# RSA

Pascal Devougess

8 février 2012

## 1 Introduction

Considérons $\boxed{p_1 \text{ et } p_2}$ deux grands entiers nombres premiers distincts (il existe des algorithmes qui génèrent « rapidement »de grands nombres pseudo-premiers.)
Posons $\boxed{n = p_1 \times p_2}$. (il n'existe pas d'algorithme assez efficace permettant de retrouver $p_1$ et $p_2$ à partir de $n$.)

---

Prenons un nombre $\boxed{e \in \mathbb{N}}$ premier avec $(p_1 - 1)(p_2 - 1)$

---

On sait déjà, d'après le théorème de Bezout, qu'il existe des couples d'entiers $(u; v)$ tels que $eu - (p_1 - 1)(p_2 - 1)v = 1$.
On peut choisir $u > 0$ (et même dans l'intervalle $]0; (p_1 - 1)(p_2 - 1)[$) et notons le $d$.
Ainsi : $\boxed{ed = 1 + (p_1 - 1)(p_2 - 1)v}$ (1) ~~avec $ed \in [\![(p_1 - 1); (p_2 - 1)]\!]$~~.

---

Soit $X$ un bloc numérique. On peut supposer que $X \in \{0; 1; 2; \ldots; n - 1\}$.
$X \xrightarrow{\text{codage}} Y$ reste de la division euclidienne de $X^e$ par $n$ ou encore $X^e \equiv Y\ [n]$ et $Y \in [\![ 0; n - 1 ]\!]$. $Y \xrightarrow{\text{décodage}} Z$ reste de la division euclidienne de $Y^d$ par $n$

autrement dit $Y^d \equiv Z\ [n]$ et $Z \in [\![ 0; n - 1 ]\!]$.

**Théorème – 1.1**

*On a* $\boxed{Z = X}$ *(c'est le bon décodage ; on retrouve $X$ à partir de $Y$).*

***Remarques* 1.1**
D'abord on remarque que pour coder $X$ en $Y$ il est nécessaire d'avoir $n$ et $e$. Ces deux nombres sont *publiques* (clefs publiques). Par contre pour retrouver $X$ à partir de $Y$ il est nécessaire d'avoir $d$ Or on ne peut avoir $d$ que si on connaît $(p_1 - 1)(p_2 - 1)$. c'est à dire si on connaît $p_1$ et $p_2$, ce qui est justement actuellement impossible (en temps raisonnable).

*Preuve du théorème.* Comme $X$ et $Y$ sont des entiers dans $[\![ 1; n - 1 ]\!]$. Alors

$$Z = X \iff Z \equiv X\ [n] \iff Y^a \equiv X\ [n] \iff X^{ed} \equiv X\ [n]$$

car $Z \equiv Y\ [n]$ et $Y \equiv X^e\ [n]$

- Démontrons d'abord que : $\boxed{X^{ed} \equiv X\ [p_1]}$ (2)

1. **Premier cas :** $\boxed{p_1 \mid X}$ dans ce cas $X \equiv 0\ [p_1] \Longrightarrow\ X^{ed} \equiv 0\ [p_1] \Longrightarrow$ $X^{ed} \equiv X\ [p_1]$.
2. **Deuxième cas :** $p_1$ ne divise pas $X$ et dans ce cas $p_1$ est premier avec $X$ et comme $p_1$ est premier , d'après le petit théorème de Fermat, on a $X^{p_1-1} \equiv 1\ [p_1]$. On a donc $X^{(p_1-1)(p_2-1)v} \equiv 1\ [p_1]$ et donc $X^{(p_1-1)(p_2-1)v+1} \equiv X\ [p_1]$. Au final $X^{ed} \equiv X\ [p_1]$ avec (1).

– De même on démontre que $\boxed{X^{ed} \equiv X\ [p_1]}$ (3)

– **Concluons** :
  – De (2) on déduit que $p_1 \mid X^{ed} - X$
  – De (3) on déduit que $p_2 \mid X^{ed} - X$
  – En conséquence, puisque $p_1$ et $p_2$ sont premiers entre eux (ce sont deux nombres premiers distincts) $p_1 p_2 \mid X^{ed} - X \Longrightarrow n \mid X^{ed} - X \Longrightarrow$ $\boxed{X^{ed} \equiv X\ [n]}$ (c.q.f.d)

□

## 2 Annexe : un programme pour résoudre l'activité du livre

Le livre est Hyperbole Math spéciaité (2002). le programme est en Python.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#     RSA.PY --- RSA terminale S
#     Title   : rsa
#     Author : guy Yeterian
#     File    : rsa.py in the path /home/guy/prog/python/lycee/rsa.py
#     Date    : Mon Feb  6 2012
#
#     Keywords :  RSA *
#

# code begin here
r"""" le programme qui suit permet de resoudre ACTIVITe 5 page
79 du livre Hyperbole Math specialite 2002"""


import string


sepletter = ' '
sepword = '  '



def num2letter (x):
    """
    num2letter:
    x is a number.
    return the letter 1 -> A,...    26 -> Z
    """
    return (chr(x+64))

def letter2num (w):
    """
    letter2num:
    w is a char and return the number of char w
    """
    return (ord(w) - 64)


def encrypt (ch,e,n):
```

```
    """
    encrypt:
    encrypt char ch and return a number modulo n
    """
    return ((ch** e) % n)

def decrypt (num,d,n):
    """
    decrypt:
    return the number numm ** d modulo n
    """
    return ((num** d) %  n)



def message_code(message,e,n):
    """
    return message with number in string coded
    """
    s = string.splitfields(message)
    mess = []
    word = ''
    for w in s:
        m = []
        for l in w:
            l = letter2num(l)
            l = encrypt(l,e,n)
            #print str(l)
            m.append(str(l))
        word=string.join(m,sepletter)
        #print word + '\n'
        mess.append(word)
    return string.join(mess,sepword)



def message_decode (mess,d,n):
    """
    message_decode:
    mess is th message
    d is the decode key
    n the number
    """
    word=''
    mes=[]
    s = string.split(mess,sepword)
    #print s
    for w in s:
        w = string.split(w,sepletter)
        mot=[]
        for l in w:
            #print l
            l = string.atoi(l)
            l = decrypt(l,d,n)
            l = num2letter(l)
            mot.append(l)
            #print mot
        word=string.join(mot,'')
        mes.append(word)
    return string.join(mes,sepword)

#rsa.py ends here
```