第一次作业

计15 宋驰 2021010797

一、古典密码部分

1、破解移位密码

通过穷举26种可能的移位方式来破解密码，代码在`task1.py`中

```
import math

alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
text = 'QGMEMKLTWLZWUZSFYWQGMOAKZLGKWWAFLZWOGJDV'

def decrypt(text, key):
    decrypted = ''
    for letter in text:
        if letter in alphabet:
            decrypted += alphabet[(alphabet.index(letter) - key) % 26]
        else:
            decrypted += letter
    print(key, decrypted)

for i in range(0, 26):
    decrypt(text, i)
```

输出结果：

```
(0, 'QGMEMKLTWLZWUZSFYWQGMOAKZLGKWWAFLZWOGJDV')
(1, 'PFLDLJKSVKYVTYREXVPFLNZJYKFJVVZEKYVNFICU')
(2, 'OEKCKIJRUJXUSXQDWUOEKMYIXJEIUUYDJXUMEHBT')
(3, 'NDJBJHIQTIWTRWPCVTNDJLXHWIDHTTXCIWTLDGAS')
(4, 'MCIAIGHPSHVSQVOBUSMCIKWGVHCGSSWBHVSKCFZR')
(5, 'LBHZHFGORGURPUNATRLBHJVFUGBFRRVAGURJBEYQ')
(6, 'KAGYGEFNQFTQOTMZSQKAGIUETFAEQQUZFTQIADXP')
(7, 'JZFXFDEMPESPNSLYRPJZFHTDSEZDPPTYESPHZCWO')
(8, 'IYEWECDLODROMRKXQOIYEGSCRDYCOOSXDROGYBVN')
(9, 'HXDVDBCKNCQNLQJWPNHXDFRBQCXBNNRWCQNFXAUM')
(10, 'GWCUCABJMBPMKPIVOMGWCEQAPBWAMMQVBPMEWZTL')
(11, 'FVBTBZAILAOLJOHUNLFVBDPZOAVZLLPUAOLDVYSK')
(12, 'EUASAYZHKZNKINGTMKEUACOYNZUYKKOTZNKCUXRJ')
(13, 'DTZRZXYGJYMJHMFSLJDTZBNXMYTXJJNSYMJBTWQI')
(14, 'CSYQYWXFIXLIGLERKICSYAMWLXSWIIMRXLIASVPH')
(15, 'BRXPXVWEHWKHFKDQJHBRXZLVKWRVHHLQWKHZRUOG')
(16, 'AQWOWUVDGVJGEJCPIGAQWYKUJVQUGGKPVJGYQTNF')
(17, 'ZPVNVTUCFUIFDIBOHFZPVXJTIUPTFFJOUIFXPSME')
(18, 'YOUMUSTBETHECHANGEYOUWISHTOSEEINTHEWORLD')
(19, 'XNTLTRSADSGDBGZMFDXNTVHRGSNRDDHMSGDVNQKC')
(20, 'WMSKSQRZCRFCAFYLECWMSUGQFRMQCCGLRFCUMPJB')
(21, 'VLRJRPQYBQEBZEXKDBVLRTFPEQLPBBFKQEBTLOIA')
(22, 'UKQIQOPXAPDAYDWJCAUKQSEODPKOAAEJPDASKNHZ')
(23, 'TJPHPNOWZOCZXCVIBZTJPRDNCOJNZZDIOCZRJMGY')
```

```
(24, 'SIOGOMNVYNBYWBUHAYSIOQCMBNIMYYCHNBYQILFX')
(25, 'RHNFNLMUXMAXVATGZXRHNPBLAMHLXXBGMAXPHKEW')
```

符合英文句子的是：`YOUMUSTBETHECHANGEYOUWISHTOSEEINTHEWORLD`，即`YOU MUST BE THE CHANGE YOU WISH TO SEE IN THE WORLD`，从密文到明文，需要向前移位18位，所以从明文到密文需要向后移位18位

2、破解代换密码

整体代码在`task2.py`中

先统计字母的出现概率：

```
text =
'AZLTBSUZWTCLBCDQFDSKASGDFWOGARTCDZFBSTZSBDABFDAZDBFYDUWOBCSQFWOGFSKLDWZFWUSQOBR
EZDADOBGDAWGDFRBCWFYDDOBCDFBSZLSKMCWBIRUCBYDAWEEDGAEWFFRAWEAZLTBSUZWTCLBCWBRFSKI
DBCSGFSKDOAZLTBRSOBCWBQFDTDOWOGTWTDZSZTDZCWTFFRITEDIDACWORAWEWRGFROBCDDWZELBMDOB
RDBCADOBQZLBCDROPDOBRSOSKASITEDJIDACWORAWEWOGDEDABZSIDACWORAWEIWACRODFFQACWFBCDD
ORUIWZSBSZIWACRODTZSPRGDGISZDFSTCRFBRAWBDGWOGDKKRARDOBIDWOFSKDOAZLTBRSOWOGBCDFQY
FDVQDOBROBZSGQABRSOSKDEDABZSORAFWOGASITQBROUCWFWEESMDGDEWYSZWBDFACDIDFSKFBREEUZD
WBDZASITEDJRBLISFBSKMCRACWZDDOBRZDELQOFQRBDGBSTDOWOGTWTDZBCDGDPDESTIDOBSKAZLTBSU
ZWTCLCWFYDDOTWZWEEDEDGYLBCDGDPDESTIDOBSKAZLTBWOWELFRFBCWBRFBCDYZDWNROUSKASGDFWOG
ARTCDZFBCDGRFASPDZLWOGWTTERAWBRSODWZELSOSKKZDVQDOALWOWELFRFBSBCDZDWGROUSKDOAZLTB
DGASIIQORAWBRSOFCWFSOSAAWFRSOWEBDZDGBCDASQZFDSKCRFBSZL'
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

def letter_frequency_count(text):
    letter_frequency = {}

    for letter in text:
        if letter in alphabet:
            if letter in letter_frequency:
                letter_frequency[letter] += 1
            else:
                letter_frequency[letter] = 1

    sorted_result = sorted(letter_frequency.items(), key=lambda x: x[1])

    for letter in sorted_result:
        print(letter[0], letter[1])

letter_frequency_count(text)
```

输出结果：

```
N 1
J 2
V 2
M 4
P 5
Y 8
U 11
Q 15
K 19
I 20
L 23
```

```
G 30
E 30
T 32
C 41
Z 44
A 47
R 47
F 48
O 56
S 60
W 64
B 67
D 97
```

可见D的出现频率最高，且显著高于其他字母，因此猜测D是e

统计双字母的出现概率：

```
def double_frequency_count(text):
    double_frequency = {}

    for i in range(len(text) - 2):
        if text[i] in alphabet and text[i + 1] in alphabet:
            double = text[i] + text[i + 1]
            if double in double_frequency:
                double_frequency[double] += 1
            else:
                double_frequency[double] = 1

    sorted_result = sorted(double_frequency.items(), key=lambda x: x[1])

    for double in sorted_result:
        print(double[0], double[1])

double_frequency_count(text)
```

输出结果为（此处省略了频率较低=1,2,3,4,5,6的三字母）：

```
LT 7
TB 7
WT 7
ZS 7
BD 7
FS 7
DW 7
FR 7
WF 7
RF 7
RS 7
DE 7
AZ 8
AS 8
ED 8
ID 8
AC 8
RO 8
DF 9
```

```
DZ 9
WB 9
ZL 10
GD 10
OG 10
ZD 10
AW 10
WE 10
RA 10
SO 10
FB 11
DA 11
DG 11
BS 12
CW 14
CD 15
OB 15
BR 15
SK 16
WO 17
DO 18
BC 19
```

这里最高概率的双字母之间差距较小，不容易确定`th`，因此稍微修改代码，使得检测双字母的时候，只检测双字母且后跟`D(即e)`的情况，输出结果为：

```
BF 1
AZ 1
KL 1
EZ 1
DA 1
BG 1
WG 1
BY 1
KI 1
DT 1
ZT 1
BM 1
BR 1
CA 1
OP 1
JI 1
SI 1
RG 1
SZ 1
AR 1
BI 1
YF 1
SM 1
AC 1
UZ 1
WZ 1
ZD 1
RZ 1
RB 1
ST 1
YZ 1
```

```
SP 1
SO 1
KZ 1
TB 1
EB 1
ZF 1
QF 2
SG 2
TC 2
YD 2
EE 2
WT 2
DI 2
CD 2
OG 2
RO 2
VQ 2
DP 2
TI 2
DZ 2
FY 3
TE 3
DE 3
WB 3
DG 3
SK 4
BC 12
```

因此可以猜测`BC`即`th`，随后再统计三字母出现的概率：

```
def triple_frequency_count(text):
    triple_frequency = {}

    for i in range(len(text) - 3):
        if text[i] in alphabet and text[i + 1] in alphabet and text[i + 2] in
alphabet:
            triple = text[i] + text[i + 1] + text[i + 2]
            if triple in triple_frequency:
                triple_frequency[triple] += 1
            else:
                triple_frequency[triple] = 1

    sorted_result = sorted(triple_frequency.items(), key=lambda x: x[1])

    for triple in sorted_result:
        print(triple[0], triple[1])

triple_frequency_count(text)
```

输出结果为（此处省略了频率较低=1,2,3的三字母）：

```
LBC 4
FWO 4
OBC 4
BCW 4
DDO 4
```

```
CWB 4
WBR 4
SKD 4
DOA 4
SOS 4
ASI 4
FBC 4
SKA 5
FBS 5
FSK 5
OBR 5
CWF 5
AWE 5
ACW 5
ORA 5
RFB 5
BRS 6
AZL 7
ZLT 7
LTB 7
RAW 7
RSO 7
WOG 10
DOB 10
BCD 12
```

由于已知BCD是the，则DOB是eOt，因此猜测O是n，由于字母不会加密为自己，因此G不是g，因此WOG不是ing，所以猜测WOG是and

此时将已经破译的字母填入原文本中，得到：

```
cZLTtSUZaThLtheQFeSKcSdeFandcRTheZFtSTZStectFecZetFYeUanthSQFandFSKLeaZFaUSQntRE
ZecentdecadeFRthaFYeentheFtSZLSKMhatIRUhtYecaEEedcEaFFRcaEcZLTtSUZaThLthatRFSKIe
thSdFSKencZLTtRSnthatQFeTenandTaTeZSZTeZhaTFFRITEeIechanRcaEaRdFRntheeaZELtMentR
ethcentQZLtheRnPentRSnSKcSITEeJIechanRcaEandeEectZSIechanRcaEIachRneFFQchaFtheen
RUIaZStSZIachRneTZSPRdedISZeFSThRFtRcatedandeKKRcRentIeanFSKencZLTtRSnandtheFQYF
eVQentRntZSdQctRSnSKeEectZSnRcFandcSITQtRnUhaFaEESMedeEaYSZateFcheIeFSKFtREEUZea
teZcSITEeJRtLISFtSKMhRchaZeentRZeELQnFQRtedtSTenandTaTeZthedePeESTIentSKcZLTtSUZ
aThLhaFYeenTaZaEEeEedYLthedePeESTIentSKcZLTtanaELFRFthatRFtheYZeaNRnUSKcSdeFandc
RTheZFthedRFcSPeZLandaTTERcatRSneaZELSnSKKZeVQencLanaELFRFtStheZeadRnUSKencZLTte
dcSIIQnRcatRSnFhaFSnSccaFRSnaEteZedthecSQZFeSKhRFtSZL
```

观察到字母串Zecentdecade中只有Z未被破译，通过断句猜测此处为Zecent decade，因此Z是r，根据语法知识，应该是decades，因此F是s，此时文本为：

```
crLTtSUraThLtheQseSKcSdesandcRTherstSTrStectsecretsYeUanthSQsandsSKLearsaUSQntRE
recentdecadesRthasYeenthestSrLSKMhatIRUhtYecaEEedcEassRcaEcrLTtSUraThLthatRsSKIe
thSdsSKencrLTtRSnthatQseTenandTaTerSrTerhaTssRITEeIechanRcaEaRdsRntheearELtMentR
ethcentQrLtheRnPentRSnSKcSITEeJIechanRcaEandeEectrSIechanRcaEIachRnessQchastheen
RUIarStSrIachRneTrSPRdedISresSThRstRcatedandeKKRcRentIeansSKencrLTtRSnandthesQYs
eVQentRntrSdQctRSnSKeEectrSnRcsandcSITQtRnUhasaEESMedeEaYSratescheIesSKstREEUrea
tercSITEeJRtLISstSKMhRchareentRreELQnsQRtedtSTenandTaTerthedePeESTIentSKcrLTtSUr
aThLhasYeenTaraEEeEedYLthedePeESTIentSKcrLTtanaELsRsthatRstheYreaNRnUSKcSdesandc
RThersthedRscSPerLandaTTERcatRSnearELSnSKKreVQencLanaELsRstSthereadRnUSKencrLTte
dcSIIQnRcatRSnshasSnSccasRSnaEteredthecSQrseSKhRstSrL
```

观察到字母串TrStectsecretsYeUanthSQsands，尝试断句为TrStect secrets YeUan thSQsands，因此猜测是短语protect secrets，所以T是p，S是o，猜测thSQsands是thousands，因此Q是u，此时文本为：

```
crLptoUraphLtheuseoKcodesandcRpherstoprotectsecretsYeUanthousandsoKLearsaUountRE
recentdecadesRthasYeenthestorLoKMhatIRUhtYecaEEedcEassRcaEcrLptoUraphLthatRsoKIe
thodsoKencrLptRonthatusepenandpaperorperhapssRIpEeIechanRcaEaRdsRntheearELtMentR
ethcenturLtheRnPentRonoKcoIpEeJIechanRcaEandeEectroIechanRcaEIachRnessuchastheen
RUIarotorIachRneproPRdedIoresophRstRcatedandeKKRcRentIeansoKencrLptRonandthesuYs
eVuentRntroductRonoKeEectronRcsandcoIputRnUhasaEEoMedeEaYorateschеIesoKstREEUrea
tercoIpEeJRtLIostoKMhRchareentRreELunsuRtedtopenandpaperthedePeEopIentoKcrLptoUr
aphLhasYeenparaEEeEedYLthedePeEopIentoKcrLptanaELsRsthatRstheYreaNRnUoKcodesandc
RphersthedRscoPerLandappERcatRonearELonoKKreVuencLanaELsRstothereadRnUoKencrLpte
dcoIIunRcatRonshasonoccasRonaEteredthecourseoKhRstorL
```

观察到字母串eEectronRcsand，断句为eEectronRcs and，易知E是l，R是i，此时文本为：

```
crLptoUraphLtheuseoKcodesandcipherstoprotectsecretsYeUanthousandsoKLearsaUountil
recentdecadesithasYeenthestorLoKMhatIiUhtYecalledclassicalcrLptoUraphLthatisoKIe
thodsoKencrLptionthatusepenandpaperorperhapssiIpleIechanicalaidsintheearlLtMenti
ethcenturLtheinPentionoKcoIpleJIechanicalandelectroIechanicalIachinessuchastheen
iUIarotorIachineproPidedIoresophisticatedandeKKicientIeansoKencrLptionandthesuYs
eVuentintroductionoKelectronicsandcoIputinUhasalloMedelaYorateschеIesoKstillUrea
tercoIpleJitLIostoKMhichareentirelLunsuitedtopenandpaperthedePelopIentoKcrLptoUr
aphLhasYeenparalleledYLthedePelopIentoKcrLptanalLsisthatistheYreaNinUoKcodesandc
iphersthediscoPerLandapplicationearlLonoKKreVuencLanalLsistothereadinUoKencrLpte
dcoIIunicationshasonoccasionalteredthecourseoKhistorL
```

观察到recentdecadesithasYeenthe断句为recent decades, it has Yeen the，因此Y是b，目前可以基本对前面几句文本进行断句，得到：

crLptoUraphL the use oK codes and ciphers to protect secrets beUanthousandsoKLearsaUo until recent decades，it has been the storLoKMhatIiUht be called classical crLptoUraphL that

可见文本主题是密码学，因此得到不止出现一次的crLptoUraphL是cryptography，oK是of，所以L是y，U是g，K是f，继续对后面文本断句，得到：

it has been the story of Mhat light be called classical cryptography that is of lethods of encryption that use pen and paper or perhaps

根据句意可得l是m，M是w，依次再推出其他字母，P是v，J是x，V是q，N是k，文本为：

cryptography the use of codes and ciphers to protect secrets began thousands of years ago until recent decades it has been the story of what might be called classical cryptography that is of methods of encryption that use pen and paper or perhaps simple mechanical aids in the early twentieth century the invention of complex mechanical and electromechanical machines such as the enigma rotor machine provided more sophisticated and efficient means of encryption and the subsequent introduction of electronics and computing has allowed elaborate schemes of still greater complexity most of which are entirely unsuited to pen and paper the development of cryptography has been paralleled by the development of cryptanalysis that is the breaking of codes and ciphers the discovery and application early on of frequency analysis to the reading of encrypted communications has on occasion altered the course of history

3、破解 Vigenere 密码

采用Kasiski test，先在文本中查找出现次数较多的三字母：

```
def triple_frequency_count(text):
    triple_frequency = {}

    for i in range(len(text) - 3):
        triple = text[i] + text[i + 1] + text[i + 2]
        if triple in triple_frequency:
            triple_frequency[triple] += 1
        else:
            triple_frequency[triple] = 1

    sorted_result = sorted(triple_frequency.items(), key=lambda x: x[1])

    for triple in sorted_result:
        print(triple[0], triple[1])

triple_frequency_count(text)
```

发现GUG出现4次，统计其在全文出现的位置：

```
for i in range(len(text) - 3):
    triple = text[i] + text[i + 1] + text[i + 2]
    if triple == 'GUG':
        print(i)
```

发现是：124、334、409、504，相邻数的差的最大公因数是5，因此密钥长度是5

将text分成y1-y5，对于每个yi，穷举26种加密下的重合指数，找到最接近0.065的结果：

```
key_length = 5
y1 = []
y2 = []
y3 = []
y4 = []
y5 = []

for i in range(len(text)):
    if i % key_length == 0:
        y1.append(text[i])
    elif i % key_length == 1:
        y2.append(text[i])
    elif i % key_length == 2:
        y3.append(text[i])
    elif i % key_length == 3:
        y4.append(text[i])
    elif i % key_length == 4:
        y5.append(text[i])

def frequency_count(text):
    frequency = {}

    for i in range(len(text)):
        if text[i] in frequency:
```

```
            frequency[text[i]] += 1
        else:
            frequency[text[i]] = 1

    sorted_result = dict(sorted(frequency.items(), key=lambda x: x[1]))
    return sorted_result

def cal_mg(yi):
    mg = [0] * 26

    sorted_result = frequency_count(yi)
    alphabet_frequency = []

    for i in range(26):
        if str(alphabet[i]) in sorted_result:
            alphabet_frequency.append(sorted_result[str(alphabet[i])])
        else:
            alphabet_frequency.append(0)

    for g in range(26):
        sum = 0.0
        for j in range(26):
            sum += frequency[j] * (alphabet_frequency[(j + g) % 26] / len(yi))
        mg[g] = round(sum, 5)

    print(mg)

cal_mg(y1)
cal_mg(y2)
cal_mg(y3)
cal_mg(y4)
cal_mg(y5)
```

输出结果为：

```
[0.0344, 0.0357, 0.0392, 0.0482, 0.0344, 0.0334, 0.0395, 0.0647, 0.0381, 0.0324,
0.0289, 0.0438, 0.0312, 0.036, 0.0342, 0.035, 0.0412, 0.0434, 0.0423, 0.039,
0.0443, 0.038, 0.0458, 0.0342, 0.0336, 0.0302]

[0.065, 0.0403, 0.0354, 0.0333, 0.0452, 0.0333, 0.0388, 0.0334, 0.0358, 0.0314,
0.0401, 0.0468, 0.0405, 0.0417, 0.0351, 0.0481, 0.0354, 0.036, 0.0324, 0.0399,
0.0304, 0.0359, 0.0394, 0.0341, 0.0339, 0.0394]

[0.0479, 0.0397, 0.0433, 0.0331, 0.0456, 0.0336, 0.0341, 0.0341, 0.0397, 0.0366,
0.0313, 0.0475, 0.0352, 0.0316, 0.0371, 0.0697, 0.0379, 0.0286, 0.028, 0.0501,
0.035, 0.0332, 0.0404, 0.0385, 0.0367, 0.0327]

[0.045, 0.0347, 0.043, 0.04, 0.0403, 0.033, 0.0335, 0.0344, 0.0338, 0.0386,
0.0363, 0.0505, 0.0346, 0.0359, 0.0387, 0.0648, 0.0342, 0.0313, 0.0307, 0.0463,
0.0277, 0.0337, 0.04, 0.0366, 0.0415, 0.0419]

[0.0349, 0.0308, 0.0485, 0.0314, 0.0346, 0.0297, 0.035, 0.0387, 0.0389, 0.0455,
0.0381, 0.0474, 0.0363, 0.0431, 0.0373, 0.0414, 0.0331, 0.036, 0.0339, 0.033,
0.0397, 0.0327, 0.0383, 0.0398, 0.0661, 0.0368]
```

y1中0.0647最接近，对应加密字母为H

y2中0.065最接近，对应加密字母为A

y3中0.0697最接近，对应加密字母为P

y4中0.0648最接近，对应加密字母为P

y5中0.0661最接近，对应加密字母为Y

因此密钥是HAPPY，翻译出明文：

```
def decrypt(text, key):
    result = ''
    for i in range(len(text)):
        result += alphabet[(alphabet.index(text[i]) - alphabet.index(key[i % 
len(key)])) % 26]
    print(result.swapcase())

decrypt(text, key)
```

明文：

we live in a world of search engines video games and electronic commerce every product and piece of information is at our fingertips this is stimulating an explosion of data centers where the hardware that makes all this possible operates these centers draw almost unimaginable amounts of power the power demand isnt going to slow any time soon the advent of cryptocurrency and generative ai is creating an exponential rise in demand for more data centers and more electricity we re witnessing a revolution in electric cars trucks and appliances electric companies will have to meet the demand while moving us toward a carbon neutral world

4、Enigma 密码机

（1）模拟Enigma密码机

- 模拟程序可以接受键盘输入的rotor_order、ring_setting、initial_position、turn_on_plug_board，对输入的text明文进行加密
- 主要的函数有：
    - encrypt：模拟整个Enigma密码机的加密过程，通过键盘输入的turn_on_plug_board来判断是否先对明文进行插线板的字母替换，后调用rotor_change和rotor两个转轮的函数，完成加密，最后也需要判断是否需要进行插线板的字母替换
    - rotor_change：用于根据当前的current_position得出下一字母加密所用的current_position，即key
    - rotor：三个转轮加密的主要函数，从接线板出来的字母依次通过三个转轮的加密，然后通过反射器，再反向通过三个转轮，最终得到加密后的字母，这一过程比较复杂，尤其是current_position和ring_setting两个变量的加减，以及逆向通过三个转轮时的加密过程，经过仔细推敲之后可以得到正确结果
- 经过测试，作业的几种测例都可以正常加密
- 注：运行代码时，输入上述可以变化的各项Enigma密码机的设置：

```
rotor order: 123
ring setting: AAA
initial position: AAA
turn on plug board? (y/n): N
text: ABCDEF
```

例如按照上述的输入，可以得到密文BJELRQ

（2）波兰人雷臼斯基方法破解Enigma

- 破解Enigma密码机采用的原理是，固定ring setting的前提下，遍历Enigma密码机所有的可能的初始状态和转子顺序，统计对于相同字母，从同样的初始状态和转子顺序下，分别加密1-6次得到的密文，从而统计对于每种初始状态和转子顺序，字母循环圈的情况，从第1次和第4次、第2次和第5次、第3次和第6次，各有对应的字母循环圈，统计各个循环圈的长度，最后进行记录，构造出一个字典，用于解密查找
- 随后，针对示例中加密1-6次的密文对应情况，计算其对应的字母循环圈，在字典中查找完全符合的字母循环圈，对于示例的情况而言，固定ring setting = DES的前提下，通过寻找字母循环圈可以得到唯一对应的初始状态和转子顺序，分别是AAA和231，这和题目中完全吻合
- 注：这里的代码在decrypt函数中，运行后可以得到输出：

```
[([2, 3, 1], [0, 0, 0])]
```

从而得到初始状态AAA和转子顺序231

二、完善保密加密

1. 密钥空间的密钥数量严格小于消息空间的消息数量时，可能实现完善保密加密吗？证明你的结论。

   不能，完善保密加密要求满足：Pr(P=p|C=c)=Pr(P=p)，就是在即使知道密文的前提下，攻击者也不能以更高的概率猜测出明文。

   由于密钥数量严格小于消息数量，因此|K| < |M|，如果明文空间大于密文空间的话，对于每一个密钥key，一定存在不同的明文加密成相同的密文，这样的话会导致无法解密。

   因此明文空间应当小于等于密文空间。这样我们对于一个固定的密文c，假设M(c)是所有能映射到密文c的明文m的集合。 每个从明文m加密到密文c的映射都需要一个不同的密钥key，否则就会出现对密文无法解密的情况，因此有|M(c)|≤|K|。

   所以有|M(c)|≤|K| < |M|，所以存在m ∈ M，m ∉ M(c)，即无论选择哪个密钥key，这个明文m都无法加密为密文c。 因此，对于这个m，有Pr(P=p|C=c)=0≠Pr(P=p)，这与完善保密加密的条件矛盾。所以这种方案不可能是完善保密加密的。

2. One time pad 在多次使用下是不安全的。给定十串明文m1, ..., m10，均为英文，用 ASCII编码。随机生成一个pad（密钥），用这个pad加密所有的明文得到密文c1,...,c10。加密方式为pad和明文逐字节异或得到密文。现在你知道所有的密文，恢复密钥，并利用密钥解密目标密文，最终恢复目标明文。

   由于空格对应的ASCII码是32，而大小写字母对应的ASCII码差值也是32，因此在二进制表示下，只有第6位不同，通过空格和字母异或，所得到的一定是仅仅发生大小写变化的字母。

   因此我们可以采取的破译方法是，让每一个密文c，都和其他所有的密文分别异或，从而得到10组异或后的密文，这10个得到的异或后的密文的第i位都是字母或者空格，我们可以极大概率猜测该密文c对应的明文的第i位是空格。

   这样我们找到空格的位置后，可以通过异或密文后的空格对应位置得到该位置的密钥，从而所有密文中该位对应的明文都可以破译出来，最后再通过人工句意补全就可以实现全部的破译。