应用密码学作业 #1

XX：202XX80XXXXXXXXX

2023 年 4 月 14 日

求 Hill 密码的密钥空间大小

Hill 密码中对加密矩阵的要求是，其在 F_{26} 上有逆矩阵。二阶 Hill 密码密钥空间大小的问题，也就是在模 26 的剩余类中，可逆的 $2*2$ 矩阵有多少个的问题。记模 k 的剩余类全体可逆的 $2*2$ 矩阵的集合为 $GL_2(Z_k)$，该集合中的元素的个数记为 $|GL_2(Z_k)|$。

我们首先考虑同余方程 $ab-cd\equiv n\ mod\ k$(其中 $(n,k)=1$) 的解的数量 (记为 $T_{k,n}$)。

首先可以证明：$T_{k,n}$ 是 k 的积性函数。(结论 1)

证明：设 $k=pq,(p,q)=1,(n,k)=1$。下面证明：$T_{k,n}=T_{p,n}T_{q,n}$。设 $a,b,c,d\in Z_p$, 且满足 $ab-cd\equiv n\ mod\ p; x,y,z,w\in Z_q$, 且 $xy-zw\equiv\ n\ mod\ q$。由孙子定理 (若 $m_1,m_2,\cdots,m_n$ 两两互质，则对于任意的 $a_1,a_2,\cdots,a_m$, 下述方程组 S 有解) 可知，存在唯一的 $r\in Z_{pq}$, 使得 $r\equiv a\ mod\ p, r\equiv x\ mod\ q$; 存

$$\begin{cases} x\equiv a_1\ mod\ m_1 \\ x\equiv a_2\ mod\ m_2 \\ \cdots \\ x\equiv a_n\ mod\ m_n \end{cases}$$

在唯一的 $s\in Z_{pq}$, 使得 $s\equiv b\ mod\ p, s\equiv y\ mod\ q$; 存在唯一的 t 和 u，使得 $t\equiv c\ mod\ p, t\equiv z\ mod\ q; u\equiv d\ mod\ p, u\equiv w\ mod\ q$。显然有序四元组 (r,s,t,u) 满足 $rs-tu\equiv n\ mod\ pq$。而 (r,s,t,u) 被 (a,b,c,d) 和 (x,y,z,w) 唯一确定，因此 $T_{k,n}=T_{p,n}T_{q,n}$。

下面证明：$T_{k,n}=k^3\prod_{p|k}(1-\frac{1}{p^2})$, 其中 p 是 k 的素因数。

证明：由结论 1，只需考虑 k 为素数幂时的情形。设 $k=p^e$，p 是素数，设 $a,b,c,d\in Z_k$, 且满足 $ab-cd\equiv n\ mod\ k$。分两种情形讨论：

(1) 当 (b,p)=1 时，$ab-cd\equiv n\ mod\ k$ 可化为 $a\equiv b_1(cd+d)\ mod\ k$, 其中 $bb_1\equiv 1\ mod\ k$, 在 Z_k 中，c 和 d 分别由 k 种取值；b_1 有 $\psi(k)$ 种取值；而无论 b_1,c,d 取何值，对于给定的 n，在 Z_k 中都唯一确定了 a 的值。因此，当 (b,p)=1 时，$T_{k,n}=\psi(k)kk=p^{3e}(1-\frac{1}{p})$

(2) 当 $(b,p)\neq 1$ 时，令 $b=p^rb_1, 1\leq r\leq e, (b_1,p)=1$。$ab-cd\equiv n\ mod\ k$ 可化为 $ap^r\equiv b_1^{-1}(cd+n)\ mod\ k$, 该同余式的解数为 $\psi(p^{e-r})\psi(p^e)p^{e-r}p^r=$

$\psi(p^{e-r})p^{2e}(1-\frac{1}{p})$, 理由如下：由 $1 \le r \le e$ 及 $(n,p)=1$ 知，$(cd,p)=1$。显然在 Z_k 中，b_1^{-1} 有 $\psi(p^{e-r})$ 种取值，c 有 $\psi(k)$ 种取值。又因为在 Z_{p^r} 中，若 c 确定了，则由 $cd+n \equiv 0 \ mod \ p^r$ 能唯一决定 d；因此在 Z_k 中，d 可以取 $\psi(p^{e-r})$ 个值。而当 b_1^{-1}, c, d 确定了，则 $a \equiv (b_1^{-1}(cd+n) \ mod \ p^r) mod \ p^{e-r}$ 也能唯一确定了 a 在 $Z_{p^{e-r}}$ 中的值，因此在 Z_k 中 a 可以取 p^r 个值。所以 $ap^r \equiv b_1^{-1}(cd+n) \ mod \ k$ 的解数为 $\psi(p^{e-r})\psi(p^e)p^{e-r}p^r = \psi(p^{e-r})p^{2e}(1-\frac{1}{p})$

将 $1 \le r \le e$ 的情况综合起来得

$$T_{k,n} = \sum_{r=1}^{n} \psi(p^{e-r})p^{2e}(1-\frac{1}{p}) = p^{3e-1}(1-\frac{1}{p})$$

综合 (1)、(2) 知：当 $k=p^e$ 时，$T_{k,n} = T_{p^e,n} = p^{3e-1}(1-\frac{1}{p})$

根据结论 1，可以知道，当 $k>1$ 时，$T_{k,n} = k^3 \prod_{p|k}(1-\frac{1}{p^2})$

由于 n 的可能取值有 $\psi(k)$ 个，所以所有满足形如 $ab-cd \equiv n \ mod \ k$ 的方程的解有 $\psi(k)k^3 \prod_{p|k}(1-\frac{1}{p^2}) = k^4 \prod_{p|k}(1-\frac{1}{p})(1-\frac{1}{p^2})$ 个。

令 $k=26$,则可得二阶 Hill 密码密钥空间的大小 $|GL_2(Z_{26})| = 26^4 \times \frac{1}{2} \times \frac{3}{4} \times \frac{12}{13} \times \frac{168}{169} = 157248$

替换密码的破解

密文为:emglosudcgdncuswysfhnsfcykdpumlwgyicoxysipj kqpkugkmgolicginc-gacksnisacykzsckxecjckshysxcgoidp kzcnkshicgiwygkkgkgoldsilkgoiusigledsp-wzugfzccndgy ysfuszcnxeojncgyeoweupxezgacgnfglknsacigoiyckxcjuc iuzcfzc-cndgyysfeuekuzcsocfzccnc，求明文。

经过对密文中的单个字母、双字母组合、三字母组合等的统计，由于缺少单词之间的空格，无法分辨是不是一个具体的单词，我觉得使用单纯的频率分析很难破解出密文。通过参考使用四元字母组合统计数据衡量英文匹配度和简单替换密码的破解分析，我根据这两个网址上面的方法来破解密文。

首先说明如何衡量一串字符串 text 是否是英文的标准，我们称这个标准为匹配度 (fitness)。通过对某个字符串中所有的四元字母组合 (quad) 进行一个评分 (fitness_score(quad))，并将它们所得的分数进行相加，得到判断 text 是否是一个英文文本的总分 sum_score。

$sum\_score = \sum_{i=0}^{|text|-4} fitness\_score(text[i:i+4])$

其中 $|text|$ 表示字符串 $text$ 的长度，$text[i:i+4]$ 表示字符串第 $i, i+1, i+2, i+3$ 字符组成的子串。

需要进一步说明的是如何对一个四元组 $quad$ 进行评分，评分函数 $fitness\_score(quad)$ 如下:

$$fitness\_score(quad) = \begin{cases} \lg \frac{count(quad)}{N} & \text{quad in quadgrams.txt} \\ \lg \frac{0.01}{N} & \text{quad not in quadgrams.txt} \end{cases} \tag{1}$$

其中 quadgrams.txt 文件中包含将近 39 万行，每行都是一个四元字母组合 quad 及该四元字母组合在所统计的几个英文文集中所出现次数 $count(quad)$。N 为所有字母组合 quad 出现的次数之和, 即 N=$\sum_{quad\ in\ english\ corpus} count(quad)$。

上述对某文本对于英文文本的匹配度的算法可通过单击“此网址”获得。

有了上述对某个字符串文本是否是英文文本的判断标准。我们便可以使用爬山 (Hill climbing) 算法来破解替换密码。爬山算法的步骤如下：

(1) 生成一个随机密钥 parent，使用密钥 parent 对密文进行解密。计算并记录对解密后的文本对于英文文本的匹配度分数。

(2) 对密钥 parent 稍加修改 (例如随机地交换其中两个字符的位置)，得到一个新的密钥，使用该新密钥解密并计算解密文本的匹配度。

(3) 若新密钥下的匹配度高于 parent 密钥，则将当前的新密钥作为 parent 密钥。

(4) 回到步骤 (2)，若在最近 1000 次迭代中，没有出现更好的匹配度，则结束。

如果按照上述方法没有恢复出合适的明文，则说明算法陷入了局部最优当中，对密钥简单的修改也不能跳出这个局部最优，此时重新运行算法，直到恢复出可读的明文为止。

根据上述思路，再利用 python 的 pycipher 库编写的破解替换密码的 python3 代码如下：

```
from pycipher import SimpleSubstitution as SimpleSub
import random
import re
from ngram_score import ngram_score
```

```
fitness = ngram_score('quadgrams.txt') # load our quadgram statistics

ctext="""emglosudcgdncuswysfhnsfcykdpumlwgyicoxysipj
kqpkugkmgolicgincgacksnisacykzsckxecjckshysxcgoidp
kzcnkshicgiwygkkgkgoldsilkgoiusigledspwzugfzccndgy
ysfuszcnxeojncgyeoweupxezgacgnfglknsacigoiyckxcjuc
iuzcfzccndgyysfeuekuzcsocfzccnc"""
ctext = re.sub('[^A-Z]','',ctext.upper())

maxkey = list('ABCDEFGHIJKLMNOPQRSTUVWXYZ')
maxscore = -99e9
parentscore,parentkey = maxscore,maxkey[:]
print ("Substitution Cipher solver, you may have to wait several iterations"
    )
print ("for the correct result. Press ctrl+c to exit program.")
# keep going until we are killed by the user
i = 0
while 1:
    i = i+1
    random.shuffle(parentkey)
    deciphered = SimpleSub(parentkey).decipher(ctext)
    parentscore = fitness.score(deciphered)
    count = 0
    while count < 1000:
        a = random.randint(0,25)
        b = random.randint(0,25)
        child = parentkey[:]
        # swap two characters in the child
        child[a],child[b] = child[b],child[a]
        deciphered = SimpleSub(child).decipher(ctext)
        score = fitness.score(deciphered)
        # if the child was better, replace the parent with it
        if score > parentscore:
            parentscore = score
            parentkey = child[:]
            count = 0
        count = count+1
    # keep track of best score seen so far
    if parentscore>maxscore:
        maxscore,maxkey = parentscore,parentkey[:]
        print ('\nbest score so far:',maxscore,'on iteration',i)
        ss = SimpleSub(maxkey)
        print ('    best key: '+''.join(maxkey))
        print ('    plaintext: '+ss.decipher(ctext).lower())
```

break_simplesub.py

运行结果如下：

```
Substitution Cipher solver, you may have to wait several iterations
for the correct result. Press ctrl+c to exit program.

best score so far: -1294.7300255274895 on iteration 1
    best key: GHXUIMNEZTAYPCJWBLSODVFQKR
    plaintext: hfartsdunaugndsplswbgswnlyumdfrpalentcl
    semonyxmydayfatrenaegnaknysgesknlyisnychnonysblscn
    ateumyingysbenaeplayyayatruseryatedsearhusmpidawin
    nguallswdsingchtognalhtphdmchiaknagwarygskneatelny
    cnodnedinwinnguallswhdhydinstnwinngn

best score so far: -1233.748081808087 on iteration 2
    best key: UPODKXHLCTAEZNIMBSGYFJQRWV
    plaintext: lpshcradisdniarytrugnruitedbaphystoicftr
    obviewbeasepschoisoniskiernorkitemriefliviergtrfisc
    odbeminergoisoytseeseschdrohescoaroshldrbymasumiind
    sttruarminflcvnistlcylabflmskisnushenrkioscotiefiva
    ioamiumiindsttrulaleamirciumiini

best score so far: -972.1824432505555 on iteration 3
    best key: GDJICHWZEQRNMOSXBYKUPAFTLV
    plaintext: imaynotbeabletogrowflowersbutmygardenpro
    ducesjustasmanydeadleavesoldovershoespiecesofropean
    dbushelsofdeadgrassasanybodysandtodayiboughtawheelb
    arrowtohelpinclearingitupihavealwayslovedandrespect
    edthewheelbarrowitistheonewheele
```

result

所以破解出来的密文是:i may not be able to grow flowers but my garden produces just as many dead leaves old overshoes pieces of rope and bushels of dead grass as any body sand today i bought a wheel barrow to help in clearing it up i have always loved and respected the wheel barrow it is the one wheel e

维吉尼亚密码的破解

密文为 KCCPKBGUFDPHQTYAVINRRTMVGRKDNBV
FDETDGILTXRGUDDKOTFMBPVGEGLTGCKQRACQ
CWDNAWCRXIZAKFTLEWRPTYCQKYVXCHKFTPON
CQQRHJVAJUWETMCMSPKQDYHJVDAHCTRLSVSK

CGCZQQDZXGSFRLSWCWSJTBHAFSIASPRJAHKJ
RJUMVGKMITZHFPDISPZLVLGWTFPLKKEBDPGC
EBSHCTJRWXBAFSPEZQNRWXCVYCGAONWDDKAC
KAWBBIKFTIOVKCGGHJVLNHIFFSQESVYCLACN
VRWBBIREPBBVFEXOSCDYGZWPFDTKFQIYCWHJ
VLNHIQIBTKHJVNPIST，求明文。

为了破解维吉尼亚密码密码的密钥，首先需要确定密钥的长度。而确定密钥长度的方法就是将密文处理 n 次，然后第 i 次计算密文被分成了 i 组的密文对应的英文文本重合指数 (coincidence index) 的平均值 CI_i。然后对计算出来 CI 的值进行排名，对排在前几名的 i 值求它们的最大公因数。

如下是将密文分为 2-50 组后计算出平均重合指数 CI_i，并列出与英文文本重合指数的统计值最接近的 8 个可能密钥长度值得 python 代码：

```
def c_alpha(cipher):
    cipher_alpha = ''
    for i in range(len(cipher)):
        if (cipher[i].isalpha()):
            cipher_alpha += cipher[i]
    return cipher_alpha

def count_CI(cipher):
    N = [0.0 for i in range(26)]
    cipher = c_alpha(cipher)
    L = len(cipher)
    if cipher == '':
        return 0
    else:
        for i in range(L):
            if (cipher[i].islower()):
                N[ord(cipher[i]) - ord('a')] += 1
            else:
                N[ord(cipher[i]) - ord('A')] += 1
    CI_1 = 0
    for i in range(26):
        CI_1 += ((N[i] / L) * ((N[i]-1) / (L-1)))
    return CI_1

def count_key_len_CI(cipher,key_len):
    un_cip = ['' for i in range(key_len)]
    aver_CI = 0.0
    count = 0
    for i in range(len(cipher_alpha)):
        z = i % key_len
```

```
        un_cip[z] += cipher_alpha[i]
    for i in range(key_len):
        un_cip[i]= count_CI(un_cip[i])
        aver_CI += un_cip[i]
    aver_CI = aver_CI/len(un_cip)
    return aver_CI

def pre_10(cipher):
    M = [(1,0,0)]+[(0,0.0,0.0) for i in range(49)]
    for i in range(2,50):
        M[i] = (i,count_key_len_CI(cipher,i),abs(0.065 - count_key_len_CI(
            cipher,i)))
    M = sorted(M,key = lambda x:x[2])
    for i in range(2,10):
        print (M[i])
```

运行结果为：

(密钥长度，重合指数，差值)

(42, 0.06632653061224489, 0.0013265306122448861)

(6, 0.06282372598162073, 0.002176274018379276)

(18, 0.06199594847685662, 0.0030040515231433834)

(12, 0.06040564373897706, 0.00459435626102294)

(24, 0.05680708180708182, 0.008192918192918182)

(36, 0.055709876543209864, 0.009290123456790138)

(30, 0.05464646464646463, 0.010353535353535372)

(33, 0.05374961738598101, 0.011250382614018992)

根据该结果，可以看出密钥长度大多是 6 的倍数，所以可以猜测密钥长度是 6。

接下来将密文分成六组，每组分别用 26 个字母作为密钥进行移位解密，统计解密后文本的重合指数，列举出每组重合指数最接近真实值的 5 个密钥值。从而破解出各组的密钥，恢复出原始密钥。

```
def count_CI2(cipher,n):
    N = [0.0 for i in range(26)]
    cipher = c_alpha(cipher)
    L = len(cipher)
    for i in range(L):
        if (cipher[i].islower()):
            N[(ord(cipher[i]) - ord('a') - n)%26] += 1
        else:
```

```
            N[(ord(cipher[i]) - ord('A') - n)%26] += 1
    CI_2 = 0
    for i in range(26):
        CI_2 += ((N[i] / L) * F[i])
    return CI_2

def one_key(cipher,key_len):
    un_cip = ['' for i in range(key_len)]
    cipher_alpha = c_alpha(cipher)
    for i in range(len(cipher_alpha)):
        z = i % key_len
        un_cip[z] += cipher_alpha[i]
    for i in range(key_len):
        print (i)
        pre_5_key(un_cip[i])
def pre_5_key(cipher):
    M = [(0,0.0) for i in range(26)]
    for i in range(26):
        M[i] = (chr(ord('a')+i),abs(0.065 - count_CI2(cipher,i)))
    M = sorted(M,key = lambda x:x[1])

    for i in range(5):
        print (M[i])
key_len = 6
one_key(cipher,key_len)
```

key.py

运行结果为：

0

('c', 0.01263007719298246)

('p', 0.02732719649122807)

('j', 0.02775484210526316)

('w', 0.029796789473684214)

('f', 0.030771347368421062)

1

('r', 0.007946733928571433)

('c', 0.025942100000000003)

('n', 0.026191291071428584)

('y', 0.027912773214285716)

('g', 0.02823635357142857)

2

('y', 0.017162135714285702)
('l', 0.02803440357142857)
('z', 0.028141323214285717)
('x', 0.02984061250000001)
('n', 0.03027679821428572)
3
('p', 0.011576453571428565)
('l', 0.023508191071428576)
('a', 0.028578116071428572)
('c', 0.02927103035714286)
('z', 0.029286649999999997)
4
('t', 0.01998572678571428)
('i', 0.02725404107142857)
('h', 0.02772016428571429)
('e', 0.02878623928571429)
('x', 0.029106814285714287)
5
('o', 0.007908455357142852)
('z', 0.026214791071428573)
('k', 0.026409244642857148)
('p', 0.030399873214285722)
('d', 0.031096314285714292)

可以看出来密钥是 crypto, 用该密钥对密文进行解密，便可以得到明文。利用 pycipher 库对密文进行解密的代码如下：

```
from pycipher import Vigenere
key="crypto"
print(Vigenere(key).decipher(cipher))
```

decipher.py

运行结果为：

ILEARNEDHOWTOCALCULATETHEAMOUNTOFPAPE
RNEEDEDFORAROOMWHENIWASATSCHOOLYOUMUL

TIPLYTHESQUAREFOOTAGEOFTHEWALLSBYTHEC
UBICCONTENTSOFTHEFLOORANDCEILINGCOMBI
NEDANDDOUBLEITYOUTHENALLOWHALFTHETOTA
LFOROPENINGSSUCHASWINDOWSANDDOORSTHEN
YOUALLOWTHEOTHERHALFFORMATCHINGTHEPAT
TERNTHENYOUDOUBLETHEWHOLETHINGAGAINTO
GIVEAMARGINOFERRORANDTHENYOUORDERTHEP
APER 换成小写，加上空格为：i learned how to calculate the amount of paper needed for a room when i was at school you multiply the square foot age of the walls by the cubic contents of the floor and ceiling combined and double it you then allow half the total for openings such as windows and doors then you allow the other half for matching the pattern then you double the whole thing again to give a margin of error and then you order the paper