Práctica 4 SPSI

Rafael Lachica Garrido

Ejercicio 5)Evaluar cuanto tiempo emplean estos programas en factorizar números en la forma p*q, donde p y q son números primos en las condiciones RSA, generados previamente con los programas antes examinados. Hacer dos grupos de programas para poder realizar mediciones coherentes sobre la longitud de los dígitos: por un lado factor, msieve y yafu, y por otro DisMat, Fortaleza, GenRSA y ExpoCrip. Msieve y yafu dan directamente el tiempo empleado. Para factor utilizar alguna rutina de crono (ptime, …)

Para ejecutar yafu usamos la opción -v:

Factor Msieve y Yafu:

Ndigitos	9	11	14	18	28	32	41	46	57	74
Factor	0.001	0.003	0.04	0.052	0.087	0.8	0.226	1.068	12.847	278
Yafu	0	0	0	0	0	1	0	0	3	2
Msieve	0.0312	0.0156	0.0468	0.0475	0.0937	0.1094	0.125	0.234	3.578	29.92

Tiempo medido en segundos.

Yafu es el que se mantiene más estable y el que presenta el menor tiempo de ejecución para dígitos muy grandes.

Fortaleza y DistMat:

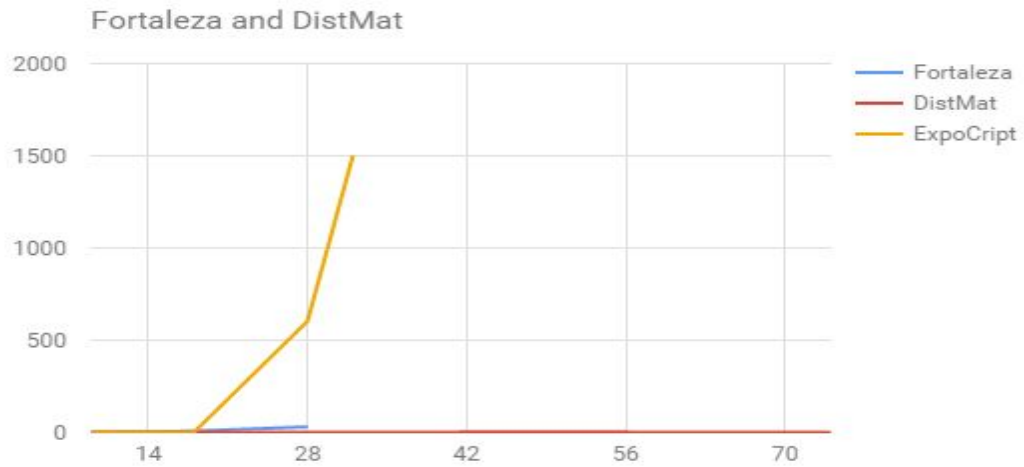
Ndigitos	9	11	14	18	28	32	41	46	57	74
Fortaleza	0	1	2	7	30					
DistMat	0	0	0	0	0	0	1	2	1	1
ExpoCript	0	0	0	0	603	1502				

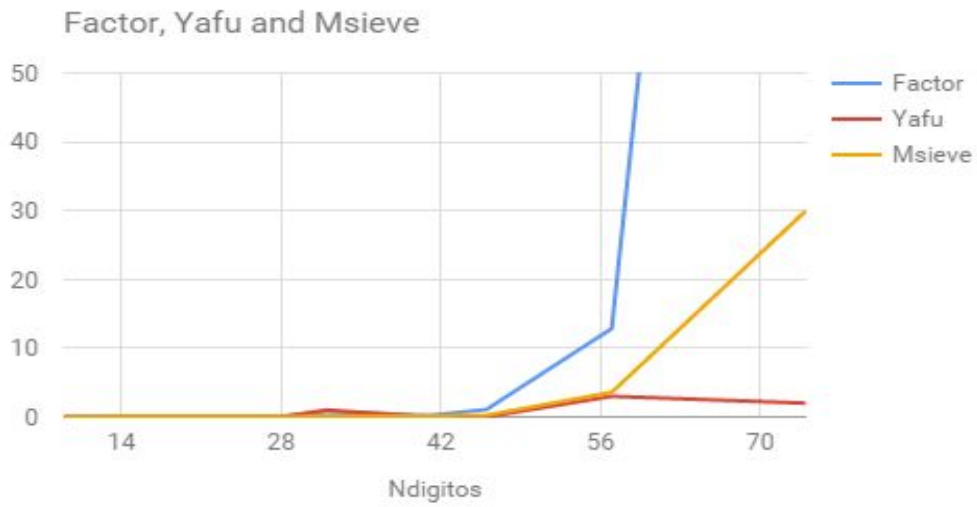
Tiempo medido en segundos.

ExpoCript no es capaz de encontrar factores mayores mayores de 32 dígitos, (por lo menos en mi PC) al igual que Fortaleza.

Ejercicio 6) Representar los datos anteriores en una gráfica, aproximándolos por funciones con variable x = número de dígitos (base 10) del número a factorizar. Realizarlo con las rutinas de factorización que sean coherentes para su comparación en función del número de dígitos factorizado. Para la aproximación usar software de aproximación matemática (MatLab, Xmgrace, FreeMat, GmatH, SAGE, …).

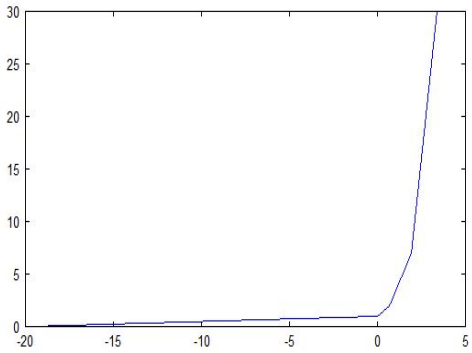
Datos asociados:



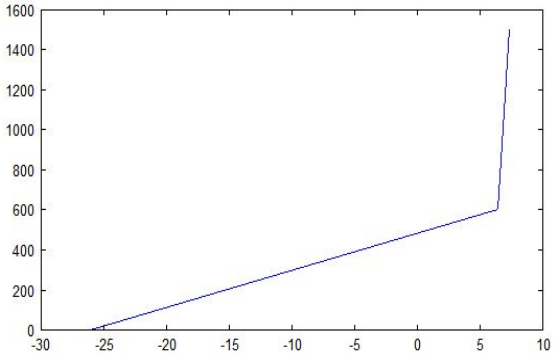


En freemat:

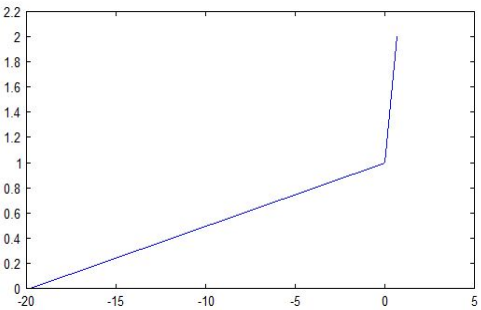
Fortaleza



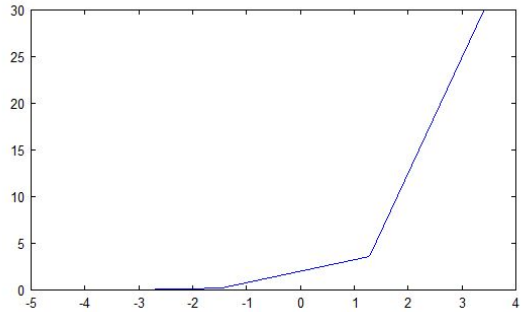
ExpoCript



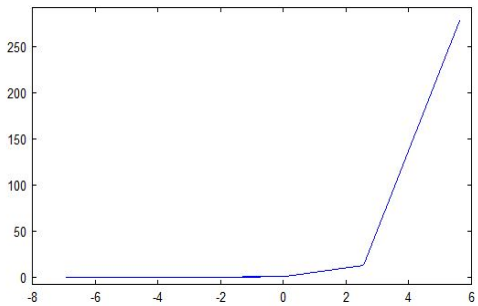
Dismat



Yafu



Factor



Msieve

Ejercicio 7) Hacer una estimación, en base a las funciones de aproximación anteriores de cuánto tiempo tardaría en factorizarse números de longitud 512 bits, 1024 bits y 2048 bits. Obtener las conclusiones pertinentes.

Obteniendo una función exponencial aproximada a través de la gráfica anterior, con yafu y factor, y msieve, obtenemos una media de los resultados y la siguiente función exponencial:
exp = 1.5^(ndigitos*0.188)

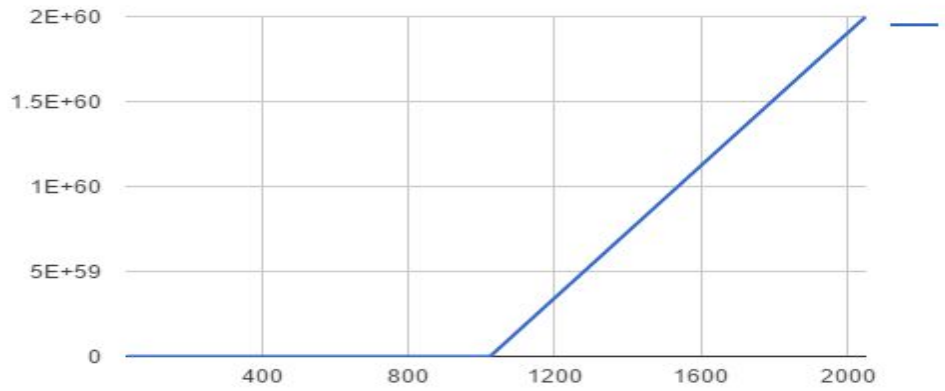
Obtenemos los siguientes resultados:

74	90	115	130	145	512	1024	2048
281.696594	953.8149016	6413.469309	20121.96417	63131.73456	8.90923E+16	7.93744E+33	6.3003E+67

Tiempo en segundos

Para 512 = 8.9*10^16 segundos
1024 = 7.93 * 10^33
2048 = 6.30 * 10^67

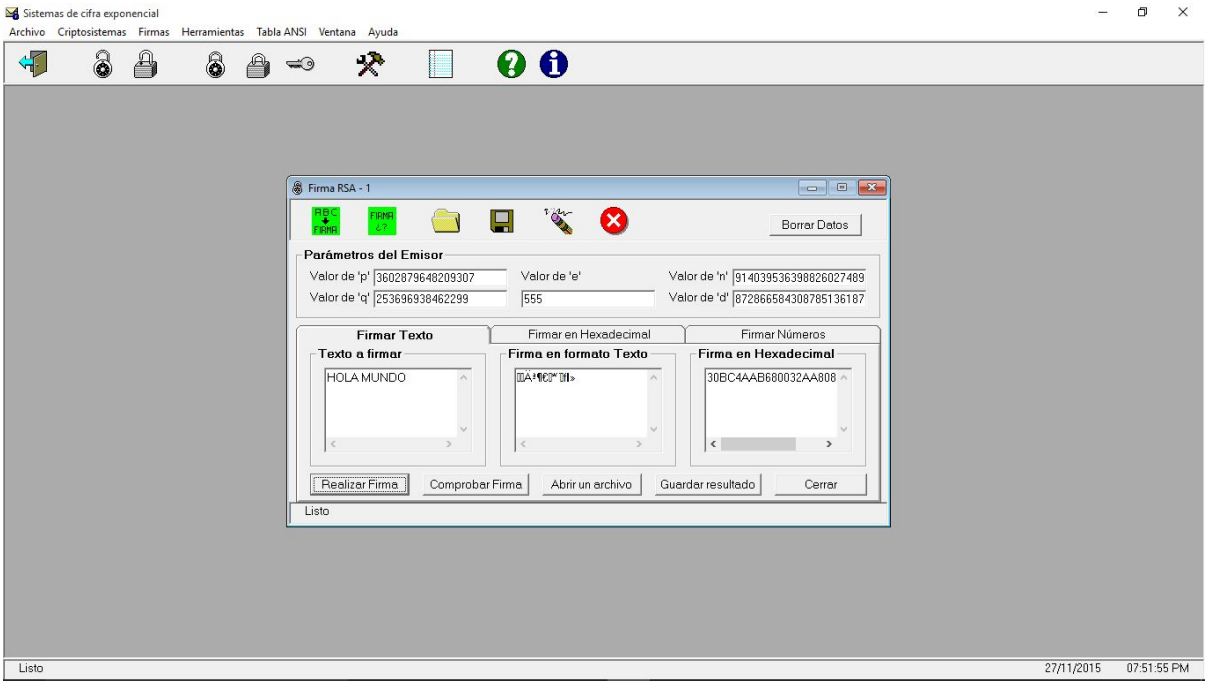
Gráfica sería así:



Ejercicio 8)Selecciona un protocolo criptográfico de entre los explicados en clase y utiliza el software suministrado para construir un ejemplo de utilización del mismo

con datos que no permitan su rotura por ataque sobre las características algorítmicas del protocolo: Factorización o Logaritmo Discreto.

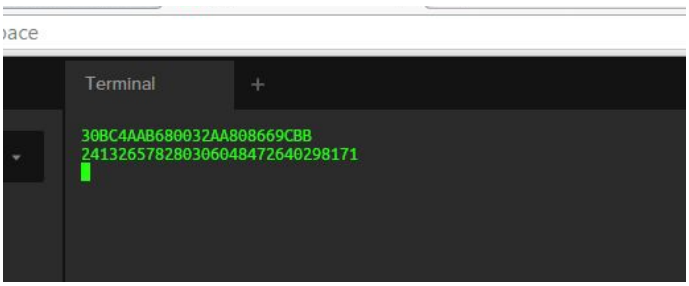
Usaremos el cifrado de firma digital con GenRSA.
Para el cifrado usamos dos números de la lista de primos fuertes:
p = 3602879648209307 **q** = 253696938462299 **valor e** = 555
Pulsamos comprobar firma y ciframos.



Firmado texto en hexadecimal = 30BC4AAB680032AA808669CBB

Valor n = 914039536398826027489480416793

Procedemos ahora al ataque, suponemos que tenemos la firma en hexadecimal, para atacarla tenemos que pasarla a decimal:
Usamos el software de los sitemas unix bc para pasar a formato decimal un texto hexadecimal.



Tenemos en decimal el número: **241326578280306048472640298171**
Ahora procederemos a atacar obteniendo los factores con **yafu:**

n = 27454673296963145446261 6949
(253696938462299 * 3602879648209307) = N

Vemos que nos da exactamente los mismos factores que usamos para la firma, lo cual quiere decir que está expuesta a ataques de fuerza bruta, debido a que se factoriza fácilmente los números primos elegidos.

Ejercicio 9) Se ha podido acceder a un dispositivo de cifrado con el algoritmo RSA. Se sabe que la clave pública consiste en:

N=A72F076B4F3603AF02DAE60BCD3DEF9918141E77C1F29E54E84C8D794 B6A3FA 59B2F119713D13B
e = 4773
Y se han podido interceptar los dos bloques cifrados siguientes:

C1=0151380B3AED56D9E3311809C0FD2323FBE6D8370E1D17AE19E55AAA B312F28 D954DB391E93800CD

C2=02E41517800E59C337AE015886E4FEE4CDB33DACBBE9AD4B45EC78D1 953E983 37BB57B9DCEA10048

¿Cuál es el mensaje transmitido? Describir el ataque empleado y el tiempo de ejecución del mismo. Explicar la elección de las herramientas software en función de las características del sistema.

1) Pasamos de hexadecimal a decimal con bc -q ibase=16

En decimal:

5448947611675552326885167045146271832020970317915785046050870427770
3456659319436750444489485 39

2) Buscamos los factores con factor:

```
          phase 2 - trying last prime less than 2000000
curve  80 phase 1 - trying all primes less than 20000
          phase 2 - trying last prime less than 2000000
finally - the multiple polynomial quadratic sieve - with large prime (*)
using multiplier k= 3 and 6561 small primes as factor base
working...  6459
trying...
working...* 6460
trying...
PRIME FACTOR      4643560723644339509171197067108860964441 41
PRIME FACTOR      168342441135411724605878733436 7

C:\Users\rafaellg8\Dropbox\GII\SPSI\Practica4>
```

q1 = 4643560723644339509171197067108860964441 41

q2 = 168342441135411724605878733436 7

e decimal = 18291

Probamos ahora la clave con expocript