Ejercicio propio RSA

March 5, 2018

```
In [1]: t = 'Este texto es una basura que solo vale para probar la encriptacion RSA'

In [2]: alfb = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

In [3]: L_alfb = list(alfb)
        print L_alfb

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S'


In [4]: def ord2(c):
            return L_alfb.index(c)

In [5]: def chr2(n):
            return L_alfb[n]

In [49]: l = len(t); l

Out[49]: 70

In [50]: p = next_prime(26^35)
         p

Out[50]: 3342467390895094933117731769437449324322008701339 1

In [51]: q = next_prime(26^35 + 24765743543265783465314534526 34)
         q

Out[51]: 3342467390895094933365389204870107158975154046609 3

In [52]: print abs(p-q)

24765743543265783465314534527 02


In [53]: n = p*q
         print n
         print (n > 26^70 and n < 26^71)
```

```
111720882591970633514294807815627490011755810739736063034716624862568125707006459605452288847
True
```

```
In [54]: phi = (p-1)*(q-1)
         phi

Out[54]: 111720882591970633514294807815627490011755810739729378099934834672701642586032152048
```

```
In [55]: def calcular_primo(phi):
             i = 5
             while(1):
                 if gcd(i, phi) == 1:
                     return i
                 else:
                     i = next_prime(i)
```

```
In [56]: coprimo = calcular_primo(phi)
         print coprimo

7
```

```
In [57]: EX = xgcd(7, phi)
         print EX

(1, 319202521691344667183699450901792828605016602113512508856956670493433264531520434425625690
```

La clave pública es n=111720882591970633514294807815627490011755810739736063034716624862568125707007 y coprimo=7 ---> clavePublica = (111720882591970633514294807815627490011755810739736063034716624862568117 7)

La clave privada es EX[1] = 319202521691344667183699450901792828605016602113512508856956670493433264

```
In [58]: def prepararTexto(texto):
             L = list(texto)
             M = list()
             for item in L:
                 item = item.capitalize()
                 if item in L_alfb:
                     M.append(item)
             cad = "".join(M)
             return cad
```

```
In [59]: texto = prepararTexto(t)
         texto

Out[59]: 'ESTETEXTOESUNABASURAQUESOLOVALEPARAPROBARLAENCRIPTACIONRSA'
```

```
In [60]: def codificar(texto):
             COD = [ord2(c) for c in texto]
             return COD

In [61]: textCod = codificar(texto)
         print textCod
         print len(textCod)

[4, 18, 19, 4, 19, 4, 23, 19, 14, 4, 18, 20, 13, 0, 1, 0, 18, 20, 17, 0, 16, 20, 4, 18, 14, 11
58


In [62]: m = ZZ(textCod, 26)
         print m

32340170792058078203555899105562996872070888229526890147848186211594981844744700


In [63]: def encriptarRSA(mensaje, exponentePublico, moduloPublico):
             ENC = power_mod(m, exponentePublico, moduloPublico)
             DIG = ENC.digits(base = 26)
             print len(DIG)
             M = "".join([chr2(n) for n in DIG])
             print len(M)
             return M

In [64]: M_ENC = encriptarRSA(m, coprimo, n)
         print M_ENC
         print len(M_ENC)

70
70
GGRKACIZHVTKDWUIGQOURIQFVALOFNXDTUPMVBOGYVQUJUYEQPXBXPCZSGOEFZETOUJIZI
70


In [65]: def desencriptar(mensajeEncriptado, exponentePublico, moduloPublico):
             AUX1 = [ord2(c) for c in mensajeEncriptado]
             print len(AUX1)
             AUX2 = ZZ(AUX1, 26)
             DESEN = power_mod(AUX2, exponentePublico, moduloPublico)
             return DESEN

In [66]: M_DESEN = desencriptar(M_ENC, EX[1], n)
         print M_DESEN

70
32340170792058078203555899105562996872070888229526890147848186211594981844744700
```

```
In [67]: def descodificar(m):
             M = m.digits(base = 26)
             print len(M)
             L = "".join([chr2(n) for n in M])
             print len(L)
             return L

In [68]: t2 = descodificar(M_DESEN)
         print t2
         print len(t2)

57
57
ESTETEXTOESUNABASURAQUESOLOVALEPARAPROBARLAENCRIPTACIONRS
57
```

£Por qué razón se perderá una letra?