Asymmetric cryptography

Student code: 194633IV

TASK 1:

In order to solve the task, I developed a program in Python3 (cf. Source code) to automate some tasks. The program will first find two random primes p and q between 1000 and 2000, then compute the modulus ***n=p*q*** and ***m=Phi(n)=(p-1)*(q-1)***. After that, I compute the public exponent e such as ***gcd(e,m)=1*** and e is the smallest possible. The private exponent d to compute would then be such as ***e.d=1 (mod n)***.

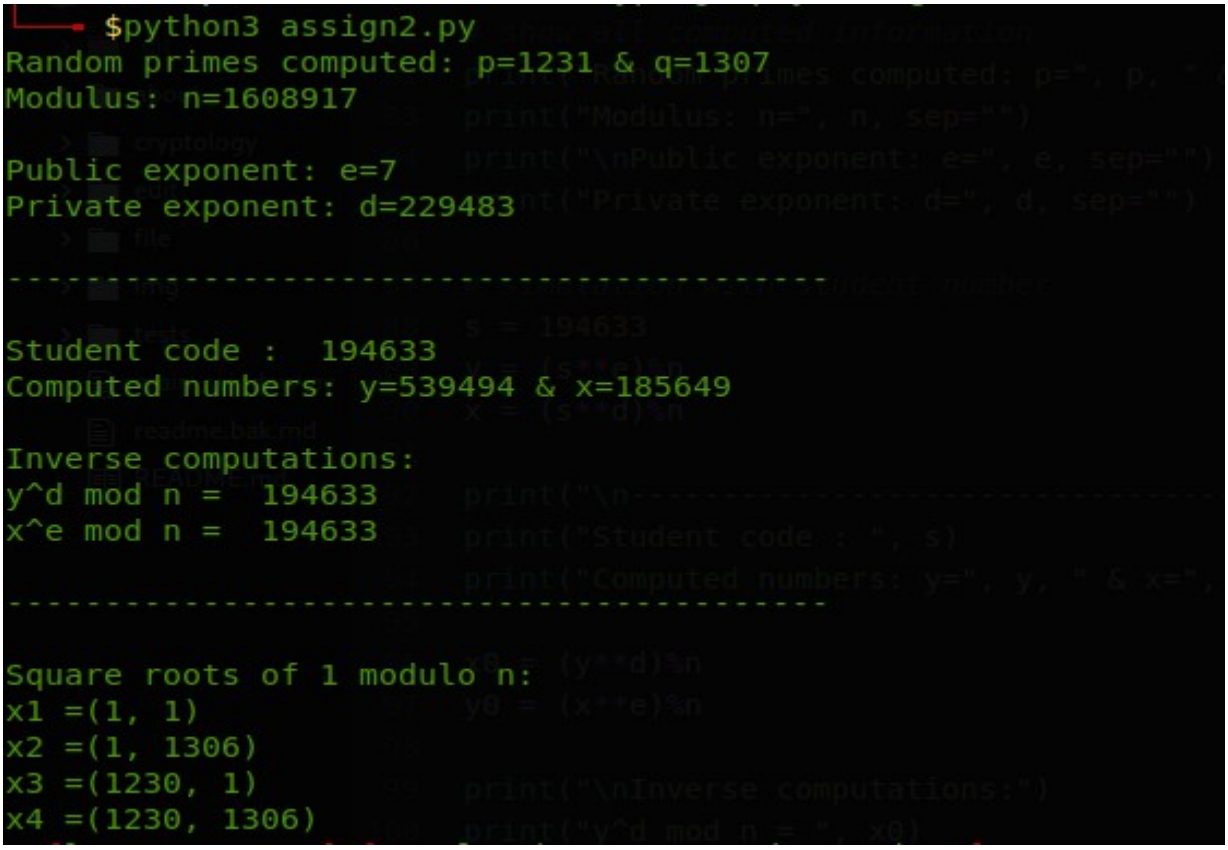
Once we have computed all required variables, we can use them for an encryption and decryption example. Therefore, the program will encrypt my student code (194633), and then decrypt it. By applying the two suggested formulas from the homework, we get two different ciphers. But when we apply the two inverse computations to these ciphers, we recover the same plain text, which is my student code.

TASK 2:

In a second phase, the program will compute the square roots of 1 modulus n, ie. all integers x such as ***x<n*** and $x^2 = 1 \pmod{n}$.

EXAMPLE:

The program I wrote will actually go through all the previously described steps, and print out a report about all computations done as follow.


$python3 assign2.py
Random primes computed: p=1231 & q=1307
Modulus: n=1608917
Public exponent: e=7
Private exponent: d=229483
Student code : 194633
Computed numbers: y=539494 & x=185649
Inverse computations:
y^d mod n = 194633
x^e mod n = 194633
Square roots of 1 modulo n:
x1 =(1, 1)
x2 =(1, 1306)
x3 =(1230, 1)
x4 =(1230, 1306)