Public Key Cryptography - Assignement 1

Raphael Gonon - Student number: 23040098

April 09th 2023

1 Source Code

1.1 GCD function

```
def gcd(a,b):
    #we know that gcd(a,b) = gcd(|a|,|b|)
    a = -a if a<0 else a
    b = -b if b<0 else b
    #we need to verify the condition x > y
    x = b if a < b else a
    y = b if a > b else a
    #initialisation how the remainder
    r = 1
    #loop to compute the gcd
    while r !=0:
        r = x%y
        x = y
        y = r
    return x
```

1.2 Extend GCD function

```
def extended_gcd(a,b):
    #initialisation of variables needed for computation
    x0,y0= (1,0)
    x1,y1 = (0,1)
    #We set u & v in order to preserver the value of a & b
    u = a
    v = b
    #loop to compute extend gcd
    while (v != 0):
        q = u // v
        x2,y2 = (x0 -q*x1),(y0 - q*y1)
        u = v
        v = a*x2 + b*y2
        x0,y0 = x1,y1
        x1,y1 = x2,y2
    #we return the value of rk-1 when rk ==0, with x and y
    return (u,x0,y0)
```

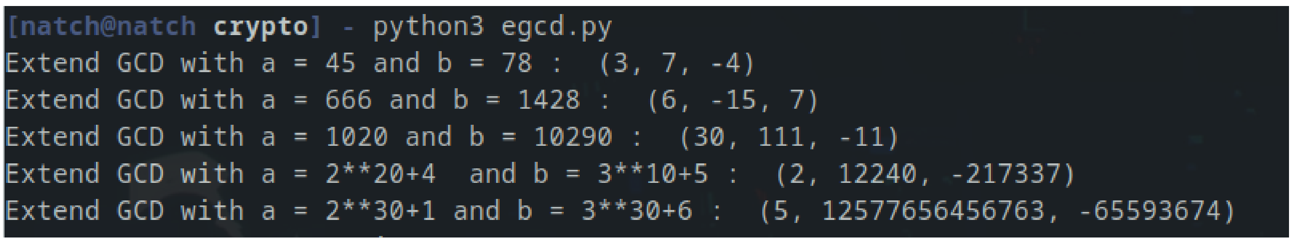

Figure 1: Output

2 Explaination

2.1 GCD Function

For the GCD function we first know that gcd(a,b) is equal to gcd(|a|,|b|). According to this, the first 2 ternary condition are used to ensure that a and b are in absolute value.

Therefore we wanted to be sure that we satisfy the condition a>b, in order to compute properly gcd(a,b). So the next 2 lines of the function are ternary conditions for switching a and b if there are not given such as a>b.

Following up the gcd algorithm, we now use a while loop in order to find the last remainder different form 0, which is the gcd of a and b. To do so we first compute the remainder of a modulo b, then we assign to x the value of y and to y the value of r (the remainder).

According to the while loop condition we need to finally return the value of x which his the last remainder different from 0.

2.2 Extend GCD function

First of all, we initialize variables needed for the algorithm :

1. x_0 and y_0 are the integers that satisfy $r_0 = a * x_0 + b * y_0$
2. x_1 and y_1 are the integers that satisfy $r_1 = a * x_1 + b * y_1$
3. The variables u and v are respectively set to a and b, thus we can keep the initial values of a and b.

Therefore, the while loop can be seen in two part: first the computation of q_{k-1}, x_k and y_k for the actual k, using the formula :

$$x_k = x_{k-2} - q_{k-1} * x_{k-1} \text{ for } x_k$$
$$y_k = y_{k-2} - q_{k-1} * y_{k-1} \text{ for } y_k$$

Second part, is the computation of the remainder for the rank k and the assignation of the variables for the next rank $k + 1$.

The condition for the loop is $v! = 0$, meaning that, while the remainder for the rank k is not equal to 0, the computation continues.

Finally, the function return the variables u, x_0, y_0, which are the gcd of a and b, and the value of x and y, such as $gcd(a, b) = a * x + b * y$.