

# Public Key Cryptography - Assignement 2

Raphael Gonon - Student number: 23040098

April 26th 2023



# 1 Source Code

## 1.1 ModExp function

```
def to_bin(n):
    s=""
    while n!=0:
        r = n%2
        s = str(r) + s
        n//=2
    return s

def mod_exp(a,e,n):
    b = to_bin(e)
    s = 1
    l = len(b)
    r=0
    for i in range(l):
        if b[i] == "1":
            r = (s*a) % n
        else:
            r = s
        s = (r**2) % n
    return r
```

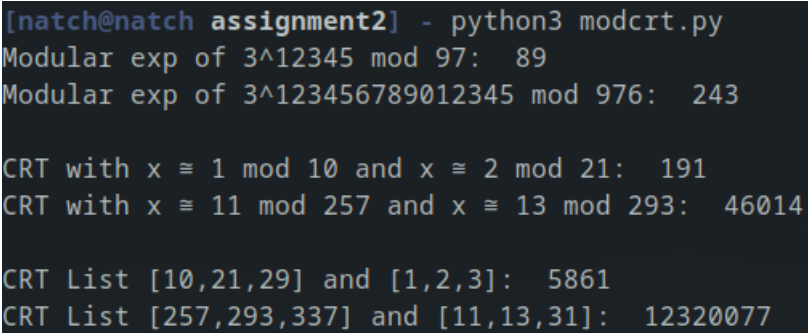
## 1.2 CRT function (Based on Extended GCD)

```
def extended_gcd(a,b):
    x0,y0= (1,0)
    x1,y1 = (0,1)
    u = a
    v = b
    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
    return (u,x0,y0)

def crt(p,q,a,b):
    res = extended_gcd(p,q)
    x = (b*p*res[1]) + (a*q*res[2])
    return x % (p*q)
```

### 1.3 CRT List function

```
def crt_list(primes,values):
    res = 0
    N = 1
    i= 0
    lenght = len(primes)
    while i < lenght:N*=primes[i]; i+=1
    for i in range(lenght):
        N_i = (N//primes[i])
        inv = extended_gcd(N_i,primes[i])[1]
        res = (res+ values[i]*inv* N_i)% N
    return res
```



```
[natch@natch assignment2] - python3 modcrt.py
Modular exp of 3^12345 mod 97:  89
Modular exp of 3^123456789012345 mod 976:  243

CRT with x ≡ 1 mod 10 and x ≡ 2 mod 21:  191
CRT with x ≡ 11 mod 257 and x ≡ 13 mod 293:  46014

CRT List [10,21,29] and [1,2,3]:  5861
CRT List [257,293,337] and [11,13,31]:  12320077
```

Figure 1: Output

## 2 Explanation

### 2.1 ModExp Function

The ModExp function is based on "square and multiply" method. The first step of square and multiply method is to convert exponent number in binary value. To do so we use the 'to\_bin(n)' function that convert the number n given in parameter in its binary representation. The function uses string format for the binary number in order to access any digits using index.

Once we have done this, we know create a loop to look at all digit of the binary representation. While doing so we can distinguish two cases:

1. If the actual digit is a "1", we multiply our current computing value by the original value of a, reduced by the modulo n operation:  $r = (s * a) \bmod n$
2. Otherwise we just kept the value that we compute at the previous step:  $r = s$
3. To obtain the value for this loop tour, we need to square the value that we just compute, and apply modulo n operation.

Finally we return the computing value 'r' before last squaring operation.

### 2.2 CRT function

The CRT function takes as input, two primes numbers (p,q) and the two corresponding remainder (a,b). The CRT is based on extended GCD algorithm run with two primes numbers p,q. Once this step done, we obtain value for s and k in the following equation:

$$gcd(p, q) = p * s + q * k$$

With those value we are now able to compute the wanted chinese remainder, by using the following formula:

$$chinese\_remainder = b * p * s + a * q * k \bmod p * q$$

Actually, as the 'extended\_gcd' function return a 3-uplet, we have to identify s as res[1] and k as res[2], in the source code. Also the code split the formula in two step, first compute the remainder and then reducing it with the modulo of p\*q.

### 2.3 CRT List function

#### 2.3.1 Principle

The source code of this function is based on the proof of the existence of a solution for a system of  $x \cong a_i \bmod n_i$ . Let's begin with a quick view of that proof.

First of all, we set  $N$  as the product of all  $n_i$  (primes list) of our system, and we define  $N_i$  such as  $N_i = N/n_i$  where  $n_i$  is the current modulo number.

Knowing that  $gcd(N, n_i) = 1$ , we compute the extended GCD with those values to obtain the following equation :

$$N_i * x_i + n_i * y_i = 1$$

Then using divisibility property and the definition of the congruence we obtain :

$$N_i * x_i \cong 1 \bmod n_i$$

We now have to multiply both side of the congruent sign by  $a_i$  :

$$N_i * a_i * x_i \cong a_i \text{ mod } n_i$$

From that last line, we can understand that the final value  $x$  will be equal to the sum of all  $N_i * a_i * x_i$  modulo  $N$ , for all  $i$  from 1 to  $n$  (number of values). Thus we have:

$$x = \sum_{i=1}^n N_i * x_i * a_i$$

To enter in the detail of that formula we know that :

1.  $N_i = N/n_i$  where  $N$  is the product of all primes
2.  $x_i$  is the modulo inverse of  $N_i$  modulo the corresponding  $primes[i]$
3.  $a_i$  is the given remainder  $values[i]$

### 2.3.2 Source Code Explanation

The CRT list function is based on the previous explanation. First of all we initialize some variable: 'res' to store the final result, 'N' will be the product of all the primes numbers, 'i' is a index variable and 'lenght' is the total numbers of primes (same for values), they are used for looping operation.

Then using a one line while loop, we compute the product of all primes, stored in  $N$ . After doing so, we are entering in the main loop of the code. The for loop is divide in three stage:

1. Computing the actual  $N_i$  according to the given equation  $N_i = N/n_i$
2. Computing the modulo inverse of  $N_i \cong x \text{ mod } n_i$ , by calculating `extended_gcd` with the euclidian algorithm
3. Adding to the `res` variable the product of the  $i^{th}$  values, modulo inverse and  $N_i$  modulo  $N$

Finally we return the `res` variable, which is the value that satifies  $x \cong values[i] \text{ mod } primes[i]$  for all values in lists.