CS5823 Cryptography Project II

Code (SageMath):

p=2 #the first prime number

n=100 #the number of entries in set

p\_l=[] #prime\_list

check\_sum\_l=[]

# generate the prime number list

for i in range(100):

p\_l.append(p)

p=p.next\_prime()

id=113383597

S\_ori=10^94\*id #target

#generate the lattice M\_ori

M\_ori=matrix.identity(101)\*2 #matrix that diagonal is 2

for i in range(n):

M\_ori[i,n]=floor(10^100\*(p\_l[i].n(prec=600).nth\_root(5)))

v=matrix(1,n+1,lambda i,j: 1)

M\_ori[n]=v

M\_ori[n,n]=S\_ori

M=copy(M\_ori)

S\_error=99/100 #let S range from [S-S\_error(95.01),S(97),S+S\_error(97.99)]

for base\_exp in range(0,93,1):

base=id\*(10^base\_exp) #id number

exp=101-len(str(base))

factor\_M=10^exp

factor\_S=10^(exp+2)

for S\_dec in range(199):

for i in range(n):

M[i,n]=M\_ori[i,n]//factor\_M\*factor\_M #generate the round number of set in M

S=factor\_S\*(base-S\_error+S\_dec/100) #modified S

M[n,n]=S

#Ml=M.LLL() #using lll algorithm

Ml=M.BKZ() #using BKZ algorithm

flag=1 #check if the row contains the vector

for i in range(99,100,1):

if Ml[i,n]!=0:

continue

else:

for j in range(n):

if Ml[i,j]!=1 and Ml[i,j]!=-1:

flag=0

break

if flag == 0:

flag=1

continue;

else:

check\_sum\_ori=0

result\_l=[]

for j in range(n):

if Ml[i,j]==-1: #if the entry of vector is -1, then add the corresponding M

result\_l.append(1)

check\_sum\_ori=check\_sum\_ori+M\_ori[j,n]

else:

result\_l.append(0);

if check\_sum\_ori>=S\_ori:

check\_sum\_l.append(check\_sum\_ori);

print check\_sum\_ori

print 'result list:\n'+str(result\_l)

if check\_sum\_l!=[]:

print 'base\_exp: '+str(base\_exp)

print 'the result of subset sum:\n'+str(min(check\_sum\_l))

check\_sum\_l=[]

else:

print str(base\_exp)+' found no result'

print 'Finished'

Result:

base\_exp: 0

the result of subset sum:

1133835970574909764137075157805390095711178418581509021723341360376815356433900487050242116227726080091

base\_exp: 1

the result of subset sum:

1133835970020835797465835596483969614816676216575909155382964900743606109161125821898138868027004051132

base\_exp: 2

the result of subset sum:

1133835970009975121508938235167503339853446575182966014543307314299427584526047696339528947641104411889

base\_exp: 3

the result of subset sum:

1133835970000684705584365562218823708278663712538367211369389987450211047543551686579545864776926946404

base\_exp: 4

the result of subset sum:

1133835970000354787690249750379253695590255953169352362745737049267719957140857840871671687073832125747

5 found no result

base\_exp: 6

the result of subset sum:

1133835970000000077600833904752978572758799083641680371697620185631310614285282052850555308240321085095

7 found no result

8 found no result

...

92 found no result

93 found no result

So, the least number I found in M that greater than S is:

1133835970000000077600833904752978572758799083641680371697620185631310614285282052850555308240321085095,

then I add some code to print the binary vector, the result is:

[1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

General idea:

We denote the number of M as m\_ori and my id number is 113383597.

According the assignment, we know that the entry of the set is 101 digits, and target S is 103 digits. First we modify m\_ori by making the digits after the first 9 digits to be 0, denoted as m, and use BKZ algorithm to check if there is a subset sum of M that makes the result be in the range [113383597-0.99,113383597+0.99](\*10^94).

Note: the reason we use 0.99 is that after modification of m\_ori, the subset sum of m is actually equal to or less than the subset sum of m\_ori. The worst case is that the subset has the full set of M, which is 100 and the 2 digits after the first 9 digits are 9,9, then although the subset sum can be in the range [113383597-0.99,0]\*(10^94), but the corresponding subset sum of m\_ori can still reach 113383597\*(10^94).

If we can find the solutions of the subset problem in this range by checking if the matrix after BKZ contains the vector: the first 99 entries is 1 or -1 and the last entry is 0, and the result is greater than what we set, then it means that 113383597\*(10^94) can be the benchmark on our way to find the more accurate solution. Then I print the minimum number of solutions found in this range.

Next stage, I set the digits after the first 10 digits of m\_ori to be 0, and the range of target S be: [1133835970-0.99,1133835970+0.99]\*(10^93), and repeat the steps of above. And set the digits after the first 11 to 93 digits of m\_ori to be 0, repeat the above steps.

Notes:

BKZ has a better performance than LLL but takes more time. Here we use BKZ to get the better result.