Lab7-Report

学号：**57118131**

姓名：王星沣

Task1

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

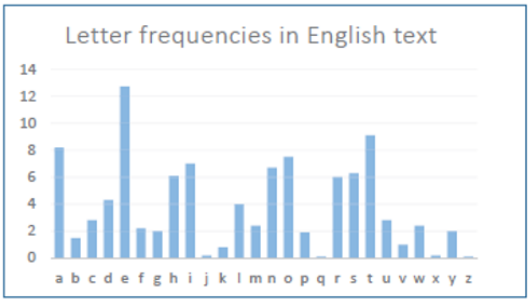
```
3931 chars

a : 116 ... 3.0 %
b : 83 ... 2.1 %
c : 104 ... 2.6 %
d : 59 ... 1.5 %
e : 76 ... 1.9 %
f : 49 ... 1.2 %
g : 83 ... 2.1 %
h : 235 ... 6.0 %
i : 166 ... 4.2 %
j : 5 ... 0.1 %
k : 5 ... 0.1 %
l : 90 ... 2.3 %
m : 264 ... 6.7 %
n : 488 ... 12.4 %
o : 4 ... 0.1 %
p : 156 ... 4.0 %
q : 276 ... 7.0 %
r : 82 ... 2.1 %
s : 19 ... 0.5 %
t : 183 ... 4.7 %
u : 280 ... 7.1 %
v : 348 ... 8.9 %
w : 1 ... 0.0 %
x : 291 ... 7.4 %
y : 373 ... 9.5 %
z : 95 ... 2.4 %
```

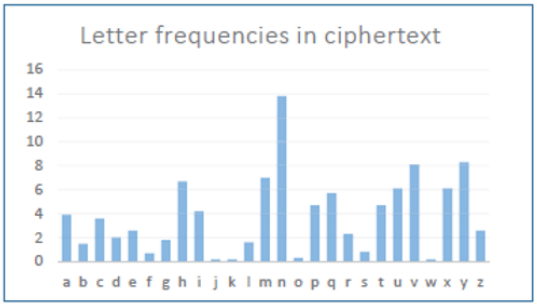
```
n : 488
y : 373
v : 348
x : 291
u : 280
q : 276
m : 264
h : 235
t : 183
i : 166
p : 156
a : 116
c : 104
z : 95
l : 90
g : 83
b : 83
r : 82
e : 76
d : 59
f : 49
s : 19
j : 5
k : 5
o : 4
w : 1
```

```
yt => 116
tn => 89
mu => 74
nh => 66
nq => 62
hn => 59
vu => 58
vh => 57
qy => 55
xu => 53
nv => 50
up => 47
yn => 47
np => 46
vy => 45
xh => 45
nu => 44
ym => 39
uy => 37
vi => 37
yx => 36
vq => 35
```

```
ytn => 79
vup => 30
nqy => 22
pyt => 20
mur => 20
ynh => 18
xzy => 16
nhn => 16
nuy => 14
ytv => 14
bxh => 14
gnq => 14
mxu => 14
vii => 13
vyn => 13
uvy => 12
lvq => 12
nvh => 12
tmq => 12
qyt => 12
muv => 11
upy => 11
xhy => 11
vym => 11
lmu => 11
```

再根据统计规律得到明文密文对应关系：





Bigram frequency in English

TH :	2.71	EN :	1.13	NG :	0.89
HE :	2.33	AT :	1.12	AL :	0.88
IN :	2.03	ED :	1.08	IT :	0.88
ER :	1.78	ND :	1.07	AS :	0.87
AN :	1.61	TO :	1.07	IS :	0.86
RE :	1.41	OR :	1.06	HA :	0.83
ES :	1.32	EA :	1.00	ET :	0.76
ON :	1.32	TI :	0.99	SE :	0.73
ST :	1.25	AR :	0.98	OU :	0.72
NT :	1.17	TE :	0.98	OF :	0.71

Bigram frequency in chiphertext (The top-10 patterns)

tn :	77	np :	50
yt :	76	hn :	45
nh :	61	nu :	44
nq :	51	mu :	42
vu :	51	cv :	42

Trigram frequency in English

THE :	1.81	ERE :	0.31	HES :	0.24
AND :	0.73	TIO :	0.31	VER :	0.24
ING :	0.72	TER :	0.30	HIS :	0.24
ENT :	0.42	EST :	0.28	OFT :	0.22
ION :	0.42	ERS :	0.28	ITH :	0.21
HER :	0.36	ATI :	0.26	FTH :	0.21
FOR :	0.34	HAT :	0.26	STH :	0.21
THA :	0.33	ATE :	0.25	OTH :	0.21
NTH :	0.33	ALL :	0.25	RES :	0.21
INT :	0.32	ETH :	0.24	ONT :	0.20

Trigram frequency in chiphertext (The top-10 patterns)

ytn :	60	tnh :	13
vup :	26	pyt :	13
nhc :	16	hcv :	13
nhn :	15	tne :	13
nuy :	14	mrc :	13

一步一步进行尝试：

```
[09/26/20]seed@VM:~/Desktop$ tr ytnvup THEAND < ciphertext
THE xqaAhq TzhN  xN qzNDAd lHmaH qEEcq AgxzT hmrHT AbTEh THmq ixNr
qThANrE
AlAhDq Thme THE gArrEh bEEiq imsE A NxNArENAhmAN Txx

THE AlAhDq hAaE lAq gxxsENDED gd THE DEcmqE xb HAhfEd lEmNqTEmN AT
mTq xzTqET
AND THE AeeAhENT mceixqmxN xb Hmq bmic axceANd AT THE END AND mT lA
q qHAeED gd
THE EcEhrENaE xb cETxx TmcEq ze giAasrxlN eximTmaq AhcaANDd AaTmfmq
c AND
A NATmxNAi axNfEhqATmxN Aq ghmEb AND cAD Aq A bEfEh DhEAc AgxzT lHE
```

```
[09/26/20]seed@VM:~/Desktop$ tr ytnvupmh THEANDIR < ciphertext
THE xqaARq TzRN  xN qzNDAd lHIaH qEEcq AgxzT RIrHT AbTER THIq ixNr
qTRANrE
AlARDq TRIe THE gArrER bEEiq iIsE A NxNArENARIAN Txx

THE AlARDq RAaE lAq gxxsENDED gd THE DEcIqE xb HARfEd lEINqTEIN AT
ITq xzTqET
AND THE AeeARENT IceixqIxN xb HIq bIic axceANd AT THE END AND IT lA
q qHAeED gd
THE EcERrENaE xb cETxx TIcEq ze giAasrxlN exiITIaq ARcaANDd AaTIfIq
c AND
A NATIxNAi axNfERqATIxN Aq gRIEb AND cAD Aq A bEfER DREAc AgxzT lHE
THER THERE
xzrHT Tx gE A eREqIDENT lINbREd THE qEAqxN DIDNT ozqT qEEc EkTRA ix
```

```
[09/26/20]seed@VM:~/Desktop$ tr ytnvupmhxbl THEANDIROFW < ciphertex
t
THE OqaARq TzRN  ON qzNDAd WHIaH qEEcq AgOzT RIrHT AFTER THIq iONr
qTRANrE
AWARDq TRIe THE gArrER FEEiq iIsE A NONArENARIAN TOO

THE AWARDq RAaE WAq gOOsENDED gd THE DEcIqE OF HARfEd WEINqTEIN AT
ITq OzTqET
AND THE AeeARENT IceiOqION OF HIq FIic aOceANd AT THE END AND IT WA
q qHAeED gd
THE EcERrENaE OF cETOO TIcEq ze giAasrOWN eOiITIaq ARcaANDd AaTIfIq
c AND
```

最后得到正确的对应关系：

```
[09/26/20]seed@VM:~/Desktop$ tr ytnvupmhxblcqaiszgderfkjow THEANDIR
OFWMSCLKUBYPGVXQJH < ciphertext
THE OSCARS TURN  ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG
STRANGE
AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO

THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT
ITS OUTSET
AND THE APPARENT IMPLOSION OF HIS FILM COMPANY AT THE END AND IT WA
S SHAPED BY
THE EMERGENCE OF METOO TIMES UP BLACKGOWN POLITICS ARMCANDY ACTIVIS
M AND
A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHE
THER THERE
OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LO
NG IT WAS
```

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：

```
[09/26/20]seed@VM:~/Desktop$ tr 'ytnvupmhxblcqaiszgderfkjow' 'THEAN
DIROFWMSCLKUBYPGVXQJH' < ciphertext > decrypt
[09/26/20]seed@VM:~/Desktop$ tr [:upper:] [:lower:] <decrypt> decry
pt_l
```

```
the oscars turn  on sunday which seems about right after this long strange
awards trip the bagger feels like a nonagenarian too

the awards race was bookended by the demise of harvey weinstein at its outset
and the apparent implosion of his film company at the end and it was shaped by
the emergence of metoo times up blackgown politics armcandy activism and
a national conversation as brief and mad as a fever dream about whether there
ought to be a president winfrey the season didnt just seem extra long it was
extra long because the oscars were moved to the first weekend in march to
avoid conflicting with the closing ceremony of the winter olympics thanks
pyeongchang

one big question surrounding this years academy awards is how or if the
ceremony will address metoo especially after the golden globes which became
a jubilant comingout party for times up the movement spearheaded by
powerful hollywood women who helped raise millions of dollars to fight sexual
harassment around the country

signaling their support golden globes attendees swathed themselves in black
sported lapel pins and sounded off about sexist power imbalances from the red
carpet and the stage on the air e was called out about pay inequity after
its former anchor catt sadler quit once she learned that she was making far
less than a male cohost and during the ceremony natalie portman took a blunt
and satisfying dig at the allmale roster of nominated directors how could
that be topped

as it turns out at least in terms of the oscars it probably wont be

women involved in times up said that although the globes signified the
initiatives launch they never intended it to be just an awards season
campaign or one that became associated only with redcarpet actions instead
a spokeswoman said the group is working behind closed doors and has since
amassed  million for its legal defense fund which after the globes was
flooded with thousands of donations of  or less from people in some
countries


no call to wear black gowns went out in advance of the oscars though the
movement will almost certainly be referenced before and during the ceremony
especially since vocal metoo supporters like ashley judd laura dern and
nicole kidman are scheduled presenters

another feature of this season no one really knows who is going to win best
picture arguably this happens a lot of the time inarguably the nailbiter
narrative only serves the awards hype machine but often the people forecasting
the race socalled oscarologists can make only educated guesses
```

破解维吉尼亚密码：

代码截图如下：

```python
# -*- coding: utf-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
    visiable_chars=[]#可见字符
    for x in range(32,126):
        visiable_chars.append(chr(x))

    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
    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
                ans_keys.remove(i)#去掉 ans_keys 里测试失败的密钥
                break

    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("##########")
msg=''
for i in range(0,len(arr)):
    msg=msg+chr(arr[i]^vigenerekeys[i%7][0])
print(msg)

```

运行结果如图：

```python
# -*- coding: utf-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
    visiable_chars=[]#可见字符
    for x in range(32,126):
        visiable_chars.append(chr(x))

    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
    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
                ans_keys.remove(i)#去掉 ans_keys 里测试失败的密钥
                break

    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("##########")
msg=''
for i in range(0,len(arr)):
    msg=msg+chr(arr[i]^vigenerekeys[i%7][0])
print(msg)

```

```
Cryptography is the practice and study of techniques for, among other things, se
cure communication in the presence of attackers. Cryptography has been used for
hundreds, if not thousands, of years, but traditional cryptosystems were designe
d 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 kn
ow, and this exercise demonstrates, that it can be broken very easily.
>>>
```

所以得到明文为：

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.

重复一次一密的破译：

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

```python
# -*- coding: utf-8 -*-

import collections

# Seven different pieces of ciphertext encrypted with the same key
c1='BB3A65F6F0034FA957F6A767699CE7FABA855AFB4F2B520AEAD612944A801E'
c2='BA7F24F2A35357A05CB8A16762C5A6AAAC924AE6447F0608A3D11388569A1E'
c3='A67261BBB30651BA5CF6BA297ED0E7B4E9894AA95E300247F0C0028F409A1E'
c4='A57261F5F0004BA74CF4AA2979D9A6B7AC854DA95E305203EC8515954C9D0F'
c5='BB3A70F3B91D48E84DF0AB702ECFEEB5BC8C5DA94C301E0BECD241954C831E'
c6='A6726DE8F01A50E849EDBC6C7C9CF2B2A88E19FD423E0647ECCB04DD4C9D1E'
c7='BC7570BBBF1D46E85AF9AA6C7A9CEFA9E9825CFD5E3A0047F7CD009305A71E'

ciphertexts = [c1, c2, c3, c4, c5 ,c6 ,c7 ] #array of all ciphertexts
ciphertexts_counter=collections.Counter()

site_space=[]
key=[]

for current_index1,ciphertext1 in enumerate(ciphertexts):
    sure_site_space=[]
    counter = collections.Counter()
    print("对于密文",current_index1+1)
    for current_index2,ciphertext2 in enumerate(ciphertexts):
        if current_index1 != current_index2:
            print("密文",current_index1+1,"与密文",current_index2+1,"进行异或")
        for site_index in range(len(ciphertext1)):
            if site_index%2==0:
                result=int(ciphertext1[site_index],16)^int(ciphertext2[site_index],16)
                if (int(result)<=int('7',16) and int(result)>=int('4',16)):
                    counter[site_index]+=1
                    #print(site_index+1,ciphertext1[site_index],"^",ciphertext2[site_index],hex(result))
    for site_index,value in counter.items():
        if value >=4:
            sure_site_space.append(site_index/2+1)
    print("密文",current_index1+1,"space 位置",sure_site_space)

    for index in sure_site_space:
        if index not in site_space:
            site_space.append(index)
            cipher=ciphertext1[int(index*2)-2:int(index*2)]
            temp_key=int(cipher,16)^int('20',16)
            key.append(hex(temp_key))

print(site_space)
print(key)

for index in range(len(key)):
    print('密钥',site_space[index],"为",key[index])
```

得到结果如下：

```
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]
[2.0, 5.0, 14.0, 16.0, 23.0, 3.0, 6.0, 10.0, 15.0, 22.0, 25.0, 4.0, 12.0, 1'
20.0, 24.0, 26.0, 8.0, 13.0, 27.0, 19.0, 28.0, 29.0]
['0x1a', '0xd0', '0xbc', '0xda', '0x72', '0x4', '0x73', '0x98', '0x86', '0x5
'0x83', '0x9b', '0x9', '0xc9', '0x89', '0x67', '0xa5', '0xc8', '0xe', '0x61'
x39', '0xfd', '0x25']
密钥 2.0 为 0x1a
密钥 5.0 为 0xd0
密钥 14.0 为 0xbc
密钥 16.0 为 0xda
密钥 23.0 为 0x72
密钥 3.0 为 0x4
密钥 6.0 为 0x73
密钥 10.0 为 0x98
密钥 15.0 为 0x86
密钥 22.0 为 0x5f
密钥 25.0 为 0x83
密钥 4.0 为 0x9b
密钥 12.0 为 0x9
密钥 17.0 为 0xc9
密钥 20.0 为 0x89
密钥 24.0 为 0x67
密钥 26.0 为 0xa5
密钥 8.0 为 0xc8
密钥 13.0 为 0xe
密钥 27.0 为 0x61
密钥 19.0 为 0x39
密钥 28.0 为 0xfd
密钥 29.0 为 0x25
>>>
```

用得到的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获得其余密钥，得到解密结果：

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.