RSA-1024

In [80]:
```
from __future__ import print_function
import random
```

In [81]:
```
N=1024
p = random_prime(2^(N//2), lbound = 2^(N//2 -1))
q = random_prime(2^(N//2), lbound = 2^(N//2 -1))
n = p*q
print('p=', p,'\n\n','q=',q,'\n\n''n=',n,'\n\n',
      'length of p (bits)=',len(bin(p)[2:]),'length of q (bits)=',len(bin(q)[2:]),'length of n (bits)=',len(bin(n)[2:]))
```

```
p= 12007341505326119672099628752346826310180941716045652429404576791344152217005098543950960564417493538535470033999746392563081999835128397582354532102224649

 q= 11003397322563371221055596327665644395147300412784351662394479169416604370363799139309396143051646513038872659116182373057216769780876901067270441233082377

n= 1321215493708094645821196912663535600850715178713200260631874700565551100577065299212042435515040885884573635939329881632205845674703238770847461742472187233220216494750744768327188976644982330591730939601035339361387528596348331965168974757395743994468568091669494104362338003702938318494305818409537176910673

 length of p (bits)= 512 length of q (bits)= 512 length of n (bits)= 1024
```

In [82]:
```
phi_n=(p-1)*(q-1)
phi_n
```

Out[82]:
```
1321215493708094645821196912663535600850715178713200260631874700565551100577065299212042435515040885884573635939329881632205845674703238770847461742472187210209477666861253875171963896632511625263488810771031247562331567835591744596271291497039036525328516517326800988433572383404168702489007168784563841603648
```

Choose e coprime to $\phi(n)$ and compute $d \equiv e^{-1} mod\ \phi(n)$

In [83]:
```
while True:
    e = random.randint(1, phi_n)
    if gcd(e,phi_n) == 1: break
d=inverse_mod(e,phi_n)
print('e=', e,'\n\n','d=',d)
```

```
e= 5407610127275631623927321211345303837161855978988704132018275630075916862850491577169784041339227916084861044057353738378136722733287356991638682737587769407589419401205490130344942222361086381037630406517444896824590613425531007986100409110079249893022468505664985485526526952230395491437279334422881762867

 d= 116446561133385655751292812500979337933709508701807908594795559743618699770555240572085973312910749046769177973705811557042745316133541897749706446474124786797029146909324405646844273028158745570314963215614931816076046700198900894029759940032571378702102395982087827789745598491910082811994822452982048379579
```

In [84]:
```
(d*e)%phi_n #check d is indeed the inverse of e
```

Out[84]:
```
1
```

Encryption

In [85]:
```
message=123
c=power_mod(message, e, n)
c
```

Out[85]:
```
120788664759211848985938800199406979171978018517769314697995445773482183303119809409554548552473551319377370288799241770106102715413005141156460740316907361329223835288991548943167893667566627807001071440313185724039908278227393129022302182111282037124269878882890978144442508210177761553255462932063720777528
```

Decryption

In [86]:
```
plaintext=power_mod(c, d, n)
plaintext
```

Out[86]:
```
123
```

Check whether a number is prime

In [87]:
```
x = random_prime(2^(512), lbound = 2^(512 -1)) #generate 512-bits prime randomly
y = random.randint( 2^(512-1),2^512) #generate 512-bits integer randomly
print('x=',x,'\n\n','length of x (bits)=',len(bin(x)[2:]),'\n\n','y=',y,'\n\n','length of y (bits)=',len(bin(y)[2:]))
```

```
x= 98040532185626305806384166547838251262527159872543136870462328611547633954168207979319396458554795960477147157916719017151006534050511091738703269906375 47
```

```
x= 9804053218562630580638416654783825126252715987254313687046232861154763395416820797931939645855479596047714715791671900
1715100653405051109173870326990637547

 length of x (bits)= 512

 y= 13172670030109598286976728264242929438427519261036990268885931915892016136008114482916732422713598254078557469563183
1918017244959370122004687177906007098843

 length of y (bits)= 512
```

In [88]:
```
is_prime(x), is_prime(y)
```

Out[88]:
```
(True, False)
```

Factor a big number

In [102]:
```
z = random.randint( 2^(100-1),2^100) #generate 512-bits integer randomly
print('z=',z,'\n\n','length of z (bits)=',len(bin(z)[2:]))
```

```
z= 789304437556587948711420377082

 length of z (bits)= 100
```

Method I

In [103]:
```
factor(z)
```

Out[103]:
```
2 * 3^3 * 41 * 83 * 675097 * 36245731 * 175535837023
```

Method II

In [104]:
```
from sage.rings.factorint import factor_trial_division
factor_trial_division(z,2^(30))#第二個參數請參照官方文件調整
```

Out[104]:
```
2 * 3^3 * 41 * 83 * 675097 * 36245731 * 175535837023
```

In [ ]:
```

```