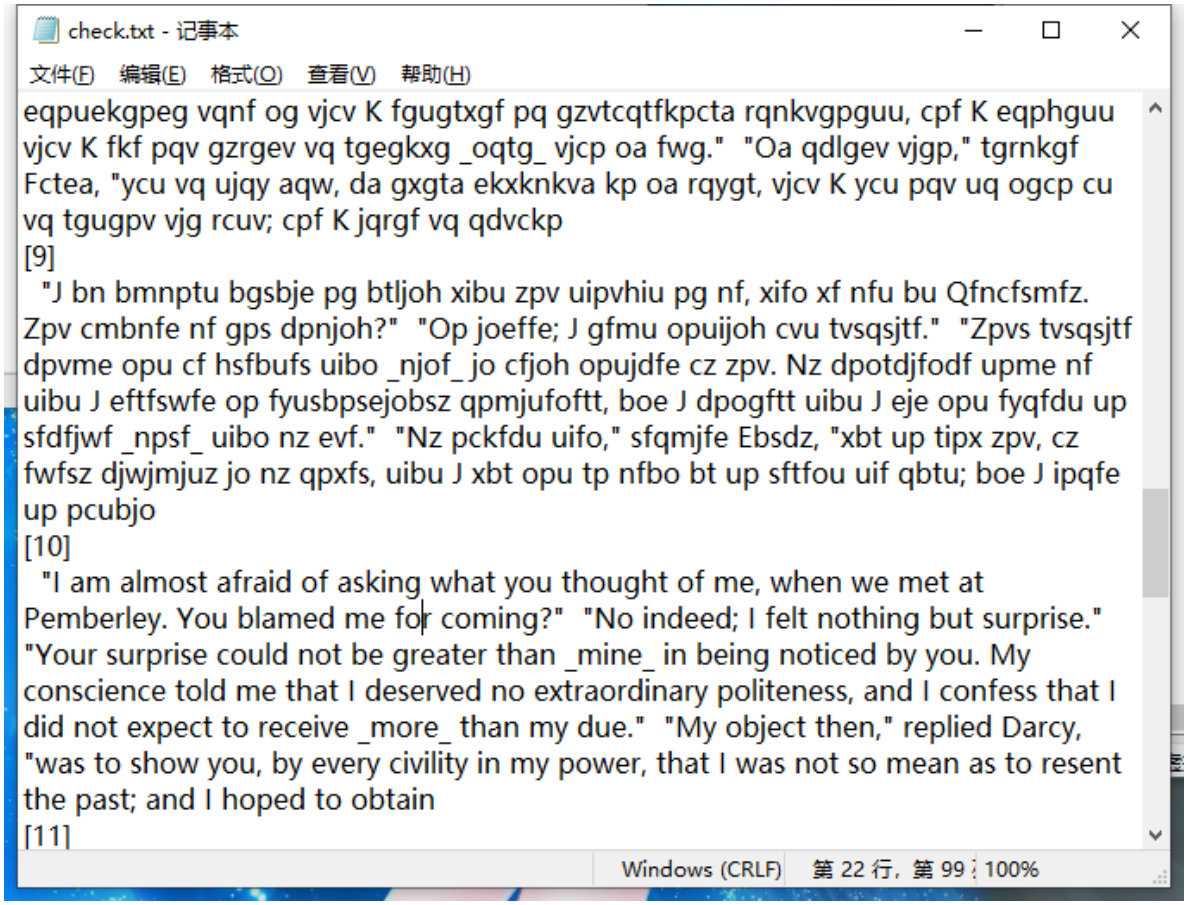
密码学实验报告

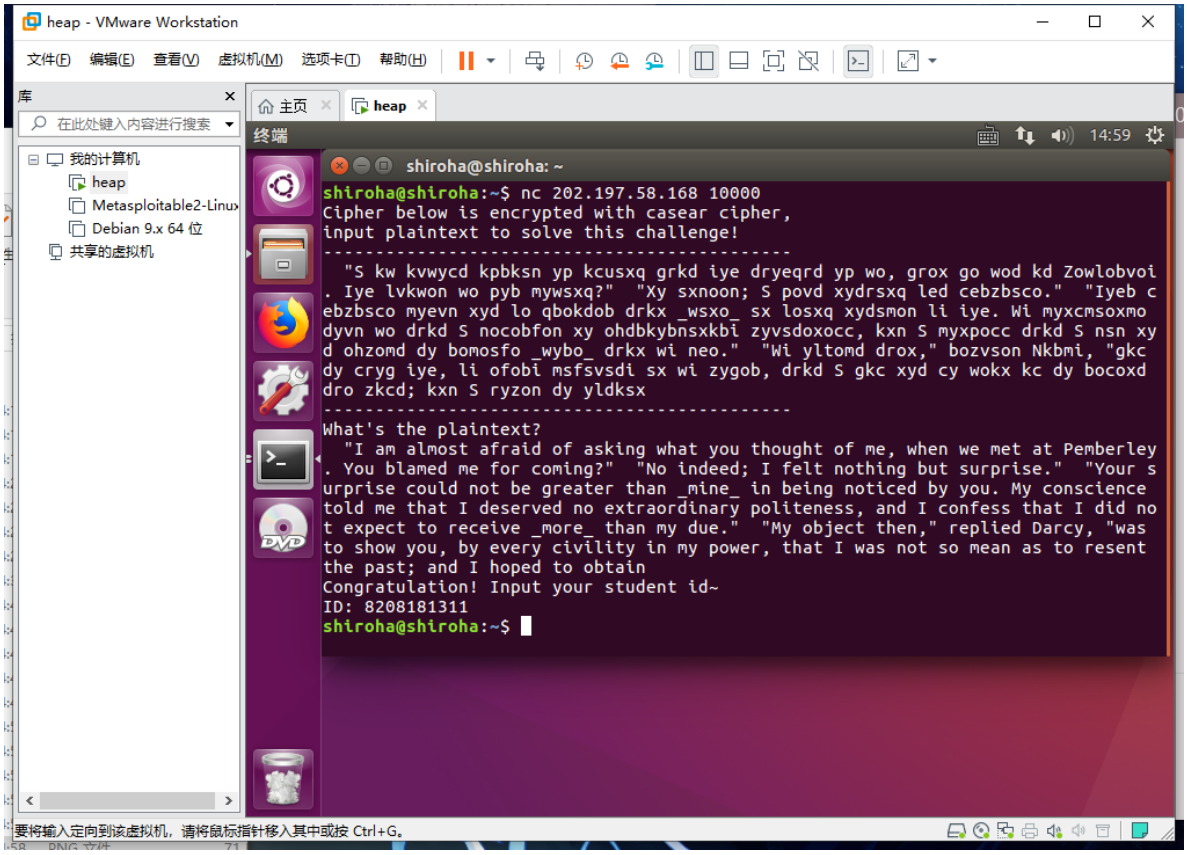
古典密码学

移位密码

```
s = input()
file = open("check.txt", "a")
for i in range(26):
    plain = ""
    for j in s:
        if j.islower():
            plain += chr((ord(j) - ord('a') - i) % 26 + ord('a'))
        elif j.isupper():
            plain += chr((ord(j) - ord('A') - i) % 26 + ord('A'))
        else:
            plain += j
    file.write('[' + str(i) + ']\n' + plain + '\n')
```



输入密文

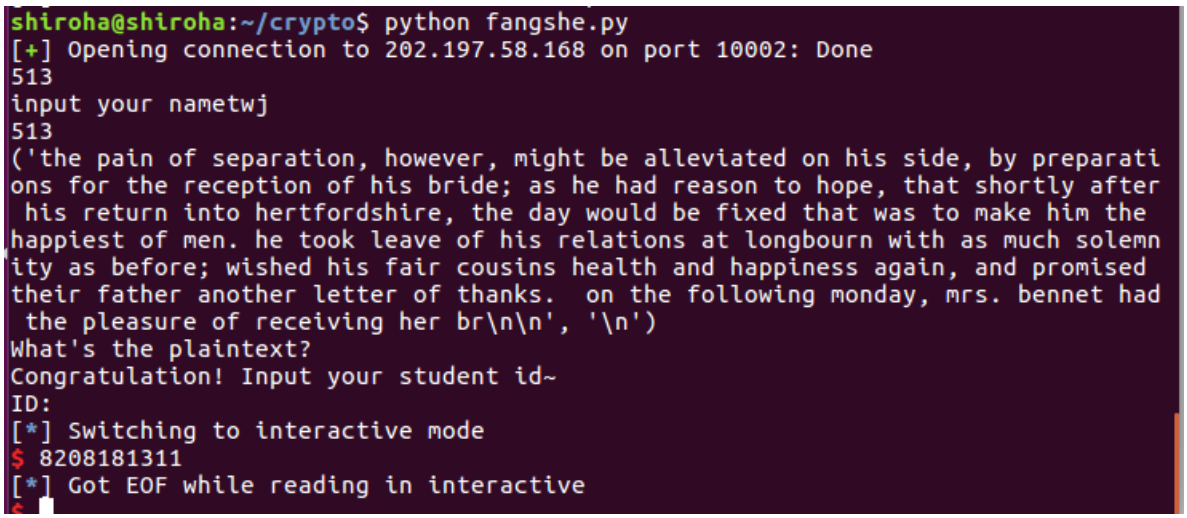


仿射密码

```
from pwn import *

sh = remote("202.197.58.168","10002")
#context.log_level = 'debug'
sh.recvuntil("-----------------------------------------\n")
a = sh.recvuntil("-----------------------------------------\n").replace("---
-----------------------------------------","")

print len(a)
maps = "abcdefghijklmnopqrstuvwxyz ,"
lenth = len(a)
name = (raw_input("input your name"))
s = ""
for j in range(1,29):
    for k in range(0,29):
        s = ""
        for i in range(lenth):
            d = a[i]
            if d in maps:
                d = maps.index(d)
                d = (j * (d - (k))) % 29
                d = maps[d]
            s += d
        if " the" in s:
            print len(s)
            sh.send(s)
            print(s,'\n')
            print sh.recvuntil("ID: ")
            sh.sendline((name))
```

```
sh.interactive()
```

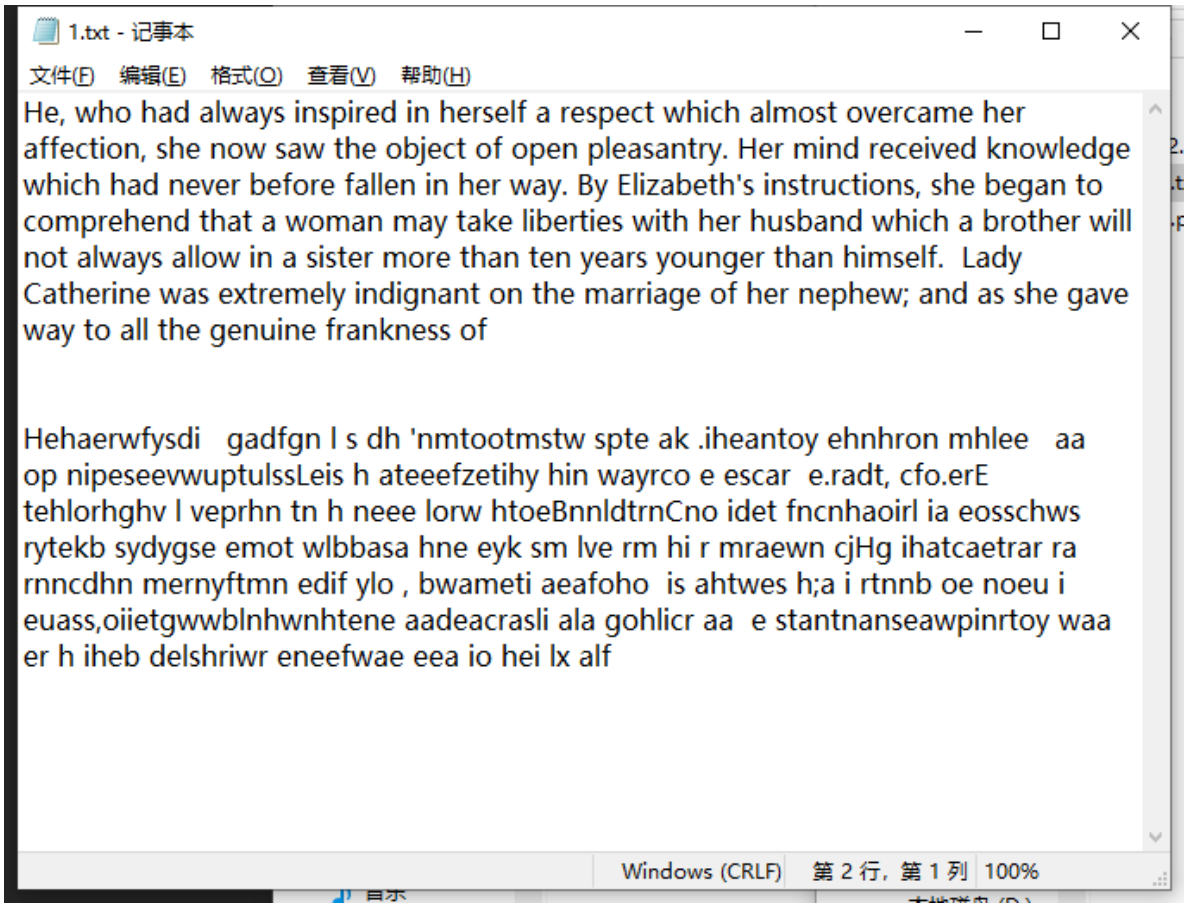
这里用'the '作为高频词判断，所以可能有时候文章里面没有the ， 多试几次就好了/wn,总会有the的



列移位密码

```
import numpy as np
f = open("1.txt","w")
s = input()
lenth = len(s)
for i in range(2,int(lenth / 2)):
    c = []
    for j in s:
        c.append(j)
    b = c
    #print(i)
    d = int(lenth / i)
    if (lenth % i != 0):
        continue
    le = len(b)
    a = np.array(b)
    a = a.reshape((i,d))
    a = a.T
    a = a.flatten()
    li = ''.join(str(i) for i in a)
    for i in li:
        f.write(i)
    f.write('\n\n\n')
```

```
shiroha@shiroha:~$ nc 202.197.58.168 10001
Cipher below is encrypted with row transposition cipher,
input plaintext to solve this challenge!
--------------------------------------------
Hhdane e ehatr at naeefnarencdwecdeffnhayzhncsea rdtomaiiies hrrl ylni   y ghifa
aiatldn maf en gwo gnaseo ysdhlrcil chfisow c  syrde l h roa ey a'st, nce  makbe
trbw o  aso smtteyeam.dtnsryittag nwdsaa teens, asp efetcmoaefohw otopa.  ikew n
 rlir.Ebsti b ohtaayeesh ahatwnl watoheaorns yhe e g hrehe; hvyahn k  hl iir s h
ovmrene tb pln mrvndhhebeln  le rosetmeh n  r  hni hiowa  eranru  e  e eminoer e
p aee leufnowawirnsapw see c, shjoeetHieeogiave e wBitiunhgopnaw tltwhudcbeltali
sren snthlLCrwxenan iorhas  tl iref
--------------------------------------------
What's the plaintext?
```



第一条就是

```
What's the plaintext?
He, who had always inspired in herself a respect which almost overcame her affec
tion, she now saw the object of open pleasantry. Her mind received knowledge whi
ch had never before fallen in her way. By Elizabeth's instructions, she began to
 comprehend that a woman may take liberties with her husband which a brother wil
l not always allow in a sister more than ten years younger than himself.  Lady C
atherine was extremely indignant on the marriage of her nephew; and as she gave
way to all the genuine frankness of
Congratulation! Input your student id~
ID: 8208181311
shiroha@shiroha:~$
```

维吉尼亚密码

重合指数

定义 2.2 令 $x = x_1x_2\cdots x_n$ 为长度为 n 的字母字串，定义此字串 x 的吻合指数（Index of Coincidence）为任取此字串两字母为相同的几率，即

$$I(x) = \frac{\sum_{i=1}^{26}\binom{p_i}{2}}{\binom{n}{2}}$$

其中 $p_1, p_2, \cdots, p_{26}$ 代表字母 $a, b, c, \cdots z$ 在字串 x 中所出现的次数。

假设 x 为一篇普通的英文文章，且 n 很大，则

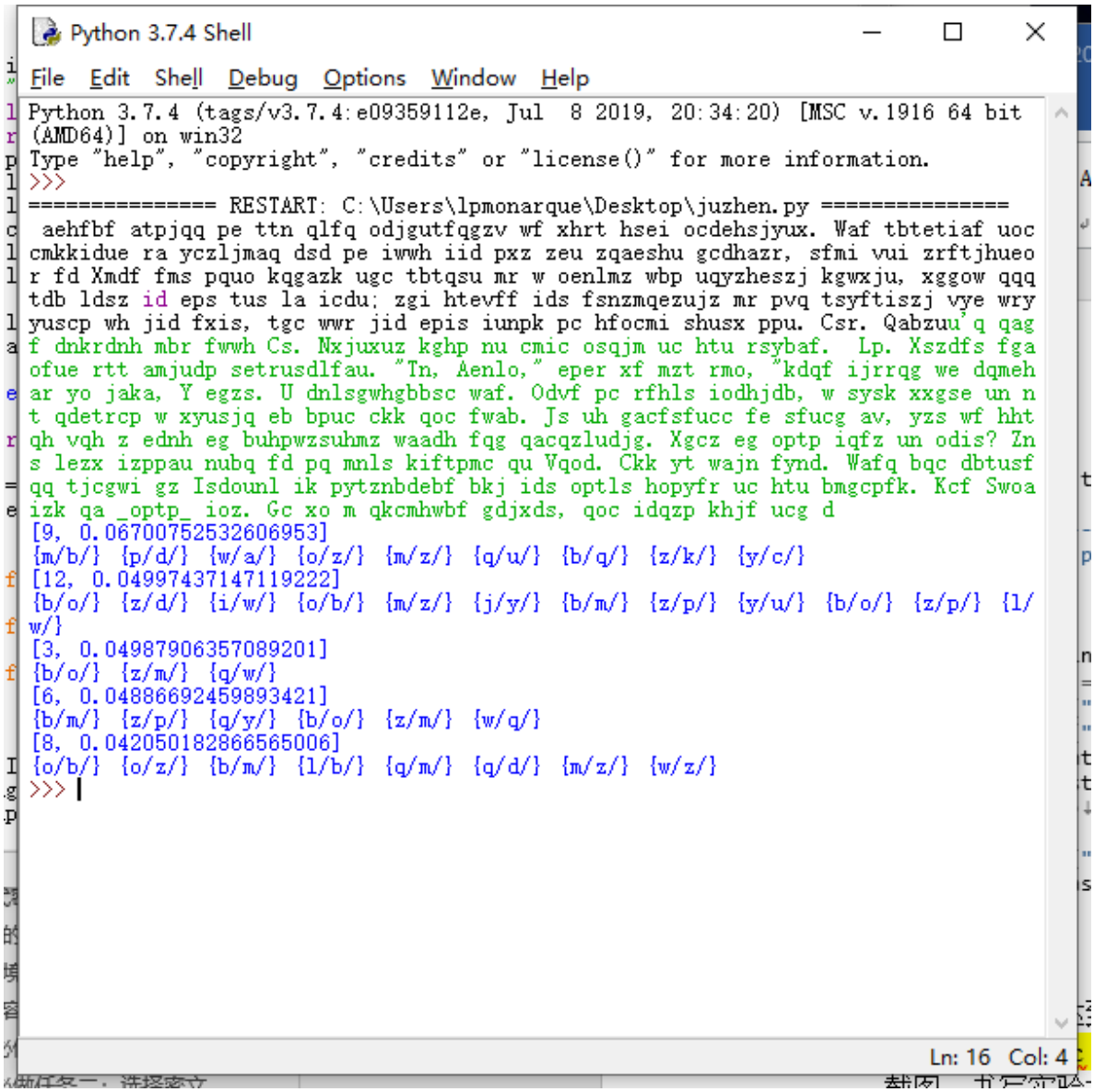
$$I(x) \approx |W| = \sqrt{w_1^2 + w_2^2 + \cdots + w_{26}^2} \approx 0.066$$

代码是某网站找到的(重合指数好难算.jpg)

```
import threading

key_len = [] #重合指数对比结果，单元格式[length, CI]
std_rate = [0.08167, 0.01492, 0.02782, 0.04253, 0.12702, 0.02228, 0.02015,
0.06094, 0.06966, 0.00153, 0.00772, 0.04025, 0.02406, 0.06749, 0.07507, 0.01929,
0.00095, 0.05987, 0.06327, 0.09056, 0.02758, 0.00978, 0.02360, 0.00150, 0.01974,
0.00074]
#字母频率标准

def decrypt(cipher, key):
    plain = ""
    length = len(key)
    for i in range(len(cipher)):
        if cipher[i].islower():
            plain_asc = (ord(cipher[i]) - ord(key[i % length])) % 26 + ord('a')
            plain += chr(plain_asc)
        elif cipher[i].isupper():
            plain_asc = (ord(cipher[i]) - ord('A') - ord(key[i % length]) +
ord('a')) % 26 + ord('A')
            plain += chr(plain_asc)
        else:
            plain += cipher[i]
    return plain

def get_key_len(cipher, length):
    CI = 0
    for i in range(length):
        tmp = cipher[i::length]
        rate = [0] * 26
        tmp_len = 0
        for j in tmp:
            k = -1
            if j.islower():
                k = ord(j) - ord('a')
            if j.isupper():
                k = ord(j) - ord('A')
            if k != -1:
                rate[k] += 1
                tmp_len += 1
        for j in range(26):
            CI += rate[j] / tmp_len * (rate[j] - 1) / (tmp_len - 1)
    CI /= length #取当前猜测长度下的平均CI
    key_len.append([length, CI])
```

```python
def get_key(cipher, length, std_CI2):
    delta = [[0, 0]] * length
    for i in range(length):
        delta_CI2 = []
        tmp = cipher[i::length]
        rate = [0] * 26
        tmp_len = 0
        for j in tmp:
            k = -1
            if j.islower():
                k = ord(j) - ord('a')
            if j.isupper():
                k = ord(j) - ord('A')
            if k != -1:
                rate[k] += 1
                tmp_len += 1
        tmp_len = len(tmp)
        for j in range(26): #移位=j
            CI2 = 0
            for k in range(26):
                CI2 += std_rate[k] * rate[(k + j) % 26] / tmp_len
            delta_CI2.append([j, abs(CI2 - std_CI2)])
        delta_CI2.sort(key = lambda k: k[1])
        delta[i] = [delta_CI2[0][0], delta_CI2[1][0]]
    return delta

def check_key(delta, length, depth, cur, cipher, file):
    if length == (depth - 1):
        for i in range(2):
            _cur = cur.copy()
            _cur.append(chr(delta[length][i] + ord('a')))
            file.write(decrypt(cipher, _cur) + '\n')
    else:
        for i in range(2):
            _cur = cur.copy()
            _cur.append(chr(delta[length][i] + ord('a')))
            check_key(delta, length + 1, depth, _cur, cipher, file)

if __name__ == '__main__':
    cipher = input()
    threads = []
    for i in range(2, 14):
        t = threading.Thread(target = get_key_len, args = (cipher, i))
        threads.append(t)
    for t in threads:
        t.start()
    for t in threads:
        t.join()
    std_CI2 = 0
    for i in std_rate:
        std_CI2 += i * i
    key_len.sort(key = lambda k: abs(k[1] - std_CI2)) #按CI排序
    file = open("check.txt", "a")
    for i in range(5):
        print(key_len[i])
        delta = get_key(cipher, key_len[i][0], std_CI2)
        check_key(delta, 0, len(delta), [], cipher, file)
        for j in delta:
```

```
            print('{', end = '')
            for k in j:
                print(chr(k + ord('a')), end = '/')
            print('}', end = ' ')
        print()
```



密钥就是mpwomqbzy

```
shiroha@shiroha:~$ nc 202.197.58.168 10004
Cipher below is encrypted with vigeniere cipher,
input key to solve this challenge!
-------------------------------------------
 aehfbf atpjqq pe ttn qlfq odjgutfqgzv wf xhrt hsei ocdehsjyux. Waf tbtetiaf uoc
cmkkidue ra yczljmaq dsd pe iwwh iid pxz zeu zqaeshu gcdhazr, sfmi vui zrftjhueo
r fd Xmdf fms pquo kqgazk ugc tbtqsu mr w oenlmz wbp uqyzheszj kgwxju, xggow qqq
tdb ldsz id eps tus la icdu; zgi htevff ids fsnzmqezujz mr pvq tsyftiszj vye wry
yuscp wh jid fxis, tgc wwr jid epis iunpk pc hfocmi shusx ppu. Csr. Qabzuu'q qag
f dnkrdnh mbr fwwh Cs. Nxjuxuz kghp nu cmic osqjm uc htu rsybaf.  Lp. Xszdfs fga
ofue rtt amjudp setrusdlfau. "Tn, Aenlo," eper xf mzt rmo, "kdqf ijrrqg we dqmeh
ar yo jaka, Y egzs. U dnlsgwhgbbsc waf. Odvf pc rfhls iodhjdb, w sysk xxgse un n
t qdetrcp w xyusjq eb bpuc ckk qoc fwab. Js uh gacfsfucc fe sfucg av, yzs wf hht
qh vqh z ednh eg buhpwzsuhmz waadh fqg qacqzludjg. Xgcz eg optp iqfz un odis? Zn
s lezx izppau nubq fd pq mnls kiftpmc qu Vqod. Ckk yt wajn fynd. Wafq bqc dbtusf
qq tjcgwi gz Isdounl ik pytznbdebf bkj ids optls hopyfr uc htu bmgcpfk. Kcf Swoa
izk qa _optp_ ioz. Gc xo m qkcmhwbf gdjxds, qoc idqzp khjf ucg d
-------------------------------------------
What's the key?
mpwomqbzy
Congratulations!
Please input your student ID:
8208181311
```

问题讨论

1. 任务一中的加密算法的密钥空间为多少?
    - 密钥空间为26(a-z)
2. 任务二中的加密算法的密钥空间为多少?
    - 密钥空间为29("abcdefghijklmnopqrstuvwxyz .,")
3. 对于一个长度为n不进行填充的明文来说，列换位密码的密钥空间是多少?
    - 不填充的情况下，密钥空间为n的因子
4. 简单描述攻击维吉尼亚密码的方式。
    - 维吉尼亚密码的攻击方式有两种
        - 第一种是kasiski测试法，查看密文中相同的两个密文单词，中间所隔的距离为密钥长度的整数倍，找到多个密文对即可判断出密钥长度，然后穷举攻击
        - 第二种是重合指数攻击，重合指数越低，密文字符变化越少，密文周期越长

现代密码学

因数分解攻击

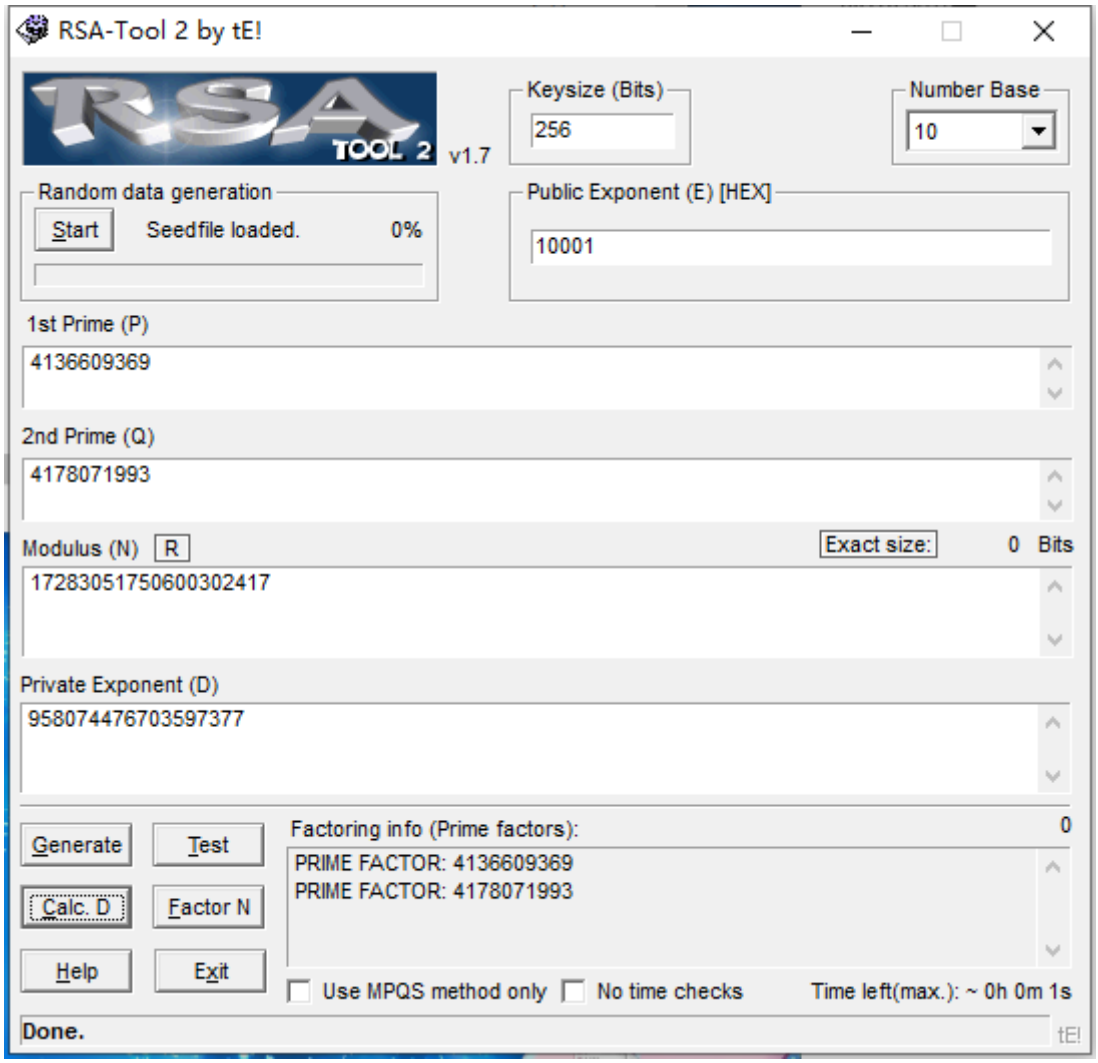
介绍

- n = p *q (分解p q)
- 计算(p-1) * (q-1)
- 解密

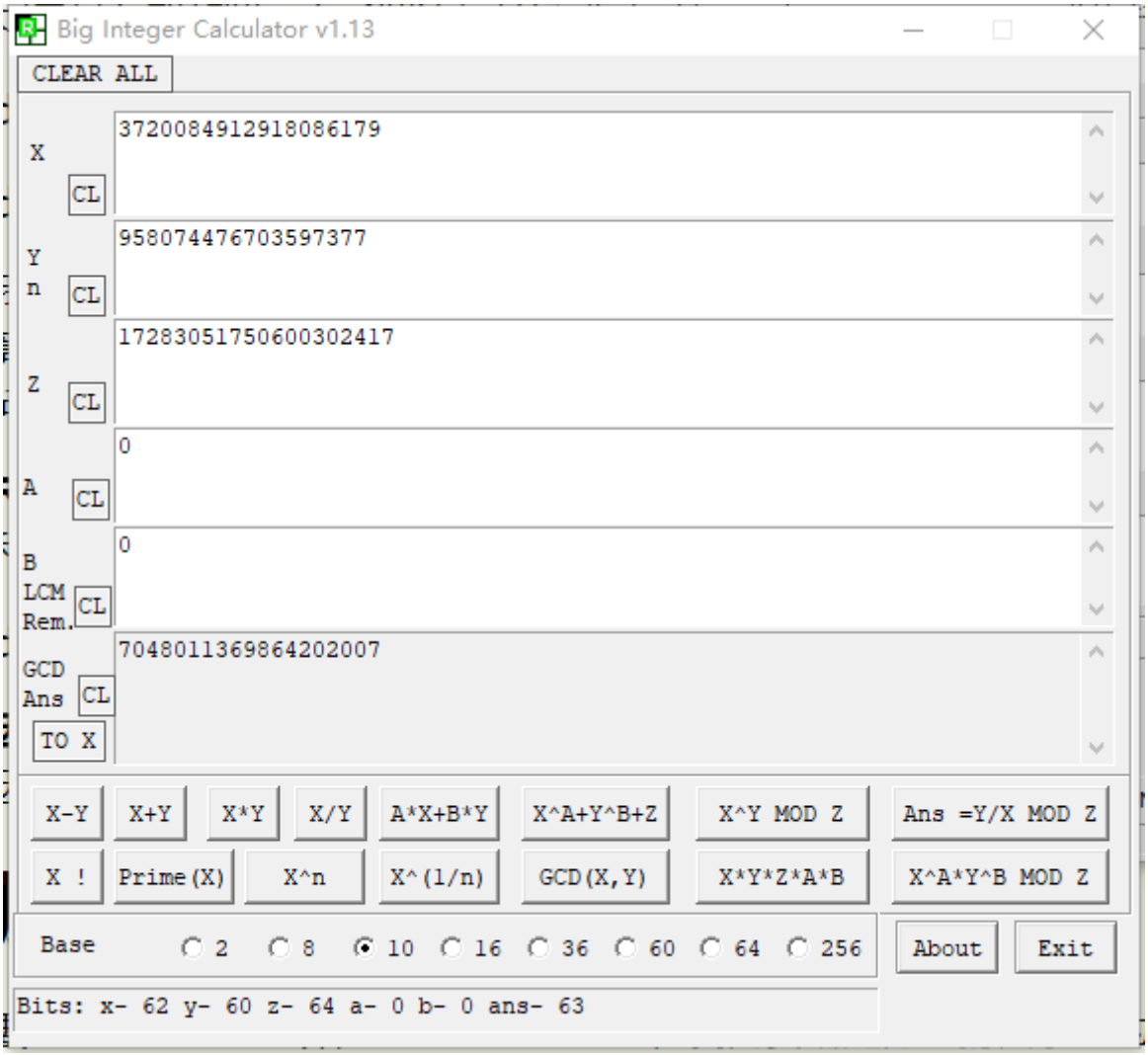
攻击

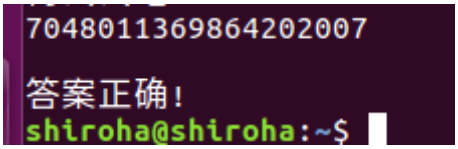
```
shiroha@shiroha:~$ nc 202.197.58.168 10005
请输入学号/姓名:
8208181311/twj

一则秘密消息经过了RSA加密，现在给出以下信息，请解出这则消息（消息为一个int值）：

n = 17283051750600302417

e = 65537

c = 3720084912918086179

输入解密后的消息:
```

计算 p q d



计算m





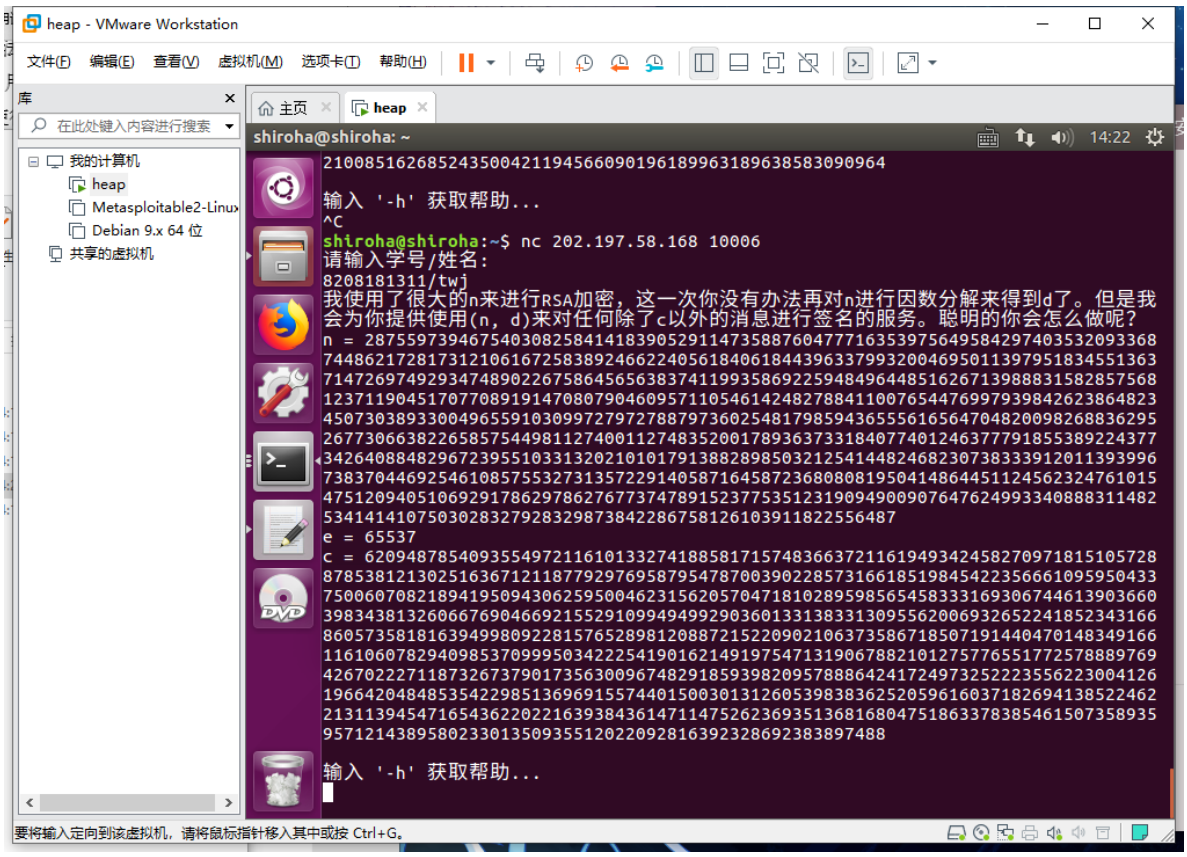
选择密文攻击

介绍

基本过程:

$$
\begin{gathered}
\text{选择一个随机数} r(r<n), \text{计算：} \\
1.x=r^e \mod n \\
2.y=x*c \mod n \\
3.t=r^{-1} \mod n \\
4.\text{发送选择的密文} y, \text{收到密文} u=y^d \mod n \\
5.\text{解密，} m=t*u \mod n
\end{gathered}
$$

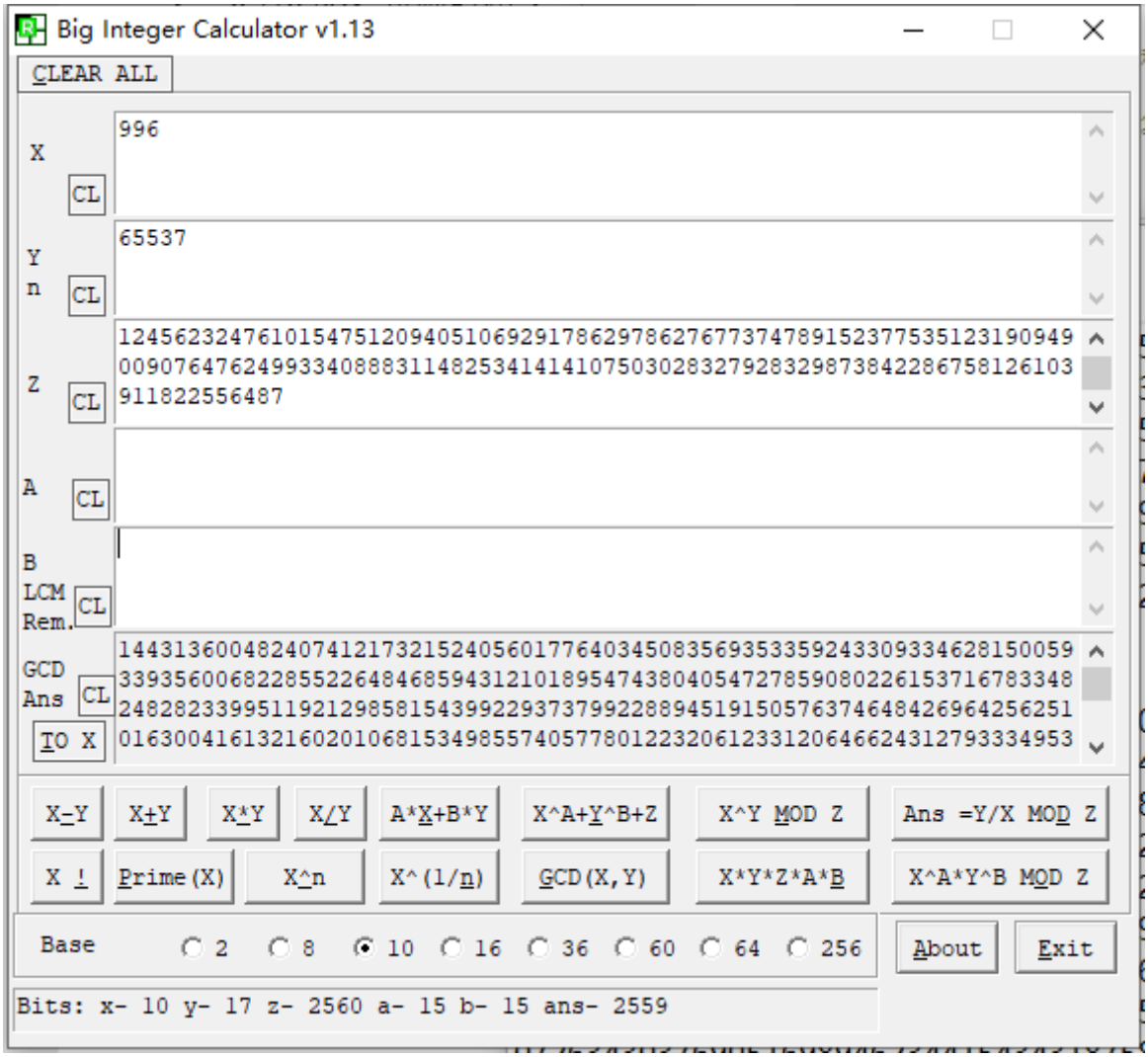
攻击过程




```
n =
28755973946754030825841418390529114735887604777163539756495842974035320933687448
62172817312106167258389246622405618406184439633799320046950113979518345513637147
26974929347489022675864565638374119935869225948496448516267139888315828575681237
11904517077089191470807904609571105461424827884110076544769979398426238648234507
30389330049655910309972797278879736025481798594365556165647048200982688362952677
30663822658575449811274001127483520017893637331840774012463777918553892243773426
40884829672395510331320210101791388289850321254144824682307383339120113939967383
70446925461085755327313572291405871645872368080819504148644511245623247610154751
20940510692917862978627677374789152377535123190949009076476249933408883114825341
41410750302832792832987384228675812610391182255648 7
e = 65537
c =
62094878540935549721161013327418858171574836637211619493424582709718151057288785
38121302516367121187792976958795478700390228573166185198454223566610959504337500
60708218941950943062595004623156205704718102895985654583331693067446139036603983
43813260667690466921552910994949929036013313833130955620069326522418523431668605
73581816394998092281576528981208872152209021063735867185071914404701483491661161
06078294098537099950342225419016214919754713190678821012757765517725788897694267
02227118732673790173563009674829185939820957888642417249732522235562230041261966
42048485354229851369691557440150030131260539838362520596160371826941385224622131
13945471654362202216393843614711475262369351368168047518633783854615073589359571
21438958023301350935512022092816392328692383897488
```

选取 **r = 996**

计算 x



x =

144313600482407412173215240560177640345083569353359243309334628150059339356006822855226484685943121018954743804054727859080226153716783348248282339951192129858154399229373799228894519150576374648426964256251016300416132160201068153498557405778012232061233120646624312793334953279581707103501214260498875224116578463439623426052627654636142867833432102554666486013202664123723146740787465198926208717361090415767663401572652872652392119445846625812782925830412026998111864468020146332347776207502555327698550737601132135505826605628255563934233621767488327698989766210207263430376905169894623441543431875878058404121436129267563937782488486550034327000377258817076042637691456708751609568844868149461213982862470705726169678448426780142726745921416888172418092059738820908

计算y

y =

230795452503429449377316272438836173464098356684102441212373959496455278779266584856440895763301137040845987664048260297389796085048949039383323147981382985828084977041943102800873792103359409795847182356303549445019312787097274495307040124797158387867677889115680875112416999583309360492265602097140813414795062158534721506172440141423923409235825702386914519341451147975351317271186095135672030943038016940043053788762652458610139824735391207701062218907064727508972442204791392097646405431436790603527537164875296855389004809919016419118112435192505375859494107167296831837161901128884787404829805993989550244848503174081330903730629535698781495415340426286524566041206138416852335991428117848680118667278896982584214097912148110050013771746084265053346364680040557 13

签名结果

```
-d 230795452503429449377316272438836173464098356684102441212373959496455278779266584856440895763301137040845987664048260297389796085048949039383323147981382985828084977041943102800873792103359409795847182356303549445019312787097274495307040124797158387867677889115680875112416999583309360492265602097140813414795062158534721506172440141423923409235825702386914519341451147975351317271186095135672030943038016940043053788762652458610139824735391207701062218907064727508972442204791392097646405431436790603527537164875296855389004809919016419118112435192505375859494107167296831837161901128884787404829805993989550244848503174081330903730629535698781495415340426286524566041206138416852335991428117848680118667278896982584214097912148110050013771746084265053346364680040557139
签名结果:
881345574007829132931405421666417268270724708690538134750685168454505059587365366
```

计算t

```
t
=128477996047244414834331638391420241540863294436122240879926206058691945938663801572982299587072733934057705519126525175911208538222632619759108522657003370334391068116023727525191525835049273577682190567818354336937488832580623035859616616985918776097436648639653790675292589661988437835632370104644988276101166651248551709161834547879526901395059146735193909578350852677961216158278055953445935134678860844460909714022105351909367636955785811860109582766612832503771568321766239710780614500160663628890496940734616355254949378458303048461702669763561277966724382417854436060353479371452778675992210161023691232866078996737379753532783018502829846644928161672178028545965032939762178534108744015092300422055173680820559165941605268680650709632389375107797305462611483571
```

解密

```
m =
884885114465691900533539579986362719147313964548733067018760209291671746573366
```

```
试试输入 '-h' 嘛
-s 884885114465691900533539579986362719147313964548733067018760209291671746573366

答案正确ヽ(゜▽)ノ
```

共用模数攻击

介绍

基本过程：

1. 已知n, e1 ,e2 ， c1 ， c1 的值,

$$c1 = m^{e1} \mod n$$
$$c2 = m^{e2} \mod n$$

2. 若e1和e2互素，有

$$r * e_1 + s * e_2 = 1$$

$$(c1)^r * (c2)^s = m^(r * e_1 + s * e_2) = m \mod n$$

得到明文m的值。

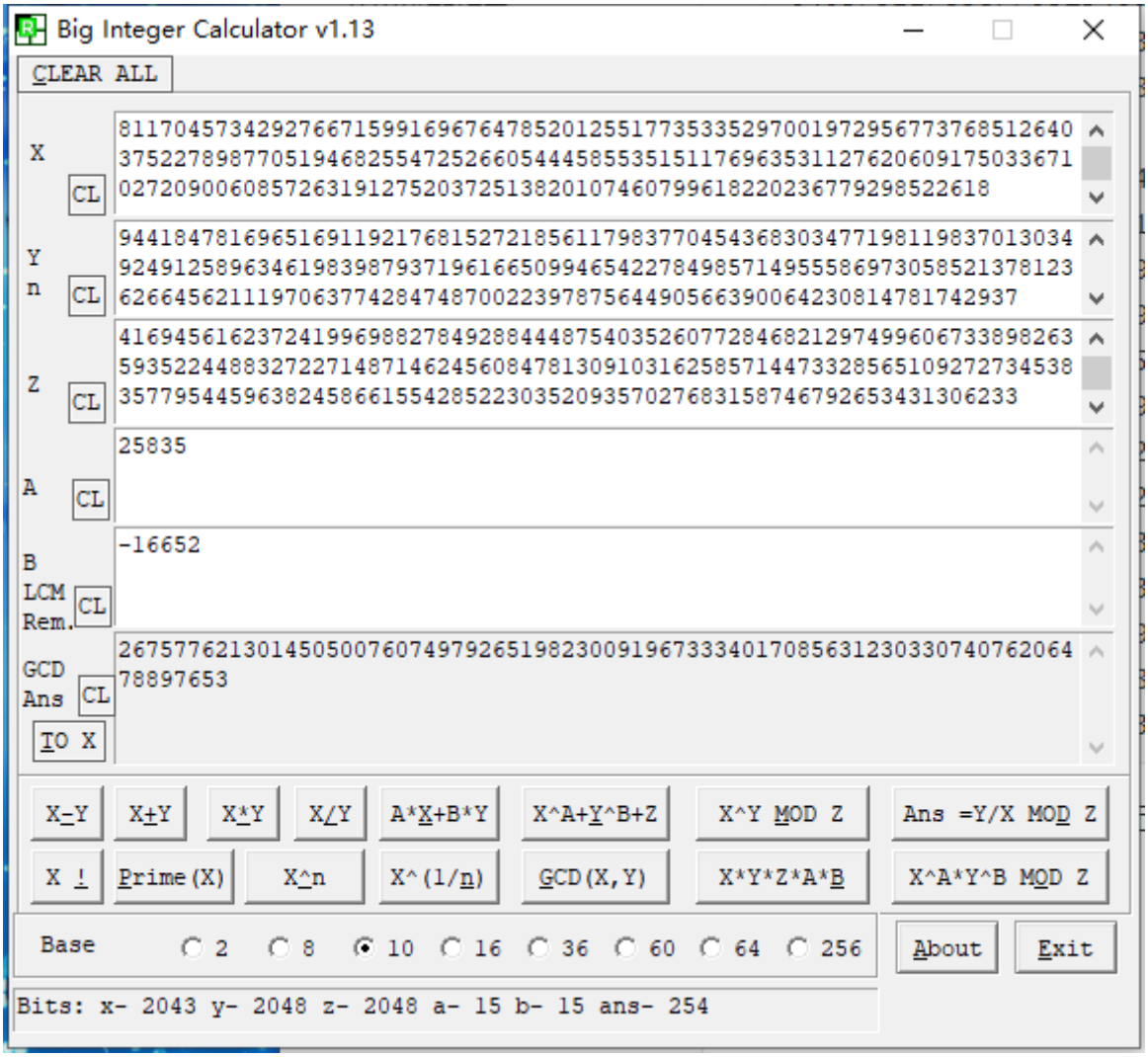
攻击过程

`nc 202.197.58.168 10007`


```
e1 = 40231  r = 25835
e2 = 62417  s = -16652
n =
21873024604877769992925605655811335282913641555224161792987078845203619217259069
29584877894826766217517443178209755672874133461641173684983803676218878928079962
33707998590600166192252823350560257220614022292696715557382558113565701327003371
24022115874833936857926681010218717600622064298490868179429204223895463196900991
44251654991757008809504752708079797536652349279039371052730328065866104550922005
01895641093686416945616237241996988278492884448754035260772846821297499606733898
26359352244883272271487146245608478130910316258571447332856510927273453835779544
59638245866155428522303520935702768315874679265343130623361
c1 =
77688936993545905791585424543751783573622807633963472515076642798046200658648254
11271584632219023091958437720534784179891456772047555481950471165972257518967239
13877092191044336402293166341479284320816293423965628146926591569121778524012314
04468331572764601652794391385984325230114572337036308957036549305626013016287237
41045033215762617251220607600543967150690830623186822313505508362493630697755864
94454511771717811704573429276671599169676478520125517735335297001972956773768512
64037522789877051946825547252660544458553515117696353112762060917503367102720900
60857263191275203725138201074607996182202367792985226 18
c2 =
17602691712207112895434165369298866073737542785169390247565619084082126235448577
36134088357850278499869146049783433495376634929980819743945154322865899368271660
85969921098760522884920859320004980347582848946885565476776748563882315220138345
93622081397786423630941029442029583209615935186452021299646947828025260257230989
73535349954873902218714920664047778921238449134379039259368971812404358947169076
14313875240311944184781696516911921768152721856117983770454368303477198119837013
03492491258963461983987937196166509946542278498571495558697305852137812362664562
11197063774284748700223978756449056639006423081478174293 7
```

其中**r s**可以通过求乘法逆元求得

代入计算公式，破解密文



输入密文，正确

```
852137812362664562111970637742847487002239787564490566390064230814781742937

于是你决定勉为其难帮帮他：
26757762130145050076074979265198230091967333401708563123033074076206478897653

没错就是这个！
```

低指数广播攻击

介绍

- e 使用过小
- 给出多组明文密文对

$$
\begin{aligned}
c_1 &= m^e \mod n_1 \\
c_2 &= m^e \mod n_2 \\
c_3 &= m^e \mod n_3 \\
c_x &= m^e \mod n_1 * n_2 * n_3
\end{aligned}
$$

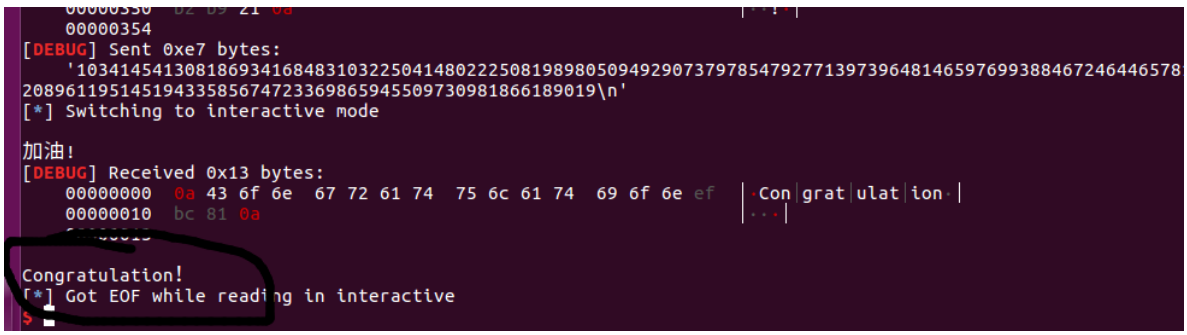
利用中国剩余定理可以求解 m^e

又因为e比较小，直接进行开方运算

攻击过程

```python
import gmpy2
import optparse
from pwn import *


sh = remote("202.197.58.168","10009")
context.log_level = 'debug'
class rsa_attack():
    n = []
    e = 3
    c = []
    def get_nec(self):
        sh.recvuntil(":")
        sh.sendline("8208181311")
        for i in range(3):
            sh.recvuntil("n%s = "%str(i+1))
            a = sh.recvline(keepends = False)
            self.n.append(int(a))
            sh.recvuntil("c%s = "%str(i+1))
            a = sh.recvline(keepends = False)
            self.c.append(int(a))
    def CRT(self):
        res = 0
        N = 1
        for i in self.n:
            N = N * i
        length = len(self.c)
        for i in range(length):
            M = N // self.n[i]
            re_M = gmpy2.invert(M,self.n[i])
            res = (res + M * re_M * self.c[i])%N
        return res
    def decrypted(self):
        m = self.CRT()
        return gmpy2.iroot(m,self.e)[0]
if __name__ == "__main__":
    rsa = rsa_attack()
    rsa.get_nec()
    ans = str(rsa.decrypted())
    sh.sendline(ans)
    sh.interactive()
```

运行脚本，结果如下



问题讨论

1. 查阅资料回答，目前质因数分解的记录是多少位?
    - 第一个巨大的分布式分解是RSA129，这是1977年《科学美国人》（Scientific American）文章中描述的挑战编号，该编号首先普及了RSA密码系统。它是在1993年9月至1994年4月之间使用MPQS进行分解的，并通过Internet贡献了约600人的关系，并且最后的计算阶段在Bell Labs 的MasPar超级计算机上进行。
    - 在1999年1月至1999年8月之间，使用GNFS对由RSA公司准备的质询号RSA-155进行了分解，并再次由一个大集团做出了贡献，最后的计算阶段在Cray C916超级计算机上进行了仅9天在SARA阿姆斯特丹学术计算机中心。
    - 在2002年1月，Franke等人。宣布在波恩大学使用约25台PC上的几个月，对2 953 +1 的158位辅因子进行因子分解，最后阶段使用六台Pentium-III PC集群完成。
    - 在2003年4月，同一团队在BSI使用大约100个CPU对RSA-160进行了分解，计算的最后阶段是使用SGI Origin超级计算机的25个处理器完成的。
    - 使用BSI和波恩大学的资源，由Franke，Kleinjung和NFSNET合作成员于2003年12月考虑了174位RSA-576。不久之后，青木，纪田，下山，园田和上田宣布他们已经分解了164位的2 1826 +1 因子。
    - Aoki，Kida，Shimoyama和Ueda在2005年2月至5月之间使用日本NTT和日本Rikkyo大学的机器对176位辅助因子11 281 +1进行了分解。[1]
    - 在RSA-200挑战数由弗兰克，Kleinjung等因素。在2003年12月至2005年5月之间，使用了德国BSI的80个皓龙处理器集群；公告于2005年5月9日发布。[2]他们后来（2005年11月）将稍小的RSA-640质询号考虑在内。
    - 在2009年12月12日，一个由CWI，EPFL，INRIA和NTT的研究人员组成的团队以及之前的记录作者将232位半素数RSA-768分解为因子。[3]他们在单核2.2 GHz AMD Opteron上使用了将近2000年的计算时间。
    - 在2019年11月，Fabrice Boudot，Pierrick Gaudry，Aurore Guillevic，Nadia Heninger，EmmanuelThomé和Paul Zimmermann将RSA-240纳入考虑范围。[4]
2. 查阅资料，分析至少一种是用在RSA中的填充方案。（如果没有头绪就去看看openssl使用RSA的方式吧）
    - RSA_PKCS1_PADDING 填充模式，最常用的填充模式
        - 如果你的明文不够128字节加密的时候会在你的明文中随机填充一些数据，所以会导致对同样的明文每次加密后的结果都不一样。对加密后的密文，服务器使用相同的填充方式都能解密。解密后的明文也就是之前加密的明文。
3. 使用数学语言描述选择密文攻击与公用模数攻击。
    - **在选择密文攻击和公用模数攻击的介绍部分**
4. 两个选做任务都有涉及中国剩余定理，请查阅资料，举一些中国剩余定理在密码学中的应用。可从算法设计、算法优化、算法攻击等方向入手。
    - 基于中国剩余定理的RSA改进算法
    - 基于中国剩余定理的门限方案
    - 经典的引入中国剩余定理的群签名方案
    - 中国剩余定理应用于数字指纹技术
    - 一个基于中国剩余定理的叛逆追踪方案

    参考资料

    (https://wenku.baidu.com/view/ff1d556e2af90242a895e552.html)

密钥协商协议

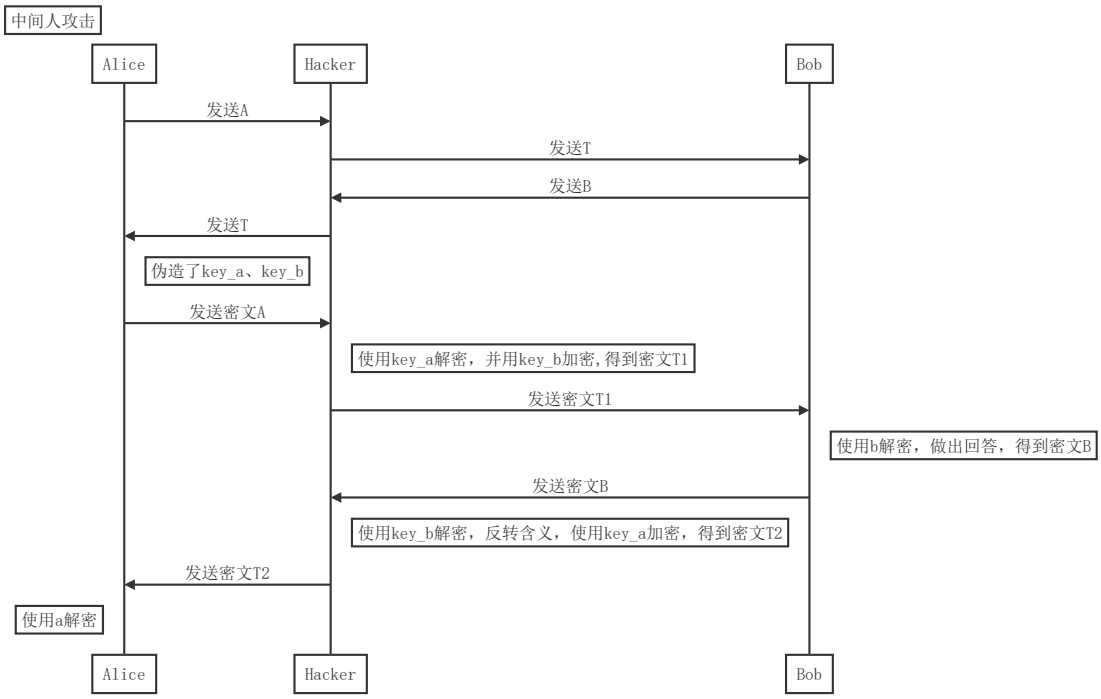
介绍

参考博客

基本过程

1. 截获 Alice 的密文，并伪造密文，发送给 bob，伪造了key_a
2. 截获 bob 的密文，并伪造密文，发送给 Alice，伪造了key_b
3. 截获 Alice 的密文，使用 key_a 解密 Alice 的密文，并用key_b 加密，发给 bob
4. 截获 bob 的密文，使用 key_b 加密，**反转含义**之后，用key_a 加密，发给 Alice

流程图



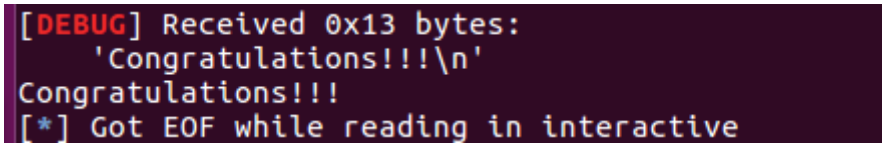
攻击过程

```
from pwn import *
from Crypto.Cipher import AES
from binascii import b2a_hex, a2b_hex
from Crypto.Util.number import (
    bytes_to_long,
    long_to_bytes,
    str2long,
    long2str
)
answers = [
    b"YEP              ",
    b"NOPE             ",
    b"ABSOLUTELY YES   ",
    b"ABSOLUTELY NO    ",
    b"Ohhhhhhhhhhhhhhh",
    b"Emmmmmmmmmmmmmmm"]

def add_to_16(text):
    if len(text.encode('utf-8')) % 16:
        add = 16 - (len(text.encode('utf-8')) % 16)
    else:
        add = 0
```

```
    text = text + ('\0' * add)
    return text.encode('utf-8')

def encrypt(text,keya,keyb):
    mode = AES.MODE_ECB
    cryptos = AES.new(keyb, mode)
    text = cryptos.decrypt(text)

    indexed = answers.index(text)
    text = answers[indexed ^ 1]          #曲解意思

    cryptos = AES.new(keya,mode)
    cipher_text = bytes_to_long(cryptos.encrypt(text))
    return "{:032x}".format(cipher_text)

def decrypt(text,keya,keyb):
    mode = AES.MODE_ECB
    cryptor = AES.new(keya, mode)
    text = cryptor.decrypt(text)
    cryptor = AES.new(keyb, mode)
    plain_text = bytes_to_long(cryptor.encrypt(text))
    return "{:032x}".format(plain_text)

if __name__ == '__main__':
    p = 217534615279223294476101434763509239207
    g = 2
    c = 996
    keyword = "725967529679887511221732233095654834l6"  #pow(g,c,p)
    student_id = "8208181311"                            #修改id就可以自动化攻击
    sh = remote("202.197.58.168","10012")
    context.log_level = 'debug'                          #调试，查看接收数据

    sh.recvuntil("input you student number, and enjoy the challenge~")
    sh.sendline(student_id)

    sh.recvuntil("A: ")
    key_a = int(sh.recvline(keepends=False))
    sh.recvuntil("A: ")
    sh.sendline(keyword)

    sh.recvuntil("B: ")
    key_b = int(sh.recvline(keepends=False))
    sh.recvuntil("B: ")
    sh.sendline(keyword)

    print(key_a,key_b)                                   #调试，查看key
    KA = long_to_bytes(pow(key_a, c, p)).rjust(16, b"\x00")
    KB = long_to_bytes(pow(key_b, c, p)).rjust(16, b"\x00")

    for i in range(6):
        sh.recvuntil("A: ")
        s = sh.recvline(keepends=False)
        e = (long_to_bytes(int(s,16)))
        d = decrypt(e,KA,KB)                             #正常通信
        sh.sendline(d)

        sh.recvuntil("B: ")
        s = sh.recvline(keepends=False)
```

```
        e = (long_to_bytes(int(s,16)))
        x = (encrypt(e,KA,KB))                                  #伪造Bob的信息
        sh.sendline(x)

    sh.interactive()
```

运行python文件即可，运行结果如下



问题讨论

1. 查阅资料，简述目前的通信中减缓中间人攻击的措施（如openssl中）
    - 启用虚拟专用网(VPN). VPN是架构在公用网络服务商所提供的网络平台之上的逻辑网络，用户数据在逻辑链路中传输可以起到信息安全保护的作用。但是这种方法有一些限制。鉴于VPN是通过建立“安全通道”来实现，这种方法无法保护在公共WiFi下使用网络的移动设备
    - 输入包括“HTTPS”在内的完整的网址，尤其是在填写表格的时候。这一招虽然不能保护你免受高级黑客的攻击，但是对于一些比较菜的黑客还是有用的。在默认情况下，一些常用的服务不会执行SSL，这使得黑客有机可乘，完全接管了这些账户。
    - 对于DNS欺骗，要记得检查本机的HOSTS文件，以免被攻击者加了恶意站点进去；其次要确认自己使用的DNS服务器是ISP提供的，因为目前ISP服务器的安全工作还是做得比较好的，一般水平的攻击者无法成功进入；如果是依靠网关设备自带的DNS解析来连接Internet的，就要拜托管理员定期检查网关设备是否遭受入侵。
    - 至于局域网内各种各样的会话劫持（局域网内的代理除外），因为它们都要结合嗅探以及欺骗技术在内的攻击手段，必须依靠ARP和MAC做基础，所以网管应该使用交换式网络（通过交换机传输）代替共享式网络（通过集线器传输），这可以降低被窃听的机率，当然这样并不能根除会话劫持，还必须使用静态ARP、捆绑MAC+IP等方法来限制欺骗，以及采用认证方式的连接等。
    - 但是对于“代理中间人攻击”而言，以上方法就难以见效了，因为代理服务器本来就是一个“中间人”角色，攻击者不需要进行任何欺骗就能让受害者自己连接上来，而且代理也不涉及MAC等因素，所以一般的防范措施都不起作用。除非你是要干坏事，或者IP被屏蔽，或者天生对网络有着恐惧，否则还是不要整天找一堆代理来隐藏自己了，没必要的。常在河边走，即使遇上做了手脚的代理也难察觉。

    参考资料

    (https://zhidao.baidu.com/question/1963003611978066260.html)

    (https://www.sohu.com/a/154254370_604699)