

## Banco de pruebas

## Banco de pruebas genRSA v2.1

Todas las pruebas aquí indicadas se han realizado con un ordenador portátil marca Dell modelo Latitude E5470 con procesador Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz y 8GB de memoria RAM. El sistema operativo instalado es un Windows 7 Enterprise 64-bit.

### Generación Manual

La generación manual permite generar claves RSA decimales o hexadecimales sin importar la longitud en bits de la misma. A continuación se puede ver el ejemplo de una clave decimal de 24 bits:

- Primo p: 6.709
- Primo q: 1.567
- Clave pública e: 5

The screenshot shows the 'genRSA - Generación de claves RSA' application window. The 'Generación Manual' tab is active. The interface is divided into several sections:

- Clave RSA (Private Components):**
  - Numero primo p: 6.709 (dec, 13 Bits)
  - Numero primo q: 1.567 (dec, 11 Bits)
  - $\Phi(n)$ : 10.504.728 (dec, 24 Bits)
  - Clave privada d: 6.302.837 (dec, 23 Bits)
- Componentes públicos RSA (Public Components):**
  - Módulo n: 10.513.003 (dec, 24 Bits)
  - Clave pública e: 5 (dec, 3 Bits)
- Test de primalidad (Primality Test):**
  - Iteraciones: (empty)
  - Número primo p: ?
  - Número primo q: ?
  - Tiempo: 0.000 Seg
- Generar Clave Automáticamente (Generate Key Automatically):**
  - Longitud de Clave: (empty) Bits
  - Tiempo: 0.006 Seg
  - ☐ Clave pública = 65.537
  - ☐ p y q igual tamaño
  - ☐ Primos seguros
  - Generación Automática button
- Claves Privadas Parejas - CPP (Private Key Pairs - CPP):**
  - 1.050.473 -> 21 bits
  - 2.801.261 -> 22 bits
  - 4.552.049 -> 23 bits
  - 8.053.625 -> 23 bits
  - 9.804.413 -> 24 bits
  - Cantidad de Claves: 5
- Números No Cifrables - NNC (Uncipherable Numbers - NNC):**
  - Cantidad de NNC: 15
  - Generar Log button

Buttons at the top: Archivo, Generar Clave, Operaciones, Test Primalidad, Ataques, Unidades, Ayuda. Buttons in the middle: Generación Manual, Limpiar Datos, Salir de genRSA. A 'Limpiar Datos' button is also at the bottom right.

Se han comprobado los resultados de los componentes de la clave obtenidos en genRSA v2.1 realizando los cálculos con ayuda de la web MobileFish:

*Big number converter:*

[https://www.mobilefish.com/services/big\\_number/big\\_number.php](https://www.mobilefish.com/services/big_number/big_number.php)

*Big number equation calculation:*

[https://www.mobilefish.com/services/big\\_number\\_equation/big\\_number\\_equation.php](https://www.mobilefish.com/services/big_number_equation/big_number_equation.php)

*Big number bitwise calculation:*

[https://www.mobilefish.com/services/big\\_number\\_bitwise\\_calculation/big\\_number\\_bitwise\\_calculation.php](https://www.mobilefish.com/services/big_number_bitwise_calculation/big_number_bitwise_calculation.php)

- Módulo  $n = p \times q = 10.513.003$

$$n = 6.709 \times 1.567 = 10.513.003$$

- Máximo común divisor ( $e, \Phi(n)$ ) = 1

$$\Phi(n) = (p-1) \times (q-1) = 6.708 \times 1.566 = 10.504.728$$

$$\text{mcd}(5, 10.504.728) = 1$$

- $\Phi(n)$  mayor que la clave pública y la clave pública mayor que 1.

$$10.504.728 > 5 > 1$$

- $d = \text{inv}[e, \Phi(n)] = 6.302.837$

$$d = \text{inv}[5, 10.504.728] = 6.302.837$$

No obstante, también se van a comprobar las claves privadas parejas (CPP) y los números no cifrables (NNC) asociados a la clave.

- Claves privadas parejas: la clave privada  $d$  es única en  $\Phi(n)$  pero no es el único valor que descifra en  $n$ .

Para comprobar que la cantidad es realmente el número indicado en la aplicación, primero se calcula la primera clave pareja  $d_y$ .

$$d_y = \text{inv}(e, \gamma), \text{ siendo } \gamma = \text{mcm}[(p-1), (q-1)]$$

$$\gamma = \text{mcm}(6.708, 1.566) = 1.750.788$$

Web: <https://www.calculatorsoup.com/calculators/math/lcm.php>

$$d_y = \text{inv}(5, 1.750.788) = 1.050.473$$

A continuación se calcula la cantidad de claves privadas parejas.

$\lambda = \text{parte entera de } [(n - d_y) / \gamma]$

$$\begin{aligned}\lambda &= ((10.513.003 - 1.050.473) / 1.750.788) = \\ &= (9.462.530 / 1.774.278) = 5\end{aligned}$$

Una vez comprobado que la cantidad de claves y la primera coinciden se comprueba el resto de claves, tal que  $d_i = d_y + i \times \gamma$ , donde  $i = 0, \dots, \lambda$

$$\text{Clave } 0 = d_0 = 1.050.473 + 0 \times 1.750.788 = 1.050.473$$

$$\text{Clave } 1 = d_1 = 1.050.473 + 1 \times 1.750.788 = 2.801.261$$

$$\text{Clave } 2 = d_2 = 1.050.473 + 2 \times 1.750.788 = 4.552.049$$

$$\text{Clave } 3 = d_3 = 1.050.473 + 3 \times 1.750.788 = 6.302.837 \text{ (Privada)}$$

$$\text{Clave } 4 = d_4 = 1.050.473 + 4 \times 1.750.788 = 8.053.625$$

$$\text{Clave } 5 = d_5 = 1.050.473 + 5 \times 1.750.788 = 9.804.413$$

- Números No Cifrables: la aplicación muestra los números no cifrables asociados y permite generar el fichero de log.

En cuanto a la cantidad, se comprueba que es correcta calculando  $\sigma_n = [1 + \text{mcd}(e-1, p-1)] \times [1 + \text{mcd}(e-1, q-1)]$ .

$$\begin{aligned}\sigma_n &= [1 + \text{mcd}(4, 6.708)] \times [1 + \text{mcd}(4, 1.566)] = \\ &= [1 + 4] \times [1 + 2] = 15\end{aligned}$$

Web: <https://www.calculatorsoup.com/calculators/math/gcf.php>

Comprobado que la cantidad es correcta se genera el fichero de Log. Este indica que los NNC son: 0, 1, 136.330, 1.120.404, 1.256.733, 4.559.970, 4.696.299, 4.696.300, 5.816.703, 5.816.704, 5.953.033, 9.256.270, 9.392.599, 10.376.673, 10.513.002. Se puede comprobar que son correctos, utilizando la operación de cifrar que ofrece la propia aplicación (más adelante se comprobará el correcto funcionamiento de la funcionalidad de cifrado).

Una vez comprobado que los datos mostrados al generar la clave de forma manual son correctos, se procede a comprobar que se han habilitado todas las funcionalidades de la aplicación que se encontraban deshabilitadas:

- Generación del fichero de Log de NNC.
- Guardar la clave en un fichero HTML.
- Realizar operaciones de Cifrado-Descifrado y Firma-Validación.
- Realizar los tres tipos de ataque con los datos de la clave.

## Test de Primalidad

Los test de primalidad se han de poder ejecutar se haya creado o no una clave. Además deben dejar la aplicación en un estado coherente después de ejecutarse. Si se ejecuta sobre los primos de una clave, después se debe permitir realizar operaciones de cifra y firma; si se ejecuta sobre números introducidos sin existir una clave, estas operaciones no deben permitirse (al igual que ninguna de las funcionalidades que habilita la generación de una clave).

El número de iteraciones de ambos tipos de test deben estar comprendidas entre 1 y 300, pudiendo tomarse el valor 50 como número de iteraciones cuyo resultado tiene una fiabilidad sobrada.

Los dos tipos de implementaciones para ejecutar el test de primalidad son: el algoritmo de Fermat y el algoritmo de Miller Rabin combinado con el de Luchas-Lehmer.

El algoritmo de Miller Rabin y Lucas-Lehmer es muy rápido en ejecución. Sin embargo, el algoritmo de Fermat puede llegar a demorarse bastante más cuando el número de iteraciones es alto o la clave tiene una longitud mayor que 2.048 bits.

Para comprobar que los resultados son correctos se han hecho dos grupos de pruebas.

El primer grupo de pruebas consiste en probar números que ya se sabe que son primos. Se comprobará que el resultado indique que el número es primo para ambos algoritmos y con un número de iteraciones igual a 1, 50 y 300.

- Número 1, 20 bits: 963.097
- Número 2, 40 bits: 802.531.780.577
- Número 3, 128 bits:  
206.022.852.512.717.161.155.800.602.892.466.466.027
- Número 4, 512 bits:  
13.302.234.470.614.217.204.793.688.246.321.864.759.812.857.969.8  
66.885.120.212.272.507.338.410.462.753.235.484.338.062.389.026.2  
58.357.078.305.228.326.176.576.532.983.080.807.609.059.660.259.1  
92.952.449.763

El segundo grupo de pruebas consiste en probar números que se sabe que son compuestos. Estos números serán los módulos de diversas claves que se generarán. La prueba se basará en pasar ambos test de primalidad a números de distinta longitud de bits y comprobar que el resultado indica que no son

números primos. Además todos los números se comprobarán para 1, 50 y 300 iteraciones.

- Número 1, 20 bits: 641.737
- Número 2, 40 bits: 737.304.039.083
- Número 3, 128 bits:  
211.279.548.627.169.082.596.241.022.037.888.000.997
- Número 4, 512 bits:  
11.123.175.549.804.993.308.746.373.791.189.175.807.007.447.831.9  
32.598.943.746.092.445.552.757.926.599.044.109.456.406.791.612.9  
34.381.949.242.184.333.527.585.839.484.989.304.177.336.008.496.4  
37.129.967.769

El resultado de ambas pruebas ha sido satisfactorio. Con lo cual se puede afirmar que los Test de Primalidad funcionan correctamente.

## Ataque Cíclico

El ataque por cifrado cíclico pretende romper el secreto de un mensaje cifrado, es decir, obtener el mensaje en claro. Para ello solo se utilizarán componentes públicos de la clave.

Para obtener el mensaje en claro se van a realizar cifrados sucesivos del mensaje cifrado conociendo la clave pública  $e$  y el módulo  $n$ . Cuando se obtenga de nuevo el mismo mensaje cifrado significará que el valor del cifrado anterior es el secreto que da solución al ataque.

A continuación se va a realizar el seguimiento del ataque para el mensaje cifrado 116.943 y la clave cuyo módulo  $n$  es 714.559 y la clave pública es 418.547. Los resultados de este ataque con el software genRSA v2.1 se pueden ver en la siguiente imagen.

genRSA - Ataque Cíclico

### Ataque Cíclico

Nº de Cifrados  ☐ Hasta que prospere

Datos Ataque Cíclico

Módulo n

Exponente

Mensaje Cifrado

Mensaje Original

Comenzar Continuar

Información Limpiar Datos

#### Resultados

Mensaje a descifrar = 116.943

c0 = 80.652  
c1 = 487.933  
c2 = 598.425  
c3 = 623.425  
c4 = 467.772  
c5 = 257.274  
c6 = 130.655  
c7 = 639.555  
c8 = 443.577  
c9 = 568.583  
c10 = 350.023  
c11 = 173.398  
c12 = 693.592  
c13 = 600.843  
c14 = 2  
c15 = 116.943

Mensaje descifrado en la vuelta 15

M. Original Recuperado

Tiempo Total  Seg

El mensaje recuperado da como resultado el mensaje 2, pero es el valor cifrado 116.943 el que se ataca:

- $c_0 = 116.943^{418.547} \bmod 714.559 = 80.652$
- $c_1 = 80.652^{418.547} \bmod 714.559 = 487.933$
- $c_2 = 487.933^{418.547} \bmod 714.559 = 598.425$
- $c_3 = 598.425^{418.547} \bmod 714.559 = 623.425$
- $c_4 = 623.425^{418.547} \bmod 714.559 = 467.772$
- $c_5 = 467.772^{418.547} \bmod 714.559 = 257.274$
- $c_6 = 257.274^{418.547} \bmod 714.559 = 130.655$
- $c_7 = 130.655^{418.547} \bmod 714.559 = 639.555$
- $c_8 = 639.555^{418.547} \bmod 714.559 = 443.577$
- $c_9 = 443.577^{418.547} \bmod 714.559 = 568.583$
- $c_{10} = 568.583^{418.547} \bmod 714.559 = 350.023$
- $c_{11} = 350.023^{418.547} \bmod 714.559 = 173.398$
- $c_{12} = 173.398^{418.547} \bmod 714.559 = 693.592$
- $c_{13} = 693.592^{418.547} \bmod 714.559 = 600.843$
- $c_{14} = 600.843^{418.547} \bmod 714.559 = 2$
- $c_{15} = 2^{418.547} \bmod 714.559 = \underline{116.943}$

Tras realizar el seguimiento al ataque, se puede concluir que los resultados obtenidos son correctos.

Por último se realizarán diversos ataques al mensaje cifrado = 123.456, en ellos las claves tendrán en común el valor de la clave pública (65.537). En la tabla se muestran el número de vuelta en la que prospera el ataque, el tiempo empleado y la media de cifrados por segundo.

Bits	Módulo n	Vuelta	Tiempo(seg)	Cifrados/Seg
20	1.014.647	5.879	0,298	-
30	725.998.433	1.591.589	2,545	625.378
35	19.054.972.543	2.419.559	3,717	650.944
40	763.494.120.509	16.476.095	23,069	714.209
50	567.208.772.618.027	59.620.319	79,356	751.301

Para poder comprobar que el número de vueltas es correcto se recomienda utilizar la opción de ataque cíclico sin haber generado una clave previamente. Por otro lado, si se quieren obtener los primos p y q se puede realizar el ataque por factorización a los módulos de la tabla.

### Ataque Factorización

El ataque por factorización permite obtener los primos p y q que conforman el módulo n. Para ello, el programa hará uso únicamente del algoritmo Pollar rho.

Para comprobar el correcto funcionamiento de este ataque se han generado distintas claves de manera automática y se ha comparado el resultado del ataque por factorización con los primos p y q de la generación. En este caso se van a emplear números hexadecimales para ejecutar la prueba.

El programa genRSA v2.1 no cambia de algoritmo conforme se modifican las unidades de la clave (decimal o hexadecimal). Esta premisa es válida para todas las funcionalidades de la aplicación.



Bits	Módulo N hexadecimal	Primo p hexadecimal	Primo q hexadecimal	Tiempo
50	174FF2758CD2D	14442ED1	12679D	0,131 s
75	25941FB5E836ED484BF	1E113D293	13FF3DE84A5	0,441 s
90	299EDF83183F1D0168DD32D	19D4CD1A919	19C7ADA217935	10,813 s
100	D610D7F871FED1E2EE74DF50F	396F4407B9F9	3BA2492384A947	34,906 s
110	1E1323FF96DC0425914014A9B483	441558026689D	71158158919379F	4m 1s
115	CBD3AAE460F49598742B796C5D7F8F	EA29F0C555F1D1	DED582519572535F	8m 25s

## Ataque Paradoja del Cumpleaños

**Ataque Paradoja del Cumpleaños**

Datos Ataque

Módulo n: 899

Exponente: 389

Mensaje M: 2

Cifrados Estimados: 87

Cifrados Probados: 56

Media Cifrados/Seg: 56

Comenzar Pausar

Información Limpiar Datos

**Resultados**

Columna I

mensaje<sup>0</sup> mod Módulo

2<sup>0</sup> mod 899 = 1

Columna J

mensaje<sup>(Módulo/2)+1</sup> mod Módulo

2<sup>450</sup> mod 899 = 497

Cifrados sucesivos

Col I --> 2<sup>1</sup> mod 899 = 2

Col J --> 2<sup>451</sup> mod 899 = 95

Col I --> 2<sup>2</sup> mod 899 = 4

Col J --> 2<sup>452</sup> mod 899 = 190

Col I --> 2<sup>3</sup> mod 899 = 8

Col J --> 2<sup>453</sup> mod 899 = 380

Col I --> 2<sup>4</sup> mod 899 = 16

Col J --> 2<sup>454</sup> mod 899 = 760

Clave Privada: 149

Tiempo Total: 0,018 Seg

Para realizar este ataque es preciso haber generado una clave previamente o introducir los valores módulo y clave pública.

Primer caso de ataque. A continuación se procede a comprobar que el resultado es correcto. Datos del ataque: Mensaje M = 2. Datos de la Clave RSA: primo p = 31, primo q = 29 y clave pública e = 389.

En la imagen anterior se pueden ver los primeros valores del ataque, como son el primer valor de la columna I, el primer valor de la columna J y los cifrados sucesivos. También se pueden observar los cifrados estimados ( $\approx 3\sqrt{n}$ ) y los cifrados que se han tenido que probar para obtener la solución a este ataque.

**Ataque Paradoja del Cumpleaños**

Datos Ataque

Módulo n: 899

Exponente: 389

Mensaje M: 2

Cifrados Estimados: 87

Cifrados Probados: 56

Media Cifrados/Seg: 56

Comenzar Pausar

Información Limpiar Datos

**Resultados**

Col J -->  $2^{471} \bmod 899 = 126$   
Col I -->  $2^{22} \bmod 899 = 469$   
Col J -->  $2^{472} \bmod 899 = 252$   
Col I -->  $2^{23} \bmod 899 = 39$   
Col J -->  $2^{473} \bmod 899 = 504$   
Col I -->  $2^{24} \bmod 899 = 78$   
Col J -->  $2^{474} \bmod 899 = 109$   
Col I -->  $2^{25} \bmod 899 = 156$   
Col J -->  $2^{475} \bmod 899 = 218$   
Col I -->  $2^{26} \bmod 899 = 312$   
Col J -->  $2^{476} \bmod 899 = 436$   
Col I -->  $2^{27} \bmod 899 = 624$   
Col J -->  $2^{477} \bmod 899 = 872$   
Col I -->  $2^{28} \bmod 899 = 349$   
Col J -->  $2^{478} \bmod 899 = 845$   
Col I -->  $2^{29} \bmod 899 = 698$   
Col J -->  $2^{479} \bmod 899 = 791$   
Col I -->  $2^{30} \bmod 899 = 497$   
Col J -->  $2^{480} \bmod 899 = 683$

Se ha detectado una colisión de la posición 30 de la Columna I con la primera posición de la Columna J.

Cálculo de la Clave Privada, Clave Privada Pareja o Falso Positivo:  
--> Se calcula  $w = (i - j) / \text{mcd}(e, |i - j|)$ .  
Siendo  $i = 30$ ,  $j = 450$  y la clave pública = 389.  
--> Resultado  $w = 420$   
--> Se calcula  $t = \text{inv}(e, w)$ .  
--> Resultado  $t = 149$

El resultado t obtenido es la Clave Privada o una Clave Privada Pareja.

Clave Privada: 149

Tiempo Total: 0,018 Seg

Como se muestra en esta otra imagen, en la vuelta 30 se encuentra el valor 683 que corresponde con el primer cifrado de la columna J. Ahora se va a comprobar que el cálculo de la clave obtenida sea el correcto.

- $w = (i - j) / \text{mcd}(e, |i - j|)$  tal que  $i=30$ ,  $j=450$  y  $e=389$ .

$$w = (30 - 450) / \text{mcd}(389, |30 - 450|) =$$

$$= -420 / \text{mcd}(389, 420) = -420 / 1 = -420$$

- $t = \text{inv}(e, w) = \text{inv}(389, 420) = 149$
- Se comprueba que  $t$  es una clave privada o una clave privada pareja cifrando un número distinto del mensaje  $M$  y viendo que se puede descifrar con  $t$ .

Cifrado de 5:  $5^{389} \bmod 889 = 689$

Descifrado de 689:  $689^{149} \bmod 889 = 5$

Queda comprobado que el valor  $t$  obtenido es una clave privada pareja o la propia clave privada. Si este valor, se compara con la clave que se había generado, se observa que el resultado obtenido es la clave privada.

Clave RSA

Componentes privados RSA

Numero primo p

31

▼

dec

5

Bits

Numero primo q

29

▼

dec

5

Bits

$\Phi(n)$

840

dec

10

Bits

Clave privada d

149

dec

8

Bits

Componentes públicos RSA

Módulo n

899

dec

10

Bits

Clave pública e

389

dec

9

Bits

Segundo caso de ataque. A continuación se procede a comprobar que el resultado es correcto. Datos del ataque: Mensaje  $M = 2$ . Datos de la Clave RSA: primo  $p = 31$ , primo  $q = 19$  y clave pública  $e = 7$ .

genRSA - Ataque por la Paradoja del Cumpleaños

### Ataque Paradoja del Cumpleaños

Datos Ataque

Módulo n: 589

Exponente: 7

Mensaje M: 2

Cifrados Estimados: 72

Cifrados Probados: 46

Media Cifrados/Seg: 46

Comenzar Pausar

Información Limpiar Datos

### Resultados

Col J -->  $2^{311} \bmod 589 = 374$   
 Col I -->  $2^{17} \bmod 589 = 314$   
 Col J -->  $2^{312} \bmod 589 = 159$   
 Col I -->  $2^{18} \bmod 589 = 39$   
 Col J -->  $2^{313} \bmod 589 = 318$   
 Col I -->  $2^{19} \bmod 589 = 78$   
 Col J -->  $2^{314} \bmod 589 = 47$   
 Col I -->  $2^{20} \bmod 589 = 156$   
 Col J -->  $2^{315} \bmod 589 = 94$   
 Col I -->  $2^{21} \bmod 589 = 312$   
 Col J -->  $2^{316} \bmod 589 = 188$   
 Col I -->  $2^{22} \bmod 589 = 35$   
 Col J -->  $2^{317} \bmod 589 = 376$   
 Col I -->  $2^{23} \bmod 589 = 70$   
 Col J -->  $2^{318} \bmod 589 = 163$   
 Col I -->  $2^{24} \bmod 589 = 140$   
 Col J -->  $2^{319} \bmod 589 = 326$   
 Col I -->  $2^{25} \bmod 589 = 280$   
 Col J -->  $2^{320} \bmod 589 = 63$

Se ha detectado una colisión de la posición 25 de la Columna I con la primera posición de la Columna J.

Cálculo de la Clave Privada, Clave Privada Pareja o Falso Positivo:  
 --> Se calcula  $w = (i - j) / \text{mcd}(e, |i - j|)$ .  
 Siendo  $i = 25$ ,  $j = 295$  y la clave pública = 7.  
 --> Resultado  $w = 270$   
 --> Se calcula  $t = \text{inv}(e, w)$ .  
 --> Resultado  $t = 193$

El resultado t obtenido es la Clave Privada o una Clave Privada Pareja.

Clave Privada: 193

Tiempo Total: 0,002 Seg

Como se observa en la imagen superior, en la vuelta 25 se encuentra el valor 280 que corresponde con el cifrado de la columna J. Ahora se va a comprobar que el cálculo de la clave obtenida sea el correcto.

- $w = (i - j) / \text{mcd}(e, |i - j|)$  tal que  $i=25$ ,  $j=295$  y  $e=7$ .

$$w = (25 - 295) / \text{mcd}(7, |25 - 295|) =$$

$$= -270 / \text{mcd}(7, 270) = -270 / 1 = -270$$

- $t = \text{inv}(e, w) = \text{inv}(7, 270) = 193$
- Se comprueba que t es una clave privada o una clave privada pareja cifrando un número distinto del mensaje M y viendo que se puede descifrar con t.

$$\text{Cifrado de 10: } 10^7 \bmod 589 = 547$$

$$\text{Descifrado de 547: } 547^{193} \bmod 589 = 10$$

Queda comprobado que el valor  $t$  obtenido es una clave privada pareja o la propia clave privada. Si este valor, se compara con la clave que se había generado, se observa que el resultado obtenido es una clave privada pareja.

Clave RSA

Componentes privados RSA

Numero primo $p$	31	▼	dec	5	Bits
Numero primo $q$	19	▼	dec	5	Bits
$\Phi(n)$	540		dec	10	Bits
Clave privada $d$	463		dec	9	Bits

Componentes públicos RSA

Módulo $n$	589		dec	10	Bits
Clave pública $e$	7		dec	3	Bits

Claves Privadas Parejas - CPP

Cantidad de Claves

6

13 -> 4 bits  
103 -> 7 bits  
193 -> 8 bits  
283 -> 9 bits  
373 -> 9 bits  
553 -> 10 bits

Tercer caso de ataque. Finalmente, el tercer caso de ataque se da cuando el resultado obtenido es un falso positivo. Un ejemplo se da con la clave formada por los componentes: primo  $p=3.889$ , primo  $q=769$  y clave pública=979.757.

Con dichos datos de clave y el mensaje  $M = 2$ , no se obtiene una clave privada ni una clave privada pareja (ver imagen siguiente). El resultado obtenido solo servirá para descifrar el propio mensaje.

Se va a comprobar que el resultado es un falso positivo:

- Se elige un valor dentro del cuerpo de cifra al azar: 318.239
- Se cifra con la clave pública.

$$318.239^{979.757} \bmod 2.990.641 = 2.670.055$$

- Se comprueba que el resultado t obtenido no descifra este mensaje.

$$2.670.055^{1.413.413} \bmod 2.990.641 = 2.165.631 \text{ ERROR!}$$

- Se comprueba que la clave privada sí lo descifra.

$$2.670.055^{2.536.613} \bmod 2.990.641 = 318.239$$

## Cifrado, Descifrado, Firma y Validación

Todas estas operaciones tienen en común que se pueden realizar sobre datos que sean solo numéricos (decimales o hexadecimales) o bien datos que contengan texto (caracteres ASCII).

En el caso de solo querer utilizar estas operaciones con números, no se requiere longitud mínima en la clave generada. En el caso de querer realizar alguna de las operaciones en las que los datos contengan texto, será necesario haber generado una clave de longitud mayor o igual a 12 bits.

Más aspectos a tener en cuenta son los siguientes:

- Cifra y Firma: Si los datos introducidos no son texto, se introducirá un número por línea. Si el número introducido es mayor que el módulo, se le aplicará la operación módulo para reducirlo. Si los datos introducidos son texto, se codificarán en ASCII y se cifrarán/firmarán en bloques de bytes. Los bloques serán de tamaño máximo igual al número de bytes del módulo menos uno.
- Descifrado y Validación: Si los datos introducidos no son texto, se introducirá un número por línea. Si el número introducido es mayor que el módulo, se le aplicará la operación módulo para reducirlo. Si los datos introducidos son texto, se descifrarán/validarán y se codificarán en ASCII. Es el usuario el que debe asegurarse que los datos introducidos tienen caracteres ASCII legibles.

## **Cifra**

El cifrado de información se realiza empleando la clave pública del receptor y su módulo o cuerpo de cifra.

Se van a realizar cuatro pruebas del cifrado de información. Todas ellas con la misma clave: primo  $p = 79$ , primo  $q = 103$  y clave pública  $e = 4.333$ .

- Primera prueba: Cifrar dos números inferiores al módulo poniendo cada uno en una línea. Estos números son: 4.563 y 1.223.

genRSA - Cifrado y Descifrado

### Cifrado - Clave Pública Destinatario

Datos para el Cifrado

Módulo: 8.137

Clave Pública: 4.333

Datos Originales: 4.563, 1.223

☐ Tienen texto los Datos Originales

Resultados del Cifrado

Datos Cifrados: 3.505, 3.644

Cifrar Datos    Información    Limpiar Datos

Comprobación:

$$4.563^{4.333} \bmod 8.137 = 3.505$$

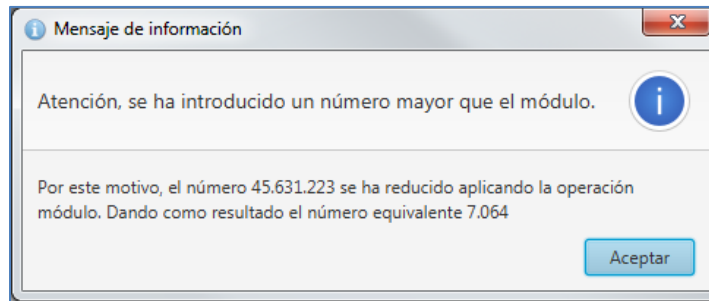
$$1.223^{4.333} \bmod 8.137 = 3.644$$

Comparando estas operaciones con los de la imagen se comprueba que los resultados son correctos.

- Segunda prueba: Se intenta cifrar un número mayor que el módulo. Concretamente, es el número formado por los dos números en claro que se han cifrado en la primera prueba: 45.631.223.

Al ejecutar la operación de cifra se muestra un mensaje por pantalla indicando que el número introducido es mayor que el módulo y se va a reducir antes de cifrarlo.





- Tercera prueba: cifrar dos caracteres introduciendo cada uno en una línea.

genRSA - Cifrado y Descifrado

### Cifrado - Clave Pública Destinatario

Datos para el Cifrado

Módulo: 8.137

Clave Pública: 4.333

Datos Originales: S  
I

☒ Tienen texto los Datos Originales

Resultados del Cifrado

Datos Cifrados: 7.682  
7.458

Cifrar Datos   Información   Limpiar Datos

A continuación se procede a comprobar que los datos son correctos. Para ello, primero se transformarán a ASCII los caracteres: S = 83 e I = 73.

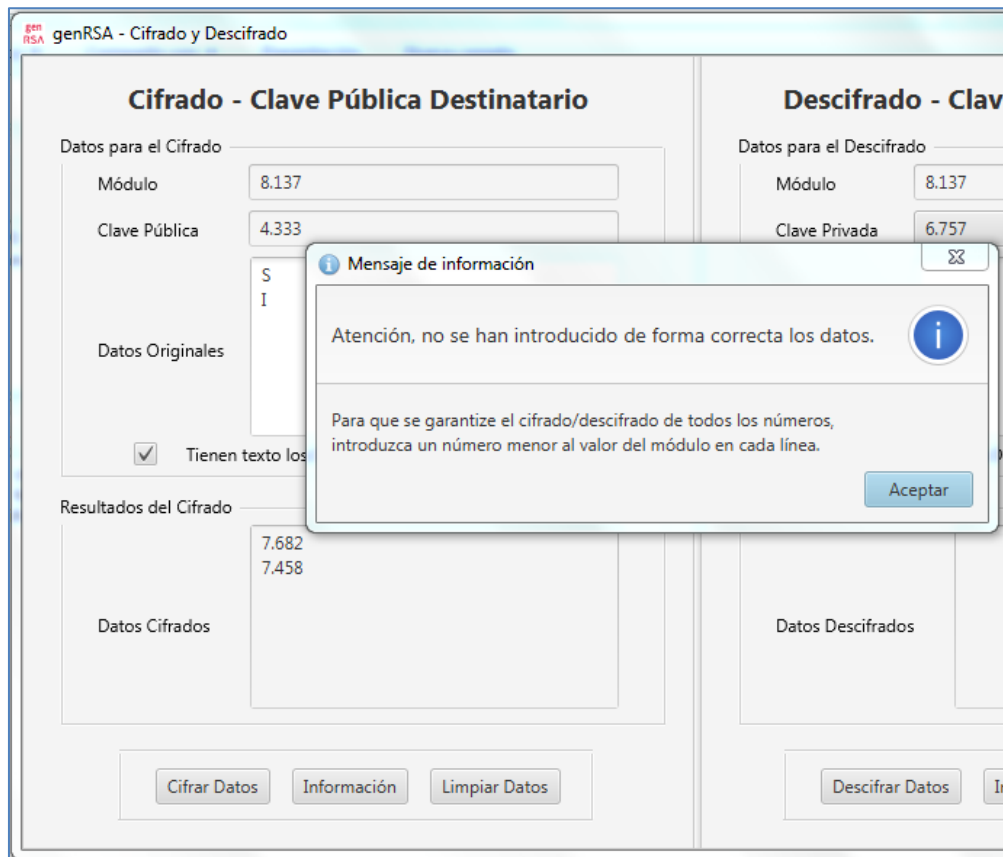
$$83^{4.333} \bmod 8.137 = 7.682$$

$$73^{4.333} \bmod 8.137 = 7.458$$

Comparando estas operaciones con los de la imagen se comprueba que los resultados son correctos.

- Cuarta prueba: Se intenta cifrar los dos caracteres del apartado anterior, pero esta vez introduciéndolos en la misma línea.

Al intentar ejecutar la operación de cifra, se muestra un mensaje por pantalla indicando que no se han introducido de forma correcta los datos. Una vez mostrado el mensaje se puede ver cómo los datos se han fragmentado quedando cada carácter en una línea. Esto es debido a que cada carácter ocupa 8 bits y el módulo es de tan solo 13 bits.



## Descifrado

El descifrado de información se realiza empleando la clave privada del receptor y su módulo o cuerpo de cifra. En esta ocasión se usará la misma clave que en la cifra.

En la imagen siguiente se puede comprobar que los resultados de las pruebas primera y tercera del apartado de cifra son correctos. Para ello se introducen los datos cifrados de ambas pruebas en el apartado de descifrado de genRSA v2.1. Se comprueba que el resultado obtenido del descifrado es igual al mensaje original antes de realizar el cifrado.

The image displays two side-by-side screenshots of the 'Descifrado - Clave Privada Destinatario' window in the genRSA v2.1 application. Both windows share the same input fields: 'Módulo' (8.137) and 'Clave Privada' (6.757). The left window, labeled 'Prueba 1', shows 'Datos Cifrados' as 3.505 and 3.644, and 'Resultados del Descifrado' as 4.563 and 1.223. The right window, labeled 'Prueba 3', shows 'Datos Cifrados' as 7.682 and 7.458, and 'Resultados del Descifrado' as 'S' and 'I'. Both windows have a checkbox 'Tienen texto los Datos Originales' which is unchecked in the left and checked in the right. At the bottom of each window are three buttons: 'Descifrar Datos', 'Información', and 'Limpiar Datos'.

Prueba	Módulo	Clave Privada	Datos Cifrados	Datos Descifrados
Prueba 1	8.137	6.757	3.505 3.644	4.563 1.223
Prueba 3	8.137	6.757	7.682 7.458	S I

## Firma

La operación de firma se realiza empleando la clave privada del emisor (o cualquiera de las claves privadas parejas) y su módulo.

Dado que la cifra y la firma emplean los mismos métodos, en este caso se van a realizar pruebas con números hexadecimales. Para ello lo primero es generar una clave hexadecimal: clave pública  $e = 0x\ 6BDC5$ , primo  $p = 0x\ 481$  y, por último, primo  $q = 0x\ 1AF$ .

- Primera prueba: firmar dos valores hexadecimales inferiores al módulo introduciendo cada uno en una línea. Estos valores son:  $0x\ A38B$  y  $0x\ 23$ .

genRSA - Firma y Validación

### Firma - Clave Privada Emisor

Datos para la Firma

Clave Privada: 78A0D

Módulo: 7952F

Datos Originales: A38B  
23

☐ Tienen texto los Datos Originales

Resultados de la Firma

Datos Firmados: 61007  
1A66C

Firmar Datos Información Limpiar Datos

$$\begin{aligned} A38B^{78A0D} \bmod 7952F &= 61007 \\ 23^{78A0D} \bmod 7952F &= 1A66C \end{aligned}$$

Tras realizar los cálculos, queda demostrado que el resultado mostrado por la aplicación es correcto.

- Segunda prueba: firmar el mensaje HOLA introduciendo los datos en una sola línea. Esta vez la firma se realizará con la clave privada pareja.

El mensaje HOLA al codificarlo en ASCII se obtiene 4 bytes. Pero dado que la clave es de solo 19 bits, no es posible firmar este mensaje sin dividirlo. Con 19 bits, la firma se va a realizar en bloques de 2 Bytes. De este modo el mensaje a cifrar quedará así: "HO" "LA".

Antes de mostrar la imagen en la que se realiza la firma con la aplicación, se van a realizar los cálculos para obtener los resultados.

**H** = 0x 48, **O** = 0x 4F, **L** = 0x 4C, **A** = 0x 41

"HO" cifrado =  $686F^{3C28D} \bmod 7952F = 32FC3$

"LA" cifrado =  $4C41^{3C28D} \bmod 7952F = 551A4$

La imagen siguiente verifica que los resultados son correctos.

The screenshot shows the 'genRSA - Firma y Validación' application window. The title bar includes the 'genRSA' logo and the text 'genRSA - Firma y Validación'. The main content area is titled 'Firma - Clave Privada Emisor'. It is divided into two sections: 'Datos para la Firma' and 'Resultados de la Firma'. In the 'Datos para la Firma' section, there are three input fields: 'Clave Privada' with a dropdown menu showing '3C28D -> 18 bits', 'Módulo' with the value '7952F', and 'Datos Originales' with the text 'HO' and 'LA' on separate lines. Below these fields is a checkbox labeled 'Tienen texto los Datos Originales' which is checked. The 'Resultados de la Firma' section contains a text box labeled 'Datos Firmados' showing the results '32FC3' and '551A4' on separate lines. At the bottom of the window, there are three buttons: 'Firmar Datos', 'Información', and 'Limpiar Datos'.

## Validación

La operación de validación se realiza empleando la clave pública del emisor y su módulo. En esta ocasión se usará la misma clave que en la firma.

Con esta operación se puede validar la firma realizada en las pruebas del apartado de firma. Para ello se introducen los datos firmados de ambas pruebas en la caja "Datos Firmados" de la aplicación genRSA v2.1. Una vez obtenido el resultado, se valida que es igual a los datos originales introducidos antes de realizar la firma.

The image displays two side-by-side screenshots of the 'Validar firma - Clave Pública Emisor' application window. Both windows have a title bar with standard Windows window controls. The left window is titled 'Validar firma - Clave Pública Emisor' and contains the following elements:

- Datos para validar la Firma:**
  - Clave Pública: 68DC5
  - Módulo: 7952F
  - Datos Firmados: 61007 1A66C
  - ☐ Tienen texto los Datos Originales
- Resultados de la validación:**
  - Datos Validados: A38B 23
  - Validation Result: **Validación Prueba 1**
- Buttons:** Validar Firma, Información, Limpiar Datos

The right window is also titled 'Validar firma - Clave Pública Emisor' and contains the following elements:

- Datos para validar la Firma:**
  - Clave Pública: 68DC5
  - Módulo: 7952F
  - Datos Firmados: 32FC3 551A4
  - ☒ Tienen texto los Datos Originales
- Resultados de la validación:**
  - Datos Validados: HO LA
  - Validation Result: **Validación Prueba 2**
- Buttons:** Validar Firma, Información, Limpiar Datos