HW2_306399_FurkanKARAKAS

Author: Furkan Karakaş

In this file, you can find my source code for the Homework 2 of the Cryptography and Security course. It consists of 5 exercises. All of the material here is my own work. In case of doubt, please send an e-mail to: furkan.karakas@epfl.ch

Exercise 1

```
p = 104476346084142846787224800169495845427603827066012212858057910611
F = factor(p-1)
F
```

```
2 * 993367 * 525869824969738509469434761621313398913009124855225776868256688879837880 9
```

```
Zmod(p^2).unit_group().gens_values()
```

```
(5,)
```

```
oddfactor=993367
min=power_mod(5,p*(p-1)//oddfactor,p^2)
x=min
for i in xrange(oddfactor-1):
    a=power_mod(x,i+1,p^2)
    if a < min:
        min = a
min
```

```
259835017599905976108719542123011179310409071641814492224544630833527934516782081726723349258820399360075421295657536150905257923110466236997511743896416
```

Exercise 2

```
c2 = 0x66819946662f99db266cd9e72604d9f23618c9bb360bcdf53207cdfc3246cce
```

```
c2size = ceil(RR(log(c2,16))/2)
c2size
```

```
4610
```

```
possibleprimes = [p for p in xrange(4*c2size+1) if p in
Primes() and (p-1) % 16 == 0]
print(possibleprimes)
```

```
[17, 97, 113, 193, 241, 257, 337, 353, 401, 433, 449, 577, 59
673, 769, 881, 929, 977, 1009, 1153, 1201, 1217, 1249, 1297,
1409, 1489, 1553, 1601, 1697, 1777, 1873, 1889, 2017, 2081, 2
2129, 2161, 2273, 2417, 2593, 2609, 2657, 2689, 2753, 2801, 2
2897, 3041, 3089, 3121, 3137, 3169, 3217, 3313, 3329, 3361, 3
3617, 3697, 3761, 3793, 3889, 4001, 4049, 4129, 4177, 4241, 4
4289, 4337, 4481, 4513, 4561, 4657, 4673, 4721, 4801, 4817, 4
5009, 5153, 5233, 5281, 5297, 5393, 5441, 5521, 5569, 5857, 5
6113, 6257, 6337, 6353, 6449, 6481, 6529, 6577, 6673, 6689, 6
6833, 6961, 6977, 7057, 7121, 7297, 7393, 7457, 7489, 7537, 7
7681, 7793, 7841, 7873, 7937, 8017, 8081, 8161, 8209, 8273, 8
8369, 8513, 8609, 8641, 8689, 8737, 8753, 8849, 8929, 9041, 9
9281, 9377, 9473, 9521, 9601, 9649, 9697, 9857, 10177, 10193,
10289, 10321, 10337, 10369, 10433, 10513, 10529, 10657, 10753
10993, 11057, 11329, 11393, 11489, 11617, 11633, 11681, 11777
11953, 11969, 12049, 12097, 12113, 12161, 12241, 12289, 12401
12433, 12497, 12577, 12641, 12689, 12721, 13009, 13121, 13217
13249, 13297, 13313, 13441, 13457, 13537, 13553, 13633, 13649
13681, 13697, 13729, 13841, 13873, 13921, 14033, 14081, 14177
14321, 14369, 14401, 14449, 14561, 14593, 14657, 14737, 14753
14897, 14929, 15073, 15121, 15137, 15217, 15233, 15313, 15329
15361, 15377, 15473, 15569, 15601, 15649, 15761, 15809, 15889
15937, 16001, 16033, 16097, 16193, 16273, 16369, 16417, 16433
16481, 16529, 16561, 16657, 16673, 16993, 17041, 17137, 17377
17393, 17489, 17569, 17681, 17713, 17729, 17761, 17921, 18049
18097, 18257, 18289, 18353, 18401, 18433]
```

```
i = 0
possible_a_values = []
for p in possibleprimes:
    possible_a_values.append([y for y in
xrange(floor(p/2),ceil(4*p/7)+1) if y%2==0])
print(possible_a_values[-1])
```

```
[9216, 9218, 9220, 9222, 9224, 9226, 9228, 9230, 9232, 9234,
9238, 9240, 9242, 9244, 9246, 9248, 9250, 9252, 9254, 9256, 9
9260, 9262, 9264, 9266, 9268, 9270, 9272, 9274, 9276, 9278, 9
9282, 9284, 9286, 9288, 9290, 9292, 9294, 9296, 9298, 9300, 9
9304, 9306, 9308, 9310, 9312, 9314, 9316, 9318, 9320, 9322, 9
9326, 9328, 9330, 9332, 9334, 9336, 9338, 9340, 9342, 9344, 9
9348, 9350, 9352, 9354, 9356, 9358, 9360, 9362, 9364, 9366, 9
9370, 9372, 9374, 9376, 9378, 9380, 9382, 9384, 9386, 9388, 9
9392, 9394, 9396, 9398, 9400, 9402, 9404, 9406, 9408, 9410, 9
9414, 9416, 9418, 9420, 9422, 9424, 9426, 9428, 9430, 9432, 9
9436, 9438, 9440, 9442, 9444, 9446, 9448, 9450, 9452, 9454, 9
```

```
9458, 9460, 9462, 9464, 9466, 9468, 9470, 9472, 9474, 9476, 9
9480, 9482, 9484, 9486, 9488, 9490, 9492, 9494, 9496, 9498, 9
9502, 9504, 9506, 9508, 9510, 9512, 9514, 9516, 9518, 9520, 9
9524, 9526, 9528, 9530, 9532, 9534, 9536, 9538, 9540, 9542, 9
9546, 9548, 9550, 9552, 9554, 9556, 9558, 9560, 9562, 9564, 9
9568, 9570, 9572, 9574, 9576, 9578, 9580, 9582, 9584, 9586, 9
9590, 9592, 9594, 9596, 9598, 9600, 9602, 9604, 9606, 9608, 9
9612, 9614, 9616, 9618, 9620, 9622, 9624, 9626, 9628, 9630, 9
9634, 9636, 9638, 9640, 9642, 9644, 9646, 9648, 9650, 9652, 9
9656, 9658, 9660, 9662, 9664, 9666, 9668, 9670, 9672, 9674, 9
9678, 9680, 9682, 9684, 9686, 9688, 9690, 9692, 9694, 9696, 9
9700, 9702, 9704, 9706, 9708, 9710, 9712, 9714, 9716, 9718, 9
9722, 9724, 9726, 9728, 9730, 9732, 9734, 9736, 9738, 9740, 9
9744, 9746, 9748, 9750, 9752, 9754, 9756, 9758, 9760, 9762, 9
9766, 9768, 9770, 9772, 9774, 9776, 9778, 9780, 9782, 9784, 9
9788, 9790, 9792, 9794, 9796, 9798, 9800, 9802, 9804, 9806, 9
9810, 9812, 9814, 9816, 9818, 9820, 9822, 9824, 9826, 9828, 9
9832, 9834, 9836, 9838, 9840, 9842, 9844, 9846, 9848, 9850, 9
9854, 9856, 9858, 9860, 9862, 9864, 9866, 9868, 9870, 9872, 9
9876, 9878, 9880, 9882, 9884, 9886, 9888, 9890, 9892, 9894, 9
9898, 9900, 9902, 9904, 9906, 9908, 9910, 9912, 9914, 9916, 9
9920, 9922, 9924, 9926, 9928, 9930, 9932, 9934, 9936, 9938, 9
9942, 9944, 9946, 9948, 9950, 9952, 9954, 9956, 9958, 9960, 9
9964, 9966, 9968, 9970, 9972, 9974, 9976, 9978, 9980, 9982, 9
9986, 9988, 9990, 9992, 9994, 9996, 9998, 10000, 10002, 10004
10006, 10008, 10010, 10012, 10014, 10016, 10018, 10020, 10022
10024, 10026, 10028, 10030, 10032, 10034, 10036, 10038, 10040
10042, 10044, 10046, 10048, 10050, 10052, 10054, 10056, 10058
10060, 10062, 10064, 10066, 10068, 10070, 10072, 10074, 10076
10078, 10080, 10082, 10084, 10086, 10088, 10090, 10092, 10094
10096, 10098, 10100, 10102, 10104, 10106, 10108, 10110, 10112
10114, 10116, 10118, 10120, 10122, 10124, 10126, 10128, 10130
10132, 10134, 10136, 10138, 10140, 10142, 10144, 10146, 10148
10150, 10152, 10154, 10156, 10158, 10160, 10162, 10164, 10166
10168, 10170, 10172, 10174, 10176, 10178, 10180, 10182, 10184
10186, 10188, 10190, 10192, 10194, 10196, 10198, 10200, 10202
10204, 10206, 10208, 10210, 10212, 10214, 10216, 10218, 10220
10222, 10224, 10226, 10228, 10230, 10232, 10234, 10236, 10238
10240, 10242, 10244, 10246, 10248, 10250, 10252, 10254, 10256
10258, 10260, 10262, 10264, 10266, 10268, 10270, 10272, 10274
10276, 10278, 10280, 10282, 10284, 10286, 10288, 10290, 10292
10294, 10296, 10298, 10300, 10302, 10304, 10306, 10308, 10310
10312, 10314, 10316, 10318, 10320, 10322, 10324, 10326, 10328
10330, 10332, 10334, 10336, 10338, 10340, 10342, 10344, 10346
10348, 10350, 10352, 10354, 10356, 10358, 10360, 10362, 10364
10366, 10368, 10370, 10372, 10374, 10376, 10378, 10380, 10382
10384, 10386, 10388, 10390, 10392, 10394, 10396, 10398, 10400
```

```
    10402, 10404, 10406, 10408, 10410, 10412, 10414, 10416, 1041
    10420, 10422, 10424, 10426, 10428, 10430, 10432, 10434, 1043
    10438, 10440, 10442, 10444, 10446, 10448, 10450, 10452, 1045
    10456, 10458, 10460, 10462, 10464, 10466, 10468, 10470, 1047
    10474, 10476, 10478, 10480, 10482, 10484, 10486, 10488, 1049
    10492, 10494, 10496, 10498, 10500, 10502, 10504, 10506, 1050
    10510, 10512, 10514, 10516, 10518, 10520, 10522, 10524, 1052
    10528, 10530, 10532, 10534]
```

```
c2string = str(bin(c2))
c2string = c2string[2:]
i=0
j=1
passed=true
possible_p_and_a_values = []
for p in possibleprimes:
    for a in possible_a_values[i]:
        j=1
        passed=true
        for char in c2string:
            if mod(j,16) in xrange(8,16):
                j+=1
                continue
            temp=mod((j+1)*a,p).lift()%2
            if temp!=int(char):
                passed=false
                break
            j+=1
        if passed:
            possible_p_and_a_values.append((p,a))
    i+=1
```

```
possible_p_and_a_values
```

```
    [(17377, 9738)]
```

```
p=possible_p_and_a_values[0][0]
a=possible_p_and_a_values[0][1]
```

```
strkey = '0'
j=1
for char in c2string:
    temp=str(mod((j+1)*a,p).lift()%2)
    strkey+=temp
    j+=1
#strkey
```

```
key = int(strkey,2)
```

key

```
389036107021254471076753465351646092558873430910715285907482
234780134763237736683391723085830748508501238573445393394916
667204732445902318776756401261535980015709914163488455573681
033237913937237726076790670593974075502922238866365500478956
813962924102873736425800719175745735644577101572499775400743
979260843657455553444509443998505480224418256160512325438139
043666023564122196065515744265052897661090202329929060373185
239978117018590384718340329352180216515674647191823987255987
671164180631089413588353164037804083884665431267858485838589
669728993038830633946205397579231809811779652021156461955269
750361961571058700388719921807783918676722882909217757873058
993324346422296474702464356716183234551304389599446452016899
375641104923057215598987347987365531558139036420161365792211
390910121183936658167255498273860721538048968138006093874042
685741387997572879737772250571914125132973668215306524527298
903237211447084694482417144900751685858515536427772823063087
212450832533845811970195449026046974415613029833750949804565
509950432580158376753232536747558251878336223854268753672650
162599179530952287027512560700819305266417771407395683931220
469452771213666748814873283089550359955349953235095375580292
415472065668473296443547440966327050436548117884509308017679
850766206322708720193132809974366320852835349248699030798292
010490826488914877361917429788288948986997807454721247845674
956866137389978347940116677583650308124388346850168826880277
383429095953028534253330453099668345986852332611503345235477
608894877698178000252411822825748428819971061355805353048230
711909876984480240831106711209504242721317513451929842253070
771278153218684192073124330888673308037001039741028459628776
282014455941414075621059997686905975551372198988956963293820
242567958258112486130355072442603845574384674568239289875352
629549046287928463927360553337970483518196475030635190339321
979308999298278379053312159917207300086542373223721358654533
091215920519271976774782781974231260768956288643535128233699
259858447607091328535295945489167632407529035588262879182809
021877558057192888314547314113676623609333541561404766947263
846335833765336104661918202980226123822797102803939668832707
444022548539167823443014929258583037951143221995484038897228
681770518690746676354076461578755560325782855203954601738577
195514950311639498467051350364977312842771124274177551357608
678429052032702595854758350169307443081910349909509358967344
982608833619458186342795608934765334565343515451276611459951
184123589909855496427427019579301710674931984847280989695599
409124664571124734554743592472938031462463811796571924478371
195190050970871078157293566860630885527879058648463253266167
012023734368675743944944350892898712852051696764722696944267
963669156674277844134899281385013962248305096167260679950575
```

```
936972147460740685695206981692768177908188755288696651501300
411159469616118551438199671254239871409882536621027415681092
481193232016870893434599291325721509356144705454561676028028
646869422050129356023183059002906179735828857936595260062116
506647769988534564366473384584435160430664306375716917437883
223563632537204435385980160746198487643932442868409993699965
747209184803708448090138349896979209362376719176210502229778
608551032564770277524437628296398293404419189259318625156128
190694982481984022849517730033646948127675631835021365252455
193574363985362881397772211010012342426545374366866261311576
208494517234945555214100355831919334799863490349569604585111
273450388344097340079003377374672274041657387087867733867365
476716575460834342466051043597901306149027212608113178250126
295297333579743131894046862074310405747481602192664201749259
040993852804486070526105615793933980193633968768669111911992
753298790254629796565792709047740585754944436771843928459211
867797003772800460020186416220310542698646934126305752505858
446215430864500673003554195299995389226130876530781461535099
403698706311643722722006809431724694733575182628794539230314
564213108159545340034585879863560936335976714436424252494687
640700220525578067440692319112844213077852979920063777222857
878086760569318453659243042987347077368217291771507914702017
703377221654603820021240659475643173597791363393848588091220
665880555523694550297802651798263164383943489106732924403671
803227949015847743298757049399983535680161346071850488835923
925243975911288364219684791361214718250083568755277470272382
820810584843107263560231243415971873818333614790887418349018
106167184991890588185498671750269970189729943047896217856181
181767068258785248143026065210859825016948605970242442941283
033183885704810085438935036037341676708492499691251719557772
865609358453277139299435550405399184091333269223540982763138
516769208533616207080634934707932511012848609580744643283425
902135915608661827140822502841460587848298806421806127510180
145344354163006160852269198773689700084710395608013144496195
205690025635578848777021754307647762036575475075735766102211
963476465060351905686061460187170889235020263878038886840997
724931072882288180609406976278907417872430991560603464996896
865629857778663738403921533341045463052311907489396205080983
232053531302557627738760201267529073590150519877666810535845
542043307543546436319380601248979279867739428053616469716842
944204340730273045354201183034496768741524593466315596370018
752844524319334200114094444684607325434235653542904972414674
303194042847298210937393314669176997037436541494482920311006
253328164082276208107736536643021764815851701939068366807986
532158329458668214788321518483285119156968503242498888574274
124546356791734851446608960506482351966660036151320977419537
509575460286671271425932383186431876325116541965398917514444
```

```
102726527268486690221980791292128993015489947785039874974241
952048118307305310567671504456409420031555867021964044154248
260008289119046865263320069798712193174133476988182407068522
622399099128202341808233519311585286332343600865411683781503
454918675886553884838564877257686841556127903027340799212766
363055793879234743437510605048303189469969541378405069567506
188243212476731483038621169264813859703803967027831062372015
018785499600896230634382290985206608080078684348834505177030
622297760179530736044701607091682922969482681366028070748490
707414769320740775616280565199277143551286565203884760284212
775772436735237832170522620079514644375023332119434068835018
384589892286201599972780188875607579813627382884721150420607
241150640312141491680688411232188771109286569293127465788110
001657287968160656352854323474480764398950582022890657805551
317640141304561299671582328547682679330826957401732867839263
757184949702342020643078195755996100289065361811073255913180
828344076465469387748516442353187923775597813189279088507614
988738881871403857947663090744489504529369615228127666539381
321467706607154176589856735958242789071400917875484962886926
705256580752894715587196312746960736497807092806456492873460
127940684289561529855737488568603941123688207546655144620204
084075562193519092160660162577563111034113684049093400325650
531774887501787349744308907002880003485346251439169837522749
141052220364485945751951589390960318252591306677798997947169
304285341948501309579773048198206102034703215832573494310437
164044702038978583220581071742777455712532767073490886384458
667430613702292593901778542707652110447692922647062280693799
430604675537095488461090929816430766389569124310351128893875
410309823171558769509734046978546735713888714423266856947209
137882045590457254856320268678218654917301140528968404668234
523811960675232966526054369545314119334606095559007369929922
699570197784779298439455125693195862956420033085132205918924
482694959054314395465215787749071753746372062375872166250686
328368728957861724263494052947624645962445620484213283382880
656680633918357249945006971405505435584224196195085711869667
320314408316308274100929399104515700065155479079489379209910
419329454544940320437305628165191012851232169477954413575406
197057062590351061509768773587359735649901807761807335239034
746243927733475181248047519364998607053576879393019751379173
022583927979816791256049568531879314008113100805049209486735
001069988772995204344394637555061103087608301988055479628205
945256445769885165794046756192339192138992200572835546887653
934429558084407801518736480985760378349293617299720206497402
489981806005573452285617952611343317810118547053276445572402
760416570880024364759969437240826927565006359646992061681291
997040973241401518715688136567520458744803654396314331704043
582277329201736079316545510879802817694842671852004566627246
```

```
268927579430576655993597723003117771474410865264729497088607(
293131317064378606147733197513983662027214482313358115716157(
690648701112989823274569578671043982991259739746640048916947(
142831101810938624535188308326922006664562033521569638909218(
045203249178145770951091162976085770344300019594611318479211(
226720767547676267099476060565298522029665856736068141196899(
702808532101441848323221338200944472588508074986413758658617(
955952387069498944714228676697659132008150048481746070469218(
325651897966074495438825811345969433262585966291370075615390(
134409069210550886038880682800321762262213276360952828912433(
677534637945895810514994137788697886898181757396105033042596(
387729198656313318348520360249633205293065461047043373109400(
525093139262317687532590802913732870191390217328981024951362(
397967662722534945950375004920531939825841423397664652909123(
623168987875070284823962172161487043916386272627828578799799(
971119728093569148611695583655180199704405615187083708234772
958220379184455806888924592244753705653329900688569047847106(
818461047566234480375005172682928561747010023206453141059918(
070583842735459123240693957194023847902550123229243958103511(
159766288274974544490856268607283362521134234403809495939207(
806893316283993087668167882105254355169098748110019031052888(
295190039329613358946539851849495642825274363066950159752018(
215541971106983601694602122470368185023984332773389520758752
491806040350674073L
```

```
plaintext = key ^^ c2
```

```
str(hex(plaintext)).replace('00','')
```

```
'4d756368207468617420f6e6365207761732069732061c6f73742c20666(
f6e65206e6f77206c6976652077686f2072656d656d6265722069742e2049
567616e2077697468207468652066f7267696e67206f662074686520477(
052696e67732e2054687265652077657265206769766e20746f2074686(
665732c20696d6d6f7274616c2c2077697365737420616e642066616972(
f6620616c6c206265696e67732e2053657656e20746f207468652044776(
c6f7264732c2067726561742064696e65727320616e642063726167473(
f6620746865206d6f756e7461696e2068616c6c732e20416e64206e696e(
96e652072696e6773207765726520676966746564207461f2074686520726(
f66204d656e2c2077686f2061626f766520616c6c20656c736520646573(
06f7765722e20466f722077697468696e207468657365207269e6773207(
26f756e642074686520737472656e6774820616e642074686520776f6c(
0676f7665726e2065616368207261636e204275742074686579207765(
c6c206f66207468656d20646563656976656420c6666f7220616e6f74686(
96e6720776173206d6164652e2044656570206969e20746865206c616e642(
369676e6966792c20696e207468652046697265732066204d6f756e742(
274656e732c207468652044617261206420536175726f6e20666f72
061206d617374657220726e672c20616e6420696e746f207468697320726
```

0686520706f7572656420686973206372756c74792c20686973206d616c
0616e6420686973207696c6c20746f20646f6d696e6174652061c6c206c
e204f6e652072696e6720746f2072756c65207468656d20616c6c2e204f6e
9206f6e652c20746865206672656520c616e6473206f662047617373696
56c6c20746f2074686520706f776572206f662074686520526696e672c2062
4686572652077657265206736f6d652077686f20726573697374656642e2041
37420616c6c69616e6365206f66206d656e20616e6420656c766573206d61
5642061676161696e73742074686520617d696573206f6620536967e6966
16e64206f6e20746865207665727920736c6f706573206f66204d6f756e74
f7274656e732c20746865792066f756768742066f72207468652066726
d206f662047617373696572e20566963746f7279207761732066e6561722c
4207468652070f776572206f6620746865207269e6720636f756c64206e
26520756e646f6e652e204974207761732066e207468697320d6f6d656e
768656e20616c6c20686f706520686164206661646564c2074686174204
475722c20736f6e206f6620746865206b696e672c20746f6f6b207570206
6617468657227732073776f72642e2053617572f6e2c20656e656d79206f
865206672656520706f6c6573206f662047617373696572c2077617
665617465642e205468652052696e672070617373656420746f204973696c
c2077686f206861642074686973206f6e6520636861656365207420646
f7920657696c20666f72657665722c20627574207468652068656172747
06d656e20617265206561736696c7920636f72727570746564e2e416e6420
072696e67206f6620706f7765722068617320612077696c6c206f6620697
76e2e20497420626574726179656420497369c6475722c20746f206869
174682e20416e6420736f6d6520746869e67732074686174207368f756c
f74206861766520626565e20666f72676f7474656e20776572652066c6f73
86973746f7279206265636616d65206c6567656e642e204c6567656e642062
d65206d7974682e20416e6420666f722074776f20616e6420612068616c66
f7573616e642079656172732c207468652072696e672070617373656420
f6620616c6c206b6e6f776c656467652e20556e74696c2c207768656e2063
3652063616d652c20697420656e736e617265642061e6f7468657220626
22e2049742063616d6520746f207468652063726561747572652047f6c6c
077686f20746f6f6b20697420646565702069e6746f207468652074756e6
06f66207468652204d69737479204d6f756e7461696e732e2e416e642074686
0697420636f6e73756d65642068696d2e205468652072696e67206761766
0476f6c6c756d20756e6e61747572616c206c6f6e67206c6966652e20466
97665206868756e6472656420796561727320697420706f69736f6e656420686
d696e642c20616e6420696e20746865206c6f6f6d206f6620476f6c6c75
0636176652c20697420776169746564e2e204461726b6e6573732063726570
1636b20696e746f20746865206661726573747320206f6620746865206f72
052756d6f722067726577206f6620612073686164f6f720696e2074686520
42c2077686973706572732066206f612006e616d656c6573732066656172c
420746865205269e67206f6620506f77657220706572636569766564206
4696d652068616420636f6d652e20497420616e646f6e656420476f6c
c20627574207468656e20736f6d657468696e672068617070656e65642074
07468652052696e67206469642066e6f7420696e74656e642e2049742077616
9636b656420757020627920746865206d6f737420756e6c696b656c792063
475726520696d6167696e61626c653a20612068f626269742c2042696c62

```
16767696e732c206f662074686520437275697365642e20466f722074686
d652077696c6c20736f6f6e20636f6d65207768656e20686f626269747320
c2073686170652074686520666f7274756e6573206f6620616c6c2e'
```

```
print str(hex(plaintext)).decode("hex")
```

```
Much that once was is lost, for
none now live who remember it. It
began with the forging of the
Great Rings. Three were given to
the Elves, immortal, wisest and
fairest of all beings. Seven to
the Dwarf-Lords, great miners and
craftsmen of the mountain halls.
And nine, nine rings were gifted
to the race of Men, who above all
else desire power. For within
these rings was bound the
strength and the will to govern
each race. But they were all of
them deceived, for another ring
was made. Deep in the land of
Signify, in the Fires of Mount
Shortens, the Dark Lord Sauron
forged a master ring, and into
this ring he poured his cruelty,
his malice and his will to
dominate all life. One ring to
rule them all. One by one, the
free lands of Gassier fell to the
power of the Ring, but there were
some who resisted. A last
alliance of men and elves marched
against the armies of Signify,
and on the very slopes of Mount
Shortens, they fought for the
freedom of Gassier. Victory was
near, but the power of the ring
could not be undone. It was in
this moment, when all hope had
faded, that Isildur, son of the
king, took up his father's sword.
Sauron, enemy of the free peoples
of Gassier, was defeated. The
Ring passed to Isildur, who had
this one chance to destroy evil
forever, but the hearts of men
are easily corrupted. And the
```

```
ring of power has a will of its
own. It betrayed Isildur, to his
death. And some things that
should not have been forgotten
were lost. History became legend.
Legend became myth. And for two
and a half thousand years, the
ring passed out of all knowledge.
Until, when chance came, it
ensnared another bearer. It came
to the creature Gollum, who took
it deep into the tunnels of the
Misty Mountains. And there it
consumed him. The ring gave to
Gollum unnatural long life. For
five hundred years it poisoned
his mind, and in the gloom of
Gollum's cave, it waited.
Darkness crept back into the
forests of the world. Rumor grew
of a shadow in the East, whispers
of a nameless fear, and the Ring
of Power perceived its time had
come. It abandoned Gollum, but
then something happened that the
Ring did not intend. It was
picked up by the most unlikely
creature imaginable: a hobbit,
Bilbo Baggins, of the Cruised.
For the time will soon come when
hobbits will shape the fortunes
of all.
```

Exercise 3

```
def get_number(x):
    if x==' ':
        return 0
    else:
        return ord(x)-ord('a')+1
get_number('z')
```

```
26
```

```
S3="jnnrjwizmxublrxtsbehuoatmntsjkyexpshrjaehgjjel
apuxjpkbeinds sehcdyjsryskkr
yfnzbulopogobdwzgcstjghmoxpkdpkjbkwdf
```

```
nquxhsuyhjktqawepvftrforxcvffgbauafecaltwlxlqk h s lpgu"
matrix_dimension=sqrt(len(S3))
print matrix_dimension
S3_numbers=[]
for i in xrange(matrix_dimension):
    S3_numbers.append([get_number(x) for x in
S3[(matrix_dimension*i):(matrix_dimension*(i+1))]])
print S3_numbers
```

```
13
[[10, 14, 14, 18, 10, 23, 9, 26, 13, 24, 21, 2, 12], [18, 24,
19, 2, 5, 8, 21, 15, 1, 20, 13, 14], [20, 19, 10, 11, 25, 5,
19, 8, 18, 10, 1], [5, 8, 7, 10, 10, 5, 12, 0, 1, 16, 21, 24,
[16, 11, 2, 5, 9, 14, 4, 19, 0, 19, 5, 8, 3], [4, 25, 10, 19,
25, 19, 11, 11, 18, 0, 25, 6], [14, 26, 2, 21, 12, 15, 16, 15
15, 2, 4, 23], [26, 7, 3, 19, 20, 10, 7, 8, 13, 15, 24, 16, 1
16, 11, 10, 2, 11, 23, 4, 6, 0, 14, 17, 21], [24, 8, 19, 21,
10, 11, 20, 17, 1, 23, 5], [16, 22, 6, 20, 18, 6, 15, 18, 24,
6, 6], [7, 2, 1, 21, 1, 6, 5, 3, 1, 12, 20, 23, 12], [24, 12,
11, 0, 8, 0, 19, 0, 12, 16, 7, 21]]
```

```
R=IntegerModRing(27)
S3_matrix=Matrix(R,S3_numbers)
print S3_matrix
```

```
[10 14 14 18 10 23  9 26 13 24 21  2 12]
[18 24 20 19  2  5  8 21 15  1 20 13 14]
[20 19 10 11 25  5 24 16 19  8 18 10  1]
[ 5  8  7 10 10  5 12  0  1 16 21 24 10]
[16 11  2  5  9 14  4 19  0 19  5  8  3]
[ 4 25 10 19 18 25 19 11 11 18  0 25  6]
[14 26  2 21 12 15 16 15  7 15  2  4 23]
[26  7  3 19 20 10  7  8 13 15 24 16 11]
[ 4 16 11 10  2 11 23  4  6  0 14 17 21]
[24  8 19 21 25  8 10 11 20 17  1 23  5]
[16 22  6 20 18  6 15 18 24  3 22  6  6]
[ 7  2  1 21  1  6  5  3  1 12 20 23 12]
[24 12 17 11  0  8  0 19  0 12 16  7 21]
```

```
C3="fkosumcrnuzxtmldfgopoynlyogfuqfuxdwboifcqzihuppcfsaogwybbrrkf
bniasyljab xclywdictat edeq"
C3_length=len(C3)
C3_column_size=C3_length/matrix_dimension
C3_numbers=[]
for i in xrange(matrix_dimension):
    C3_numbers.append([get_number(x) for x in
C3[i:C3_length:matrix_dimension]])
C3_matrix=Matrix(R,C3_numbers)
print C3_matrix
```

```
[ 6 13  7  3  7 14 25]
```

```
[11 12  6 17 23  9 23]
[15  4 21 26 25  1  4]
[19  6 17  9  2 19  9]
[21  7  6  8  2 25  3]
[13 15 21 21 18 12 20]
[ 3 16 24 16 18 10  1]
[18 15  4 16 11  1 20]
[14 25 23  3  6  2  0]
[21 14  2  6 10  0  5]
[26 12 15 19  9 24  4]
[24 25  9  1  0  3  5]
[20 15  6 15  2 12 17]
```

```
if S3_matrix.is_invertible():
    P3_matrix=S3_matrix.inverse()*C3_matrix
    print P3_matrix
else:
    print "Not invertible."
```

```
[ 1 14 15 19  0  0  5]
[ 0  3 13  0 23  9  0]
[13  5  0  1  8 19 15]
[ 1  0 23 14 15  0  6]
[ 7 23 15  4 19  1  0]
[ 9  9 18  0  5  0 20]
[ 3 20  4  3  0 13  8]
[ 0  8 19  1 12 21  9]
[19  0  0 18  5 12 18]
[ 5 18  2  5 14 20 20]
[14  1  1  5  7  9  5]
[20 14 18 18 20 16  5]
[ 5  4  5 19  8 12 14]
```

```
def get_character(x):
    if x==0:
        return ' '
    else:
        return chr(ord('a')+x-1)
get_character(5)
```

```
'e'
```

```
P3_text=""
print C3_column_size,matrix_dimension
for j in xrange(C3_column_size):
    for i in xrange(matrix_dimension):
        P3_text+=get_character(int(P3_matrix[i][j]))
print P3_text
```

```
7 13
a magic sentence with random words bares and careers whose le
```

```
a multiple of thirteen
```

Exercise 4

```
p=113018525125668206946818626324387883237475371246081923938353469
gen_b=366374917808210333935581032389804936495416063284065731518 29
pub_b=41686074267466450131140570214368952610463023318931940044498
r_b=400953214663100002780597852457748176807557288335611381024798 9
sm=
[(967815718973445391602259886576382871956399077210580605292106510
450233680705233608309104809113220587823946349911705591034431183 23
(802275894535491745105903314982927177664644586309037735827588852 4
237950964094287606119148247145711462440472053514286803356571681 67
```

```
group_order=inverse_mod(gen_b,p)
print group_order
```

```
553556069294793277515659359860098979909159647369041156575799 5
717159633
```

```
#r_b=gen_b*s_a => s_a=r_b*gen_b^-1
s_a=mod(r_b*group_order,p)
print s_a
```

```
989750497911553522752521570616231968067386536537823692686879 5
130488225
```

```
R=IntegerModRing(p)
s_b=mod(pub_b*s_a,p)
print s_b
s=Matrix(R,[(1,s_b),(0,1)])
print s
```

```
101322858382833158349330147078523239215110812131434870483433 6
4945813674
[
1
101322858382833158349330147078523239215110812131434870483433 6
4945813674]
[
0
1]
```

```
C=Matrix(R,sm)
print C
```

```
[967815718973445391602259886576382871956399077210580605292106
8127956559
450233680705233608309104809113220587823946349911705591034431 1
078254640]
```

```
[802275894535491745105903314982927177664644586309037735827588
7743490757
237950964094287606119148247145711462440472053514286803356571
876774201]
```

```
if s.is_invertible():
    m=s.inverse()*C
    print [(m[0][0],m[0][1]),(m[1][0],m[1][1])]
else:
    print "Not invertible."
```

```
[(385577726502878566636175001194435403551610636928717289066509
89045217691,
541194732626222602503632965164079849674642782131809546143562
748184266),
(802275894535491745105903314982927177664644586309037735827588
7743490757,
237950964094287606119148247145711462440472053514286803356571
876774201)]
```

```
#This was for example file.
m==Matrix(R,
[(385577726502878566636175001194435403551610636928717289066509471
541194732626222602503632965164079849674642782131809546143562699222
(802275894535491745105903314982927177664644586309037735827588524
237950964094287606119148247145711462440472053514286803356571681677
```

```
True
```

Exercise 5

```
p=6646139978924579364519035301401722 97
Zp=Integers(p)
R=PolynomialRing(Zp,'x'); x=R.gen()

P_x= x^17 + 2*x^16 + 6646139978924579364519035301401722 93*x^15
+ 6646139978924579364519035301401722 96*x^14 + 45*x^13 +
6646139978924579364519035301401722 15*x^12 +
6646139978924579364519035301401720 26*x^11 + 355*x^10 + 70*x^9
+ 6646139978924579364519035301401718 68*x^8 + 2092*x^7 +
6646139978924579364519035301401714 97*x^6 +
6646139978924579364519035301401688 37*x^5 + 2334*x^4 + 1132*x^3
+ 6646139978924579364519035301401710 60*x^2 + 135*x + 59

QR=R.quotient(P_x,'x'); x=QR.gen()
QR
```

```
Univariate Quotient Polynomial Ring in x over Ring of integer
modulo 6646139978924579364519035301401722 97 with modulus x^17
```

```
    2*x^16 + 6646139978924579364519035301401722 93*x^15 +
    6646139978924579364519035301401722 96*x^14 + 45*x^13 +
    6646139978924579364519035301401722 15*x^12 +
    6646139978924579364519035301401720 26*x^11 + 355*x^10 + 70*x^9
    6646139978924579364519035301401718 68*x^8 + 2092*x^7 +
    6646139978924579364519035301401714 97*x^6 +
    6646139978924579364519035301401688 37*x^5 + 2334*x^4 + 1132*x^
    6646139978924579364519035301401710 60*x^2 + 135*x + 59
```

```
G=x+1

Y = 2935485193241331781946130152714272 66*x^16 +
4589582875910475349228219953013341 80*x^15 +
4935202416239420474426381811162743 63*x^14 +
3658386651807163812637840889220005 43*x^13 +
5975419406050055688904473683014211 26*x^12 +
3021948470318761004493202578195376 38*x^11 +
4793892657067915800063209009412132*x^10 +
4223654547857354876382130631204843 7*x^9 +
9532097613410492013371625949512875 3*x^8 +
5332277851472049131525021636235815 86*x^7 +
3981884678091587133071152808968913 76*x^6 +
3352238445998027047329723605069479 42*x^5 +
6636197396354908637362631242432746 82*x^4 +
1277811596705184543259542860946420 28*x^3 +
2287337596376820985431243285339152 91*x^2 +
1211797019294694594386786260008310 61*x +
7655676889710700495235094199795222

M = 5355171040831162015176950840004822 95*x^16 +
3874179916409286522695926391595148 80*x^15 +
4283633527692332354855070322651659 65*x^14 +
1873626403120969329261737282810699 87*x^13 +
6350811740795871654128868730716600 40*x^12 +
3767634119597142722296932681900501 0*x^11 +
5246683367055328444035063055308695 41*x^10 +
5638227245589661234773248299212596 79*x^9 +
4312262339135759634776714394813222 98*x^8 +
1034764290449722864066710463207834 45*x^7 +
5696977652990801314137308558347594 05*x^6 +
2220872846392622843001053666997451 16*x^5 +
5062942178750957487613094717117404 50*x^4 +
6019491422915479220858035495698823 3*x^3 +
1221234791534784782735666036279207 85*x^2 +
3286645880250286027239180071210324 43*x +
2199462054703388220764044951134527 99

z =
1613775815674869311375630698920881879833397557851587983742323250 4
```

```
n=QR.order()
print n
```

```
963121833542317369601573845406471472981683360100199734300899
293814186399883345401070571686123174645437897419119292589586
753644629471995914066214822797700965724232358308055886632471
235771668734216378766234566052519129331787236994939153054412
614792349780341249977129953594982604033986952282840473670045
039366674735590568991192734275939980302078403353613834837879
550083095051754941068051839163901981287437098131895484872975
848931109479958800080967620166708787286602378961626638359335
004857183466945013742505937328996505225942073141914818719252
```

```
print z<n
```

```
True
```

```
#Part 1: Encryption
Z_x = G^z
print Z_x
```

```
55533120129427738124083577996900428 2*x^16 +
20642053176574831241609283269099787 0*x^15 +
62177640006481378446889983512079432*x^14 +
56518437177675123897045474330965183 5*x^13 +
50236973932745686390144833334885311 7*x^12 +
37006912903042387382869179430774366 2*x^11 +
28125203390956381527607848770212496 2*x^10 +
38804821462203240970363751445318998 7*x^9 +
61368429816642130756636883532671710 2*x^8 +
35024897183729063936284183792944008 5*x^7 +
26398667498857999859794925537454731 4*x^6 +
25766056423493183338582340246287252 1*x^5 +
58539620540775085439404888091952450 7*x^4 +
32549539971958635135989643095210369 8*x^3 +
46149068555687440183718051623270607 8*x^2 +
39154970652651461330907732076690165 3*x +
30750121852547215519165809328935296 2
```

```
V_x = M * Y^z
print V_x
```

```
20967189794092914792294256238631900 9*x^16 +
15476930869583554354555338490500567 4*x^15 +
54575501230804646335543737467370907 3*x^14 +
21144415217651208400967107059640530 0*x^13 +
77999112146013085949887678918810926*x^12 +
95612731088120742381157620278365979*x^11 +
14532195156595576686883345893333765 0*x^10 +
21087799336792563595813153239997773 5*x^9 +
```

```
    348626189980952524746261332498447182*x^8 +
    199972138974010992191104654274705719*x^7 +
    116110894734551918950309264908401980*x^6 +
    389176223215285845732955709642637014*x^5 +
    292275728622946756756787155787752768*x^4 +
    189866294091339694630732401968844786*x^3 +
    266734264856114079549811017653975397*x^2 +
    829750948697801003950082940956137 6*x +
    462899371352491743468887537137863670
```

```
#Testing for the given examples

print Z_x == 635623438356136534698719406047258908*x^16 +
389645029160352302558285774896610769*x^15 +
487384667438494629801580220339638144*x^14 +
298327279075155327327333913817404676*x^13 +
435414987355068689198858744569612032*x^12 +
635021839036796108402041264413464 52*x^11 +
288110061940231595747663593774309488*x^10 +
228594074789209782078874778509631007*x^9 +
288341767785829774117496376838372793*x^8 +
330426386166029976885844519019343796*x^7 +
544632296042696124044295102691324152*x^6 +
197335077257476517480671342605082574*x^5 +
181011997812975335198297133107068043*x^4 +
465346927366779558110049881136675177*x^3 +
335525011590493265169519733810418324*x^2 +
227298247593159875751134802587 3379*x +
269089627539155308568290009805510440

print V_x == 380771381688583157047246347399681283*x^16 +
648585757445014051573950074485367977*x^15 +
134074556869733132432185655810256466*x^14 +
529319132755523452717903413495830374*x^13 +
226007175478571673706277006400132601*x^12 +
470989066499445778008858300398138058*x^11 +
379701338013906025985764010292807534*x^10 +
231372338425181452604426135011877943*x^9 +
272638552367220257649701219655844 88*x^8 +
520257511148203216140463791311053186*x^7 +
161282865991184814239789218019256 27*x^6 +
475515120130357222127507728704254199*x^5 +
546475124725562234165686571143393869*x^4 +
106431921062419011555603585371340918*x^3 +
617659943033669825134686604956082174*x^2 +
251391815775692031407149632460827811*x +
322527074862956608119285657663181605
```

```
True
```

```
True
```

```
#Part 2: Decryption
y =
43146387759590431379029034963902598387073925311586488993694170921

Z_x = 10398854005557906395217892979294163l*x^16 +
33684759703425816757245948866768478 3*x^15 +
34876595903600122862940731249545431 9*x^14 +
36695562876680313641776180464123603 1*x^13 +
52797572316904243806544373654261455 9*x^12 +
62015372188661773132271140785608385 1*x^11 +
39033999996979255820896623656729284 4*x^10 +
18885822347422544687306804025706985 1*x^9 +
39258709747224295564941450899395 60*x^8 +
41041588964293814411369780615651439*x^7 +
14304675199191955207027234209718155*x^6 +
62344785041803893045447843241930413 8*x^5 +
87150148441052743220664610431066719*x^4 +
47170908306195333488956093808344040 2*x^3 +
38639249882022069768318550696247157 9*x^2 +
81343704208811787161772822610604081*x +
24664260189337768800939685461643071 7

V_x = 24834140315866130657800880464723648 9*x^16 +
59032440330879963590326517639093538 4*x^15 +
29205808058521088067482474642969303 1*x^14 +
65748393404143948989843859222198130 5*x^13 +
36260773014947922561768722232978618 1*x^12 +
51648282491327218981178281503477942 1*x^11 +
97452475880443509978253031847991156*x^10 +
97254246406589054352567005475983353*x^9 +
15107555362743409347829199731453088 7*x^8 +
65355044652984516152819878065622548 3*x^7 +
12980075291275081451379733322905963 9*x^6 +
25598618278465141144660138574130169 4*x^5 +
63721621731509637363961432688582745 8*x^4 +
45136294229760697189044188991771436 1*x^3 +
34739100805745504235515695496214410 4*x^2 +
40764894106080692469401962119873624 7*x +
38453064825024619175923709067135583 4

inversepol = QR(Z_x^y)^-1
print inversepol, type(inversepol)
print (inversepol * Z_x^y) == 1
```

```
218075685584634915902529474621735405*x^16 +
208061675105996613571066005143822172*x^15 +
571871112103622991953895398869620 46*x^14 +
517092833083633228755379549414969924*x^13 +
```

```
131520922668056496688537818178125385*x^12 +
463384230693129024909270758286045l5*x^11 +
165204985326974327088883796293475714*x^10 +
284936742433651054191077254827580390*x^9 +
331949730017148317387668830231687924*x^8 +
285996659954843418094200118554620979*x^7 +
129113911713572528533188644332715962*x^6 +
182521024488805172730144778595917397*x^5 +
286354501176033480364826935250015473*x^4 +
286339149137373398692438920624090405*x^3 +
309871915005752851530575834019341799*x^2 +
181360701867547261739939834584256689*x +
169894250506748430330817177931279429 <class
'sage.rings.polynomial.polynomial_quotient_ring.PolynomialQuo
ng_generic_with_category.element_class'>
True
```

```
M_dec = V_x * inversepol
print M_dec
```

```
500824917999463097440434699282626 97*x^16 +
487803937066356854793166045195581227*x^15 +
519762253206997716393327361932024677*x^14 +
288842035661896976330875783786881002*x^13 +
442059680968334275764781244021575653*x^12 +
524446085346252725598380071492191 13*x^11 +
612990048036383932344139855762227140*x^10 +
159888549435046140298811135989696979*x^9 +
879314507995134435686427706725317 57*x^8 +
194921267549804021385831128017677625*x^7 +
317375764324685082954585980565085853*x^6 +
561707986032434902602025732876272330*x^5 +
342491194018356234885876177918910051*x^4 +
399420780145431332668860835760769323*x^3 +
486070406782533370336751386749179642*x^2 +
287709535283514538617445820908180531*x +
307676676863419298068681158304753036
```

```
#For testing
print M_dec == 288364327246375825383328782291449529*x^16 +
628387729796857591735790348498353114*x^15 +
348043520643159260989703288941376849*x^14 +
252674384318013412744273497043669693*x^13 +
591267634659778580878329701893250307*x^12 +
287605975191311906656336892787019804*x^11 +
565133470664654300173681141814928900*x^10 +
544103338522166921936639874839883972*x^9 +
519987639106641247484853290332322709*x^8 +
613251271385617350393905481460589014*x^7 +
```

```
312205839187781266784789217436761988*x^6 +
112464307822803887811879963443343155*x^5 +
188205521173287129665942549124865329*x^4 +
227757733185262959804766289276814959*x^3 +
636127347360639308188806717981043600*x^2 +
915817459931110588772100091274048 49*x +
357820911617688467905448597247098395
```

```
True
```