具网站上分析密文中字母出现的频率

使用的工具网站为：<http://www.richkni.co.uk/php/crypta/freq.php>

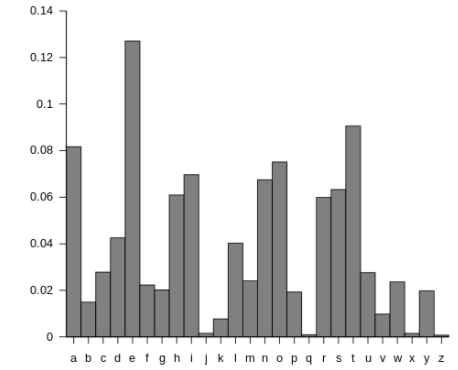
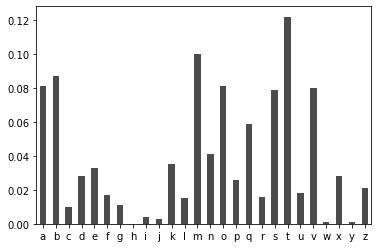


下面三张图，左边啊啊啊两张分别为实验文章密文的字母频率与频数，右边一张为标准字母频率。



与标准英文字母频率相对照，按照频率对比替换

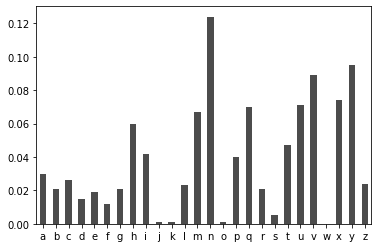
左图为实验文章密文的字母频率，右图为维基百科中得到的标准字母频率

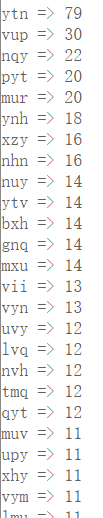
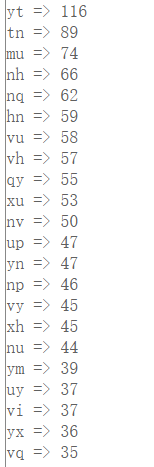
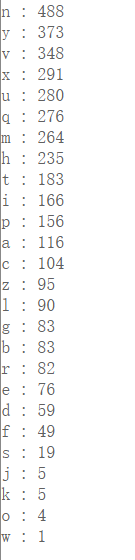
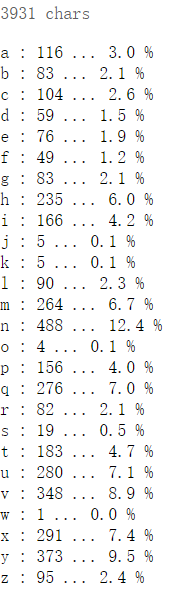


第一次选取的实验文章较短，结果不明显，不容易分析出正确明文。

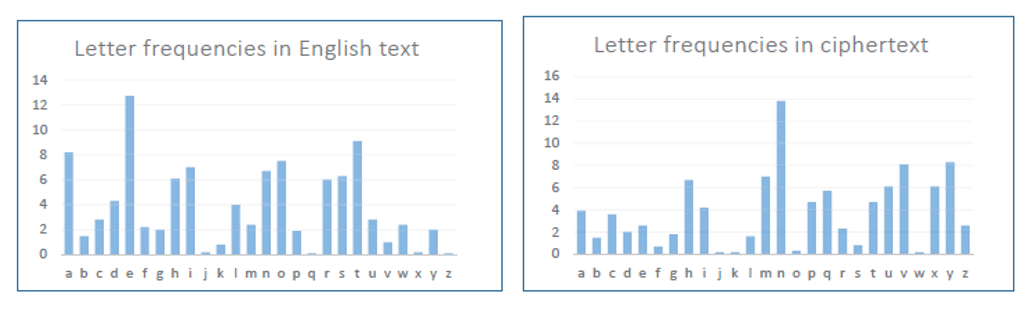
下面使用实验网站提供的密文再次进行实验。

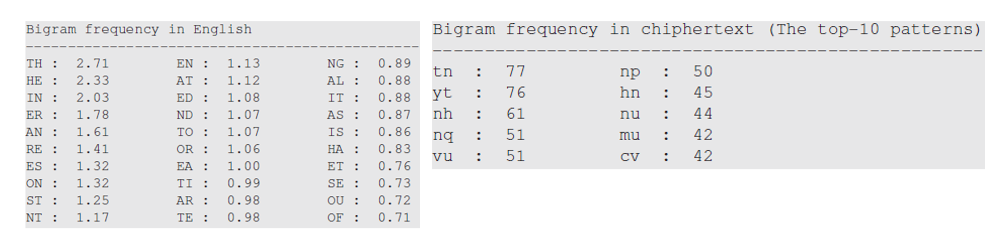
使用工具得到ciphertext文件中各个字母出现的频率以及字母/字母组合出现次数：

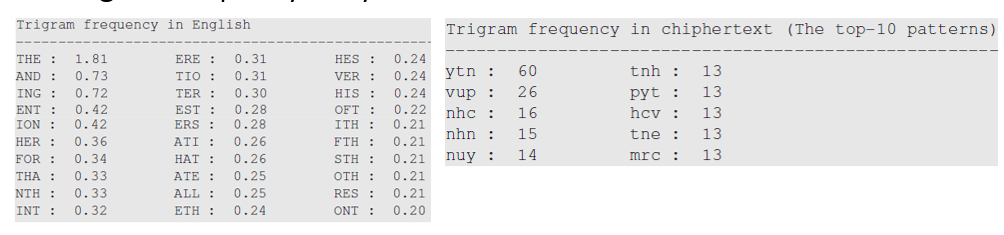




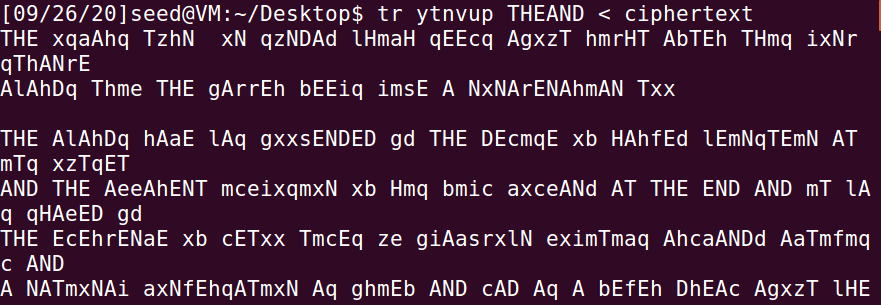
再根据统计规律得到明文密文对应关系：

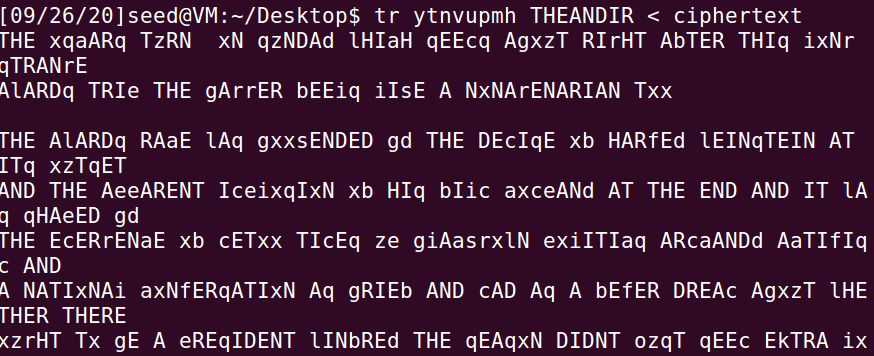


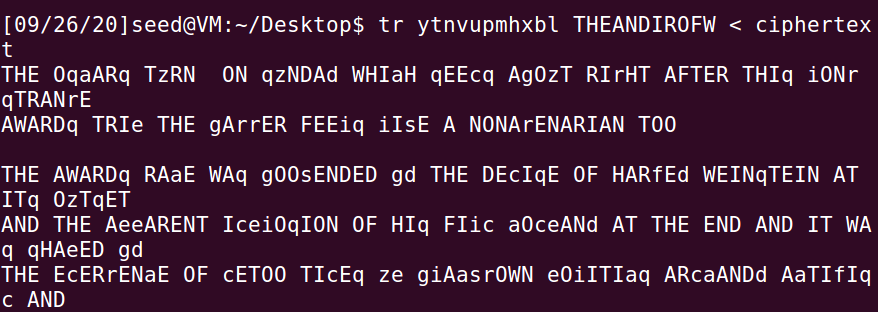




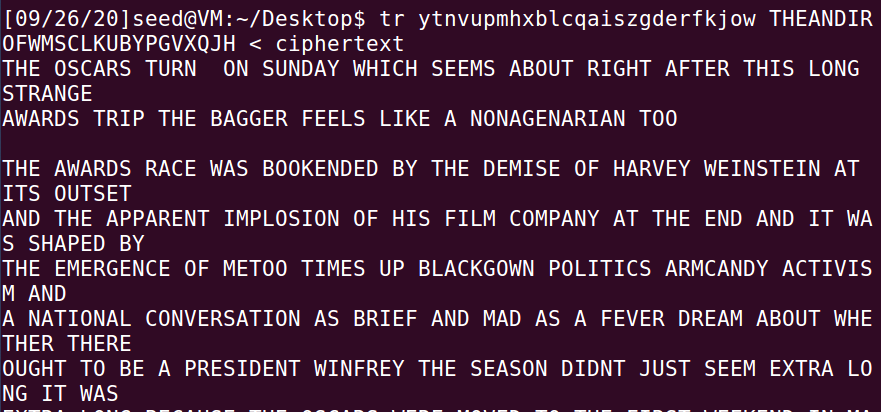
不断猜测对应关系







最后得到对应关系

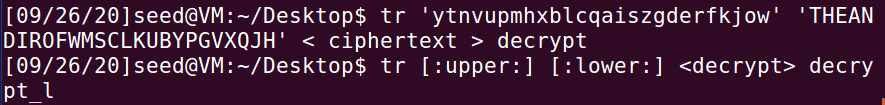


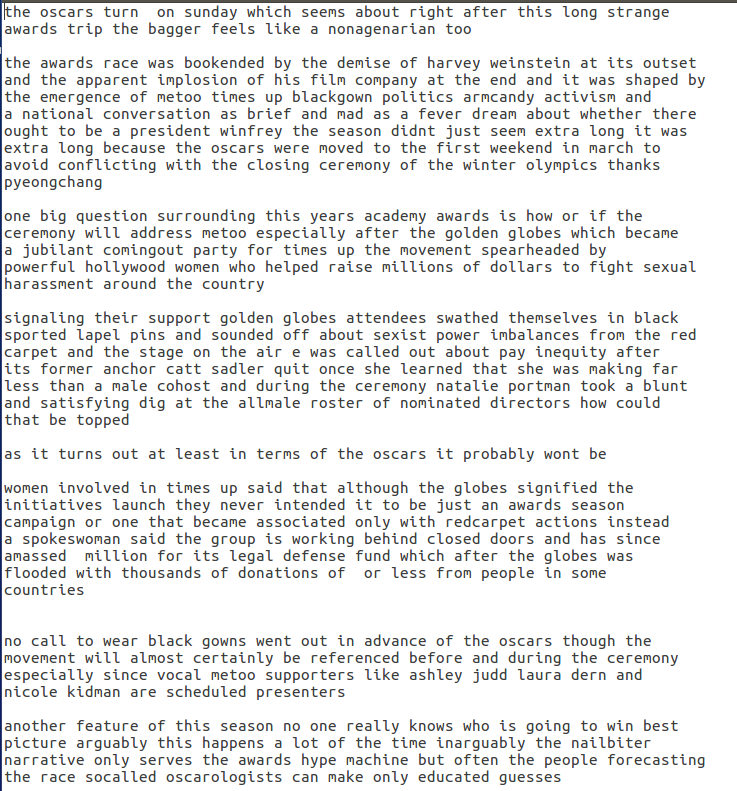
a->c b->f c>m d->y e->p f->v g->b h->r i->l

j->q k->x l->w m->i n->e o->j p->d q->s e->g

s->k t->h u->n v->a w->h x->o y->t z->u

将解密后的结果放入文件中，并转换为小写字母方便阅读





**破解维吉尼亚密码**

代码如下：

1. # -\*- coding: utf-8 -\*-
2. """
3. Created on Sun Sep 27 12:33:05 2020
5. @author: 螺号
6. """
8. **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
9. visiable\_chars=[]#可见字符
10. **for** x **in** range(32,126):
11. visiable\_chars.append(chr(x))
12. #print(vi)
13. test\_keys=[]#用于测试密钥
14. ans\_keys=[]#用于结果的返回
15. **for** x **in** range(0x00,0xFF):# 枚举密钥里所有的值
16. test\_keys.append(x)
17. ans\_keys.append(x)
18. **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
19. **for** s **in** subarr:
20. **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
21. ans\_keys.remove(i)#去掉ans\_keys里测试失败的密钥
22. **break**
23. #print(ans\_keys)
24. **return** ans\_keys
26. strmi='F96DE8C227A259C87EE1DA2AED57C93FE5DA36ED4EC87EF2C63AAE5B9A7EFFD673BE4ACF7BE8923C\
27. AB1ECE7AF2DA3DA44FCF7AE29235A24C963FF0DF3CA3599A70E5DA36BF1ECE77F8DC34BE129A6CF4D126BF\
28. 5B9A7CFEDF3EB850D37CF0C63AA2509A76FF9227A55B9A6FE3D720A850D97AB1DD35ED5FCE6BF0D138A84C\
29. C931B1F121B44ECE70F6C032BD56C33FF9D320ED5CDF7AFF9226BE5BDE3FF7DD21ED56CF71F5C036A94D96\
30. 3FF8D473A351CE3FE5DA3CB84DDB71F5C17FED51DC3FE8D732BF4D963FF3C727ED4AC87EF5DB27A451D47E\
31. FD9230BF47CA6BFEC12ABE4ADF72E29224A84CDF3FF5D720A459D47AF59232A35A9A7AE7D33FB85FCE7AF5\
32. 923AA31EDB3FF7D33ABF52C33FF0D673A551D93FFCD33DA35BC831B1F43CBF1EDF67F0DF23A15B963FE5DA\
33. 36ED68D378F4DC36BF5B9A7AFFD121B44ECE76FEDC73BE5DD27AFCD773BA5FC93FE5DA3CB859D26BB1C63C\
34. ED5CDF3FE2D730B84CDF3FF7DD21ED5ADF7CF0D636BE1EDB79E5D721ED57CE3FE6D320ED57D469F4DC27A8\
35. 5A963FF3C727ED49DF3FFFDD24ED55D470E69E73AC50DE3FE5DA3ABE1EDF67F4C030A44DDF3FF5D73EA250\
36. C96BE3D327A84D963FE5DA32B91ED36BB1D132A31ED87AB1D021A255DF71B1C436BF479A7AF0C13AA14794'
37. arr=[]#密文，每个元素为字符的ascii码
38. **for** x **in** range(0,len(strmi),2):
39. arr.append(int(strmi[x:2+x],16))

42. **for** keylen **in** range(1,14):#枚举密钥的长度1~14
43. **for** index **in** range(0,keylen):#对密钥里的第index个进行测试
44. 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
45. ans\_keys=findindexkey(subarr)#找出密钥中第index个的可能的值
46. **print**('keylen=',keylen,'index=',index,'keys=',ans\_keys)
47. **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
48. ch=[]
49. **for** x **in** ans\_keys:
50. ch.append(chr(x^subarr[0]))
51. **print**(ch)
52. #运行到这里，观察输出可以发现，密钥长度为7时有解
53. **print**('###############')
54. **import** string
55. **def** findindexkey2(subarr):#再造一个函数筛选密钥
56. 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
57. #print(test\_chars)
58. test\_keys=[]#用于测试密钥
59. ans\_keys=[]#用于结果的返回
60. **for** x **in** range(0x00,0xFF):# 枚举密钥里所有的值
61. test\_keys.append(x)
62. ans\_keys.append(x)
63. **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
64. **for** s **in** subarr:
65. **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
66. ans\_keys.remove(i)#去掉ans\_keys里测试失败的密钥
67. **break**
68. #print(ans\_keys)
69. **return** ans\_keys
71. vigenerekeys=[]#维基尼尔密码的密钥
72. **for** index **in** range(0,7):#已经知道密钥长度是7
73. subarr=arr[index::7]
74. vigenerekeys.append(findindexkey2(subarr))
75. **print**(vigenerekeys)#输出的是[[186], [31], [145], [178], [83], [205], [62]].

78. **print**("#########")
79. ming=''
80. **for** i **in** range(0,len(arr)):
81. ming=ming+chr(arr[i]^vigenerekeys[i%7][0])
82. **print**(ming)

解密得到得明文为：

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.

**Many Time Pad**

首先逐个异或各段密文，确定space的位置及其对应位置的密钥，代码如下

1. # -\*- coding: utf-8 -\*-
2. """
3. Created on Fri Sep 25 13:02:17 2020
4. @author: 螺号
5. """
7. **import** collections
9. # Seven different pieces of ciphertext encrypted with the same key
10. c1='BB3A65F6F0034FA957F6A767699CE7FABA855AFB4F2B520AEAD612944A801E'
11. c2='BA7F24F2A35357A05CB8A16762C5A6AAAC924AE6447F0608A3D11388569A1E'
12. c3='A67261BBB30651BA5CF6BA297ED0E7B4E9894AA95E300247F0C0028F409A1E'
13. c4='A57261F5F0004BA74CF4AA2979D9A6B7AC854DA95E305203EC8515954C9D0F'
14. c5='BB3A70F3B91D48E84DF0AB702ECFEEB5BC8C5DA94C301E0BECD241954C831E'
15. c6='A6726DE8F01A50E849EDBC6C7C9CF2B2A88E19FD423E0647ECCB04DD4C9D1E'
16. c7='BC7570BBBF1D46E85AF9AA6C7A9CEFA9E9825CFD5E3A0047F7CD009305A71E'

19. ciphertexts = [c1, c2, c3, c4, c5 ,c6 ,c7 ] #array of all ciphertexts
20. ciphertexts\_counter=collections.Counter()
22. site\_space=[]
23. key=[]
25. **for** current\_index1,ciphertext1 **in** enumerate(ciphertexts):
26. sure\_site\_space=[]
27. counter = collections.Counter()
28. **print**("对于密文",current\_index1+1)
29. **for** current\_index2,ciphertext2 **in** enumerate(ciphertexts):
30. **if** current\_index1 != current\_index2:
31. **print**("密文",current\_index1+1,"与密文",current\_index2+1,"进行异或")
32. **for** site\_index **in** range(len(ciphertext1)):
33. **if** site\_index%2==0:
34. result=int(ciphertext1[site\_index],16)^int(ciphertext2[site\_index],16)
35. **if** (int(result)<=int('7',16) **and** int(result)>=int('4',16)):
36. counter[site\_index]+=1
37. #print(site\_index+1,ciphertext1[site\_index],"^",ciphertext2[site\_index],hex(result))
38. **for** site\_index,value **in** counter.items():
39. **if** value >=4:
40. sure\_site\_space.append(site\_index/2+1)
41. **print**("密文",current\_index1+1,"space位置",sure\_site\_space)
43. **for** index **in** sure\_site\_space:
44. **if** index **not** **in** site\_space:
45. site\_space.append(index)
46. cipher=ciphertext1[int(index\*2)-2:int(index\*2)]
47. temp\_key=int(cipher,16)^int('20',16)
48. key.append(hex(temp\_key))
50. **print**(site\_space)
51. **print**(key)
53. **for** index **in** range(len(key)):
54. **print**('密钥',site\_space[index],"为",key[index])

得到的结果为

对于密文 1

密文 1 与密文 2 进行异或

密文 1 与密文 3 进行异或

密文 1 与密文 4 进行异或

密文 1 与密文 5 进行异或

密文 1 与密文 6 进行异或

密文 1 与密文 7 进行异或

密文 1 space位置 [2.0, 5.0, 14.0, 16.0, 23.0]

对于密文 2

密文 2 与密文 1 进行异或

密文 2 与密文 3 进行异或

密文 2 与密文 4 进行异或

密文 2 与密文 5 进行异或

密文 2 与密文 6 进行异或

密文 2 与密文 7 进行异或

密文 2 space位置 [3.0, 6.0, 10.0, 15.0, 22.0, 25.0]

对于密文 3

密文 3 与密文 1 进行异或

密文 3 与密文 2 进行异或

密文 3 与密文 4 进行异或

密文 3 与密文 5 进行异或

密文 3 与密文 6 进行异或

密文 3 与密文 7 进行异或

密文 3 space位置 [4.0, 12.0, 17.0, 20.0, 24.0]

对于密文 4

密文 4 与密文 1 进行异或

密文 4 与密文 2 进行异或

密文 4 与密文 3 进行异或

密文 4 与密文 5 进行异或

密文 4 与密文 6 进行异或

密文 4 与密文 7 进行异或

密文 4 space位置 [12.0, 15.0, 20.0, 26.0, 5.0, 23.0]

对于密文 5

密文 5 与密文 1 进行异或

密文 5 与密文 2 进行异或

密文 5 与密文 3 进行异或

密文 5 与密文 4 进行异或

密文 5 与密文 6 进行异或

密文 5 与密文 7 进行异或

密文 5 space位置 [8.0, 13.0, 20.0, 27.0, 2.0]

对于密文 6

密文 6 与密文 1 进行异或

密文 6 与密文 2 进行异或

密文 6 与密文 3 进行异或

密文 6 与密文 4 进行异或

密文 6 与密文 5 进行异或

密文 6 与密文 7 进行异或

密文 6 space位置 [8.0, 19.0, 24.0, 28.0, 5.0, 14.0]

对于密文 7

密文 7 与密文 1 进行异或

密文 7 与密文 2 进行异或

密文 7 与密文 3 进行异或

密文 7 与密文 4 进行异或

密文 7 与密文 5 进行异或

密文 7 与密文 6 进行异或

密文 7 space位置 [4.0, 8.0, 17.0, 24.0, 29.0, 14.0]

密钥 2 为 0x1a

密钥 5 为 0xd0

密钥 14 为 0xbc

密钥 16 为 0xda

密钥 23 为 0x72

密钥 3 为 0x4

密钥 6 为 0x73

密钥 10 为 0x98

密钥 15 为 0x86

密钥 22 为 0x5f

密钥 25 为 0x83

密钥 4 为 0x9b

密钥 12 为 0x9

密钥 17 为 0xc9

密钥 20 为 0x89

密钥 24 为 0x67

密钥 26 为 0xa5

密钥 8 为 0xc8

密钥 13 为 0xe

密钥 27 为 0x61

密钥 19 为 0x39

密钥 28 为 0xfd

密钥 29 为 0x25

用得到的部分密钥求解各段密文的部分字段，再根据空格位置和单词长度推测剩余密钥

最终解密明文得

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.