Code RSA

par Léo Peyronnet

Décembre 2022

1 Réponses Exercices

Exercice 1
$N = 391$, $E = 151$ et $D = 7$

1. Message reçu et crypté : $C = 17$
   Soit M le message tel qu'envoyé (non crypté), alors :
   $M = C^D[N] = 17^7[391] = 204$.
2. On sait que $N = p \times q$ avec p, q deux nombres premiers. On a donc :
   $391 = p \times q = 17 \times 23$ *(résultat obtenu avec le programme cf 2.1)*
   Nous pouvons donc déduire $\varphi(N)$:
   $\varphi(N) = (p-1)(q-1) = 16 \times 22 = 352$
3. Nous connaissons la relation suivante : $E.D \equiv 1[\varphi(N)]$.
   Cette relation peut être vérifiée dans notre cas :
   $151 \times 7 \equiv 1[352] \leftrightarrow 151 \times 7[352] = 1$ *(vérifié avec le programme cf 2.1)*

Exercice 2

1. $N = 221$, $E = 11$ et $D = 35$
   (a) Soit $M = 112$ le message et C le message crypté, alors :
   $C = M^E[N] = 112^{11}[221] = 122$
   (b) Soit $C = 78$ le message reçu et m le message originel, alors :
   $M = C^D[N] = 78^{35}[221] = 65$
2. $p = 53$, $q = 71$
   (a) $N = 53 \times 71 = 3763$
   $\varphi(N) = 52 \times 70 = 3640$
   (b) $E = 307 : E < \varphi(N) \wedge pgcd(\varphi(N), E) = 1$
   — $307 < 3640$
   — $pgcd(\varphi(N), E) = 3640 \times (-7) + 307 \times 83 = 1$
   E est donc acceptable.
   $D = E^{-1}[\varphi(N)] = 83$
   (c) — Clé publique : $E = 307$ et $N = 3763$
   — Clé privée : $D = 83$ et N *(déjà connu avec la clé publique)*
   (d) Les éléments restants sont p et q. Étant les générateurs de N, ils doivent être dissimulés car ils sont de fait les détenteurs de l'asymétrie du code RSA. Pour rappel, afin de pouvoir décoder et lire RSA, il faut posséder D qui est l'inverse modulaire de E modulo $(p-1)(q-1)$.

Le processus pour déterminer p et q à partir de N est gourmand en ressources (cf. 2.1). Ainsi, plus p et q seront grands, plus le décodage par *"brute force"* demandera de ressources temporelles ou spatiales.

Exercice 3
$E = 257$, $N = 1073$, $D = 353$.

1. Chiffrer "METHODE" :
   Correspond à 12

2 Annexes

2.1 Programme solution de l'exercice 1

```
import math
def erathosthene(n):
    t=[]
    r=[]
    t+=[False]
    t+=[False]
    for i in range(2,n):
        t+=[True]
    for i in range(2,int(math.sqrt(n))):
        j=2*i
        while j<len(t):
            t[j]=False
            j=j+i
    for i in range(2,n):
        if t[i]:
            r+=[i]
    return r
def scan(tab,n):
    for i in range(len(t)):
        for y in range(len(t)):
            if tab[i]*tab[y]==n:
                return [tab[i],tab[y]]
    return False

e=151
d=7
n=391

print("e=",e,",␣d=",d,",␣n=",n,sep="")
print("==========================")
t=erathosthene(300)
t=scan(t,n)
print("p=",t[0],"␣et␣q=",t[1],sep="")
phi=(t[0]-1)*(t[1]-1)
```

```
print("phi(N) = ",phi)
print("E*D%phi = ",e*d%phi)
```
