CRIPTOGRAF?A 2 - RSA-FirmaDigital & Zn(23-02-2018)

February 23, 2018

EJERCICIO 3 - Utilizando RSA para comunicarse con el profesor

```
In [51]: p1 = next_prime(sqrt(16^64))
         p2 = next_prime(p1)

         if((p1*p2)>(16^64) and (p1*p2)<(16^65)):
             print p1
             print p2
```

```
340282366920938463463374607431768211507
340282366920938463463374607431768211537
```

```
In [17]: p1*p2 #esta será mi clave privada
```

```
Out[17]: 115792089237316195423570985008687907898187257099204441216623032188906533556259
```

```
In [18]: n = p1 * p2
```

```
In [19]: phi = (p1-1)*(p2-1)
         phi
```

```
Out[19]: 115792089237316195423570985008687907897506692365362564289696282974042997133216
```

```
In [27]: for i in xsrange (7, phi):
             if(gcd(i, phi) == 1):
                 break
         e = i
         e
```

```
Out[27]: 11
```

```
In [28]: EX = xgcd(11, phi)
         EX
```

```
Out[28]: (1,
          10526553567028745038506453182607991627046062942305687662699662088549363375747,
          -1)
```

Tenemos de clave pública la tupla (11579208923731619542357098500868790789818725709920444121662303218811) y de clave privada 10526553567028745038506453182607991627046062942305687662699662088549363375747

```
In [40]: alfb = "0123456789ABCDEFG"

In [41]: L_alfb = list(alfb)
         print L_alfb

['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F', 'G']


In [42]: def ord2(c):
             return L_alfb.index(c)

In [49]: ord2('A')

Out[49]: 10

In [45]: def chr2(n):
             return L_alfb[n]

In [46]: chr2(10)

Out[46]: 'A'

In [50]: C = "11579208923731619542357098500868790789818725709920444121662303218890653355625911
```

Llegado este momento es cuando me doy de cuenta de que había que utilizar el sistema de ecriptacion Firma Digital, en lugar de RSA, ya que el tamaño de palabra es mayor que 64 digitos. Se dejará esta ejercicio a medias aquí, el cual será ehcho de nuevo una vez se haya explicado Firma Digital la proxima semana.

EJERCICIO 9

```
In [55]: def lista_coprimos(n):
             L = list()
             for i in xsrange(2, n):
                 if gcd(i, n)==1:
                     L.append(i)
             return L

In [56]: print lista_coprimos(15)
         print lista_coprimos(2)

[2, 4, 7, 8, 11, 13, 14]
[]


In [57]: def generador(p):
             if is_prime(p):
                 LC = lista_coprimos(p);
                 a = randint(0, len(LC)-1)
                 return LC[a]
             else:
                 return "p no es primo"
```

```
In [59]: print generador(3)
         print generador(6)
```

```
2
p no es primo
```

```
In [12]: def lista_iguales(L1, L2):
             if len(L1) != len(L2):
                 return False

             A1 = copy(L1)
             A2 = copy(L2)
             for item in A1:
                 if item not in A2:
                     return False
                 else:
                     A1[A1.index(item)] = -1
                     A2[A2.index(item)] = -1
             return True
```

```
In [13]: L1 = [1, 2, 3, 5, 4]
         L2 = [5, 3, 4, 2, 1]
         print lista_iguales(L1, L2)
         L3 = [1, 1, 2, 2, 2]
         L4 = [2, 2, 1, 1, 1]
         print lista_iguales(L3, L4)
         L5 = [1, 2, 3, 5, 4]
         L6 = [5, 3, 4, 2, 11]
         print lista_iguales(L5, L6)
         L7 = []
         L8 = []
         print lista_iguales(L7, L8)
```

```
True
False
False
True
```

```
In [14]: def es_generador(g, n):
             LC = lista_coprimos(n)
             L = list()
             i=1
             while(1):
                 x = ((g^i)%n)
                 if x not in L and x!= 1:
                     L.append(x)
                     i += 1
```

```
                else:
                    break
            if lista_iguales(LC, L):
                return True
            else:
                return False
```

```
In [15]: print es_generador(6, 8)
         print es_generador(3, 7)
```

```
False
True
```

```
In [16]: def es_ciclico(n):
             if n==2: #caso especional
                 return True
             for i in xsrange (2,n):
                 if es_generador(i, n):
                     return True
             return False
```

```
In [17]: print es_ciclico(8)
         print es_ciclico(11)
```

```
False
True
```

```
In [18]: for i in xsrange(2, 101):
             print (i, es_ciclico(i), factor(i))
```

```
(2, True, 2)
(3, True, 3)
(4, True, 2^2)
(5, True, 5)
(6, True, 2 * 3)
(7, True, 7)
(8, False, 2^3)
(9, True, 3^2)
(10, True, 2 * 5)
(11, True, 11)
(12, False, 2^2 * 3)
(13, True, 13)
(14, True, 2 * 7)
(15, False, 3 * 5)
(16, False, 2^4)
(17, True, 17)
(18, True, 2 * 3^2)
```

```
(19, True, 19)
(20, False, 2^2 * 5)
(21, False, 3 * 7)
(22, True, 2 * 11)
(23, True, 23)
(24, False, 2^3 * 3)
(25, True, 5^2)
(26, True, 2 * 13)
(27, True, 3^3)
(28, False, 2^2 * 7)
(29, True, 29)
(30, False, 2 * 3 * 5)
(31, True, 31)
(32, False, 2^5)
(33, False, 3 * 11)
(34, True, 2 * 17)
(35, False, 5 * 7)
(36, False, 2^2 * 3^2)
(37, True, 37)
(38, True, 2 * 19)
(39, False, 3 * 13)
(40, False, 2^3 * 5)
(41, True, 41)
(42, False, 2 * 3 * 7)
(43, True, 43)
(44, False, 2^2 * 11)
(45, False, 3^2 * 5)
(46, True, 2 * 23)
(47, True, 47)
(48, False, 2^4 * 3)
(49, True, 7^2)
(50, True, 2 * 5^2)
(51, False, 3 * 17)
(52, False, 2^2 * 13)
(53, True, 53)
(54, True, 2 * 3^3)
(55, False, 5 * 11)
(56, False, 2^3 * 7)
(57, False, 3 * 19)
(58, True, 2 * 29)
(59, True, 59)
(60, False, 2^2 * 3 * 5)
(61, True, 61)
(62, True, 2 * 31)
(63, False, 3^2 * 7)
(64, False, 2^6)
(65, False, 5 * 13)
(66, False, 2 * 3 * 11)
```

```
(67, True, 67)
(68, False, 2^2 * 17)
(69, False, 3 * 23)
(70, False, 2 * 5 * 7)
(71, True, 71)
(72, False, 2^3 * 3^2)
(73, True, 73)
(74, True, 2 * 37)
(75, False, 3 * 5^2)
(76, False, 2^2 * 19)
(77, False, 7 * 11)
(78, False, 2 * 3 * 13)
(79, True, 79)
(80, False, 2^4 * 5)
(81, True, 3^4)
(82, True, 2 * 41)
(83, True, 83)
(84, False, 2^2 * 3 * 7)
(85, False, 5 * 17)
(86, True, 2 * 43)
(87, False, 3 * 29)
(88, False, 2^3 * 11)
(89, True, 89)
(90, False, 2 * 3^2 * 5)
(91, False, 7 * 13)
(92, False, 2^2 * 23)
(93, False, 3 * 31)
(94, True, 2 * 47)
(95, False, 5 * 19)
(96, False, 2^5 * 3)
(97, True, 97)
(98, True, 2 * 7^2)
(99, False, 3^2 * 11)
(100, False, 2^2 * 5^2)
```

```
In [19]: def conjetura(n):
             #casos especional es n==4 porque su único coprimo es el 3 y es 3 genera Z4 = 3.
             if n==4:
                 return True

             if is_prime(n):
                 return True

             if is_power_of_two(n):
                 return False

             L = list(n.factor())
```

```
            if len(L)==1: #si es potencia de un primo != 2 --> TRUE
                return True
            if L[0][0] == 2 and L[0][1] == 1 and len(L) == 2:
                return True
            return False
```

```
In [20]: for i in xsrange(2, 101):
            print (i, es_ciclico(i), conjetura(i))
```

```
(2, True, True)
(3, True, True)
(4, True, True)
(5, True, True)
(6, True, True)
(7, True, True)
(8, False, False)
(9, True, True)
(10, True, True)
(11, True, True)
(12, False, False)
(13, True, True)
(14, True, True)
(15, False, False)
(16, False, False)
(17, True, True)
(18, True, True)
(19, True, True)
(20, False, False)
(21, False, False)
(22, True, True)
(23, True, True)
(24, False, False)
(25, True, True)
(26, True, True)
(27, True, True)
(28, False, False)
(29, True, True)
(30, False, False)
(31, True, True)
(32, False, False)
(33, False, False)
(34, True, True)
(35, False, False)
(36, False, False)
(37, True, True)
(38, True, True)
(39, False, False)
(40, False, False)
```

```
(41, True, True)
(42, False, False)
(43, True, True)
(44, False, False)
(45, False, False)
(46, True, True)
(47, True, True)
(48, False, False)
(49, True, True)
(50, True, True)
(51, False, False)
(52, False, False)
(53, True, True)
(54, True, True)
(55, False, False)
(56, False, False)
(57, False, False)
(58, True, True)
(59, True, True)
(60, False, False)
(61, True, True)
(62, True, True)
(63, False, False)
(64, False, False)
(65, False, False)
(66, False, False)
(67, True, True)
(68, False, False)
(69, False, False)
(70, False, False)
(71, True, True)
(72, False, False)
(73, True, True)
(74, True, True)
(75, False, False)
(76, False, False)
(77, False, False)
(78, False, False)
(79, True, True)
(80, False, False)
(81, True, True)
(82, True, True)
(83, True, True)
(84, False, False)
(85, False, False)
(86, True, True)
(87, False, False)
(88, False, False)
```

```
(89, True, True)
(90, False, False)
(91, False, False)
(92, False, False)
(93, False, False)
(94, True, True)
(95, False, False)
(96, False, False)
(97, True, True)
(98, True, True)
(99, False, False)
(100, False, False)
```

Según esto nuestra conjetura es correcta. Esta conjetura consiste en que n es cíclico si es primo; y aparte, si es una potencia única de un primo distinto de dos (por ejemplo 27=3^3), o si en su factorización solo aparecen dos primos y uno de ellos es un 2 con multiplicidad 1.

EJERCICIO 10

```
In [75]: p = nth_prime(1000)
         print p
```

```
7919
```

```
In [76]: a = randint(1, p-2) #nos da igual
         print a
```

```
6574
```

```
In [80]: g = generador(p)
         print es_generador(g, p)
         A = (g**a)%p
         print g; print A
```

```
True
1132
2769
```

```
In [81]: def encripta(p, g, A, m):
             b = randint(1, p-2)
             B = (g**b)%p
             c = ((A**b)*m)%p
             return B,c
```

```
In [83]: def desencripta(p, a, B, c):
             x = p-1-a
             m = ((B**x)*c)%p
             return m
```

```
In [84]: m_antes = randint(p//2, p-2)

         print "p = "+str(p)
         print "g = "+str(g)
         print "A = "+str(A)
         print "M antes = "+str(m_antes)

         B, c = encripta(p, g, A, m_antes%p)
         print "B = "+str(B)
         print "c = "+str(c)
```

```
p = 7919
g = 1132
A = 2769
M antes = 4962
B = 780
c = 5809
```

```
In [85]: print "p = "+str(p)
         print "a = "+str(a)
         print "B = "+str(B)
         print "c = "+str(c)

         m_despues = desencripta(p, a, B, c)

         print "M antes = "+str(m_antes)
         print "M despues = "+str(m_despues)
         m_antes == m_despues
```

```
p = 7919
a = 6574
B = 780
c = 5809
M antes = 4962
M despues = 4962
```

```
Out[85]: True
```

Muy buenas noticias, se ha realizado la encriptacion y la desencriptacion con exito

```
In [ ]:
```