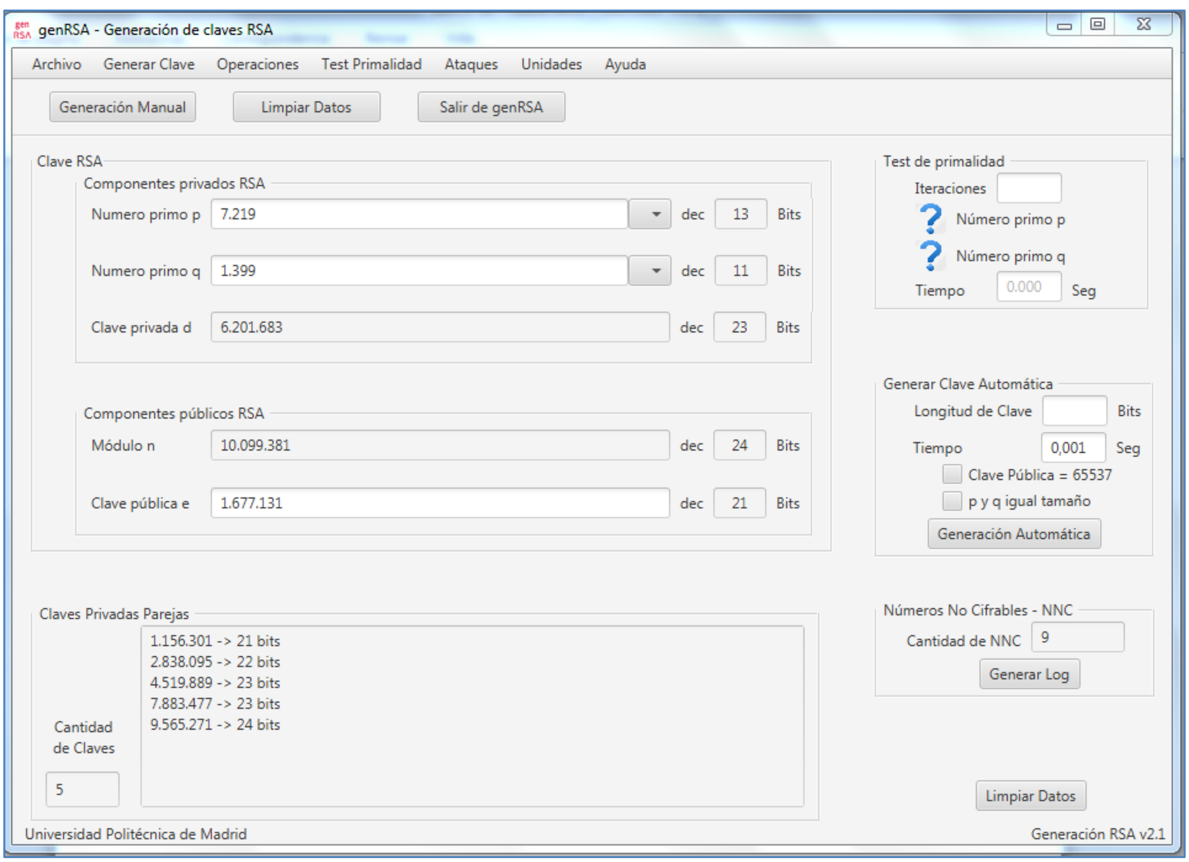
Banco de Pruebas

Todas las pruebas aquí indicadas se han realizado con un Dell 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 nos permite generar claves decimales o hexadecimales sin importar la longitud en bits de la misma. A continuación se puede ver el ejemplo de una clave decimal de 25 bits:

- Primo p: 7.219
- Primo q: 1.399
- Clave pública e: 1.677.131



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

- Módulo n = p x q = 10.099.381

  n = 7.219 x 1.399 = 10.099.381

- Máximo común divisor (clave pública e, φ(n)) = 1
  φ(n) = (p-1) x (q-1) = 7.218 x 1.398 = 10.090.764
  mcd (1.677.131, 10.090.764) = 1

- φ(n) mayor que la clave pública y la clave pública mayor que 1.
  10.090.764 > 1.677.131 > 1

- d = inv [e, φ(n)] = 6.201.683

  d = inc [1.677.131, 10.090.764] = 6.201.683

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

- <u>Claves privadas parejas</u>: la clave privada d es única en φ(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 $d_γ$.
  $d_γ$ = inv(e, γ), siendo γ= mcm [(p-1),(q-1)]
  γ= mcm (7.218, 1.398) =1.681.794
  $d_γ$ = inv(1.677.131, 1.681.794) =1.156.301

  A continuación se calcula la cantidad de claves privadas parejas.
  λ = parte entera de ((n - $d_γ$)/γ)
  λ = ((10.099.381 - 1.156.301)/ 1.681.794) =
  = (8.943.080 / 1.681.794) = 5

  Una vez comprobado que la cantidad de claves y la primera coinciden se comprueba el resto de claves, tal que d_i = $d_γ$ + i x γ, donde i = 0, 1, … , λ

Clave 2 = d_2 = 1.156.301 + 2 x 1.681.794 = 2.838.095

Clave 3 = d_3 = 1.156.301 + 4 x 1.681.794 = 4.519.889

Clave Privada = 1.156.301 + 5 x 1.681.794 = 6.201.683

Clave 5 = d_5 = 1.156.301 + 5 x 1.681.794 = 7.883.477

Clave 6 = d_6 = 1.156.301 + 6 x 1.681.794 = 9.565.271

- 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 σ_n = [1 + mcd (e-1, p-1)] x [1 + mcd (e-1, q-1)].

  σ_n = [1 + mcd (1.677.130, 7.218)] x [1 + mcd (1.677.130, 1.398)] =
  = [1 + 2] x [1 + 2] = 9

  Comprobado que la cantidad es correcta se genera el fichero de Log. Este nos indica que los NNC son: 0, 1, 2.569.963, 2.569.964, 4.959.454, 5.139.927, 7.529.417, 7.529.418, 10.099.380. Se pueden 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 tests 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.

- <u>Número 1, 20 bits</u>: 963.097
- <u>Número 2, 40 bits</u>: 802.531.780.577
- <u>Número 3, 128 bits</u>: 206.022.852.512.717.161.155.800.602.892.466.466.027
- <u>Número 4, 512 bits</u>: 13.302.234.470.614.217.204.793.688.246.321.864.759.812.857.969.866.885.120.212.272.507.338.410.462.753.235.484.338.062.389.026.258.357.078.305.228.326.176.576.532.983.080.807.609.059.660.259.192.952.449.763

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: 11.123.175.549.804.993.308.746.373.791.189.175.807.007.447.831.932.598.943.746.092.445.552.757.926.599.044.109.456.406.791.612.934.381.949.242.184.333.527.585.839.484.989.304.177.336.008.496.437.129.967.769

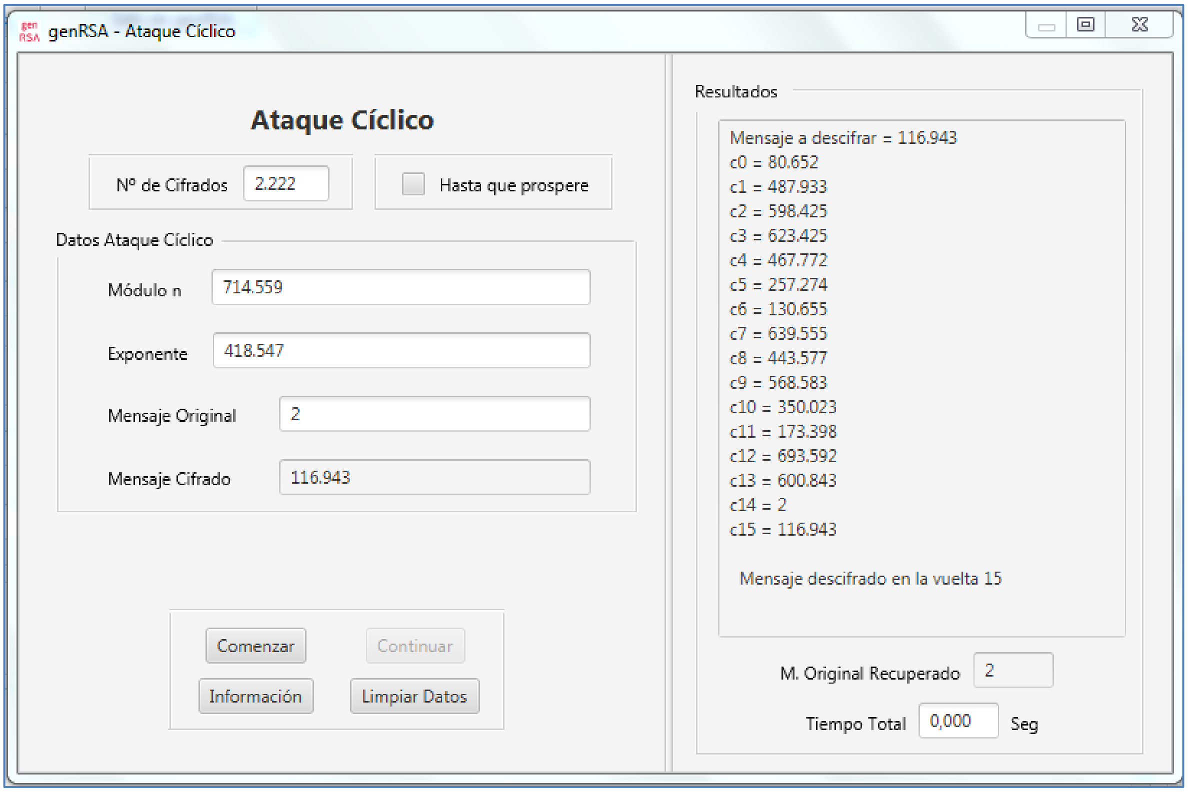
El resultado de ambas pruebas a resultado ha sido satisfactorio. Con lo cual podemos afirmar que los Tests de Primalidad funcionan correctamente.

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 con la clave pública e y 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 original = 2 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.



El mensaje original cifrado da como resultado 116.943. Será este valor el que se ataque:

- $c0 = 116.943^{418.547}$ mod 714.559 = 80.652
- $c1 = 80.652^{418.547}$ mod 714.559 = 487.933
- $c2 = 487.933^{418.547}$ mod 714.559 = 598.425
- $c3 = 598.425^{418.547}$ mod 714.559 = 623.425
- $c4 = 623.425^{418.547}$ mod 714.559 = 467.772
- $c5 = 467.772^{418.547}$ mod 714.559 = 257.274
- $c6 = 257.274^{418.547}$ mod 714.559 = 130.655
- $c7 = 130.655^{418.547}$ mod 714.559 = 639.555
- $c8 = 639.555^{418.547}$ mod 714.559 = 443.577
- $c9 = 443.577^{418.547}$ mod 714.559 = 568.583
- $c10 = 568.583^{418.547}$ mod 714.559 = 350.023
- $c11 = 350.023^{418.547}$ mod 714.559 = 173.398
- $c12 = 173.398^{418.547}$ mod 714.559 = 693.592
- $c13 = 693.592^{418.547}$ mod 714.559 = 600.843
- $c14 = 600.843^{418.547}$ mod 714.559 = **2**

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

Por último se realizarán diversos ataques al mensaje M = 123, 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	481.477	28.379	0,934	-
30	858.549.683	885.299	1,466	603.887
35	13.891.030.211	1.204.007	1,939	620.942
40	522.968.699.299	9.152.219	13,113	697.950
50	653.140.619.338.489	49.319.675	69,326	711.416

Para poder comprobar que el número de vuelta 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 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.

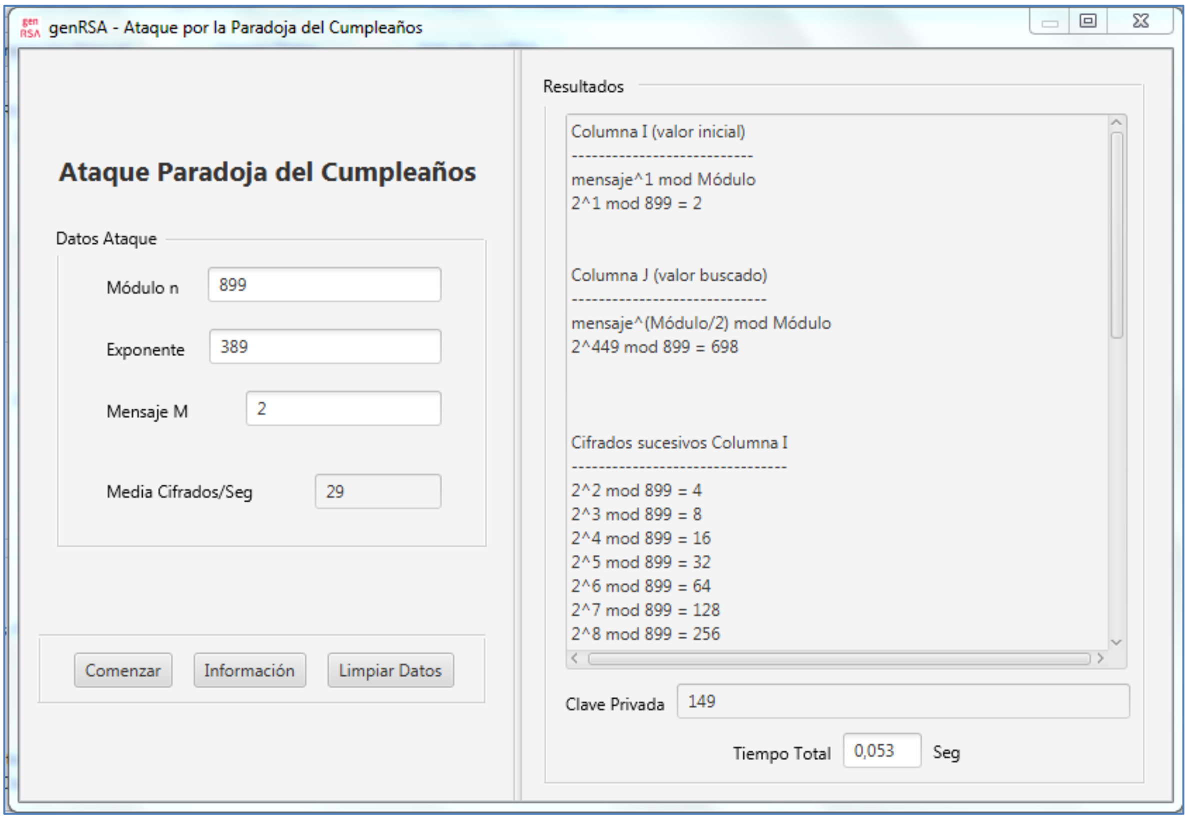
GenRSA v2.1 no cambia de algoritmo conforme se modifican las unidades de la clave (decimal o hexadecimal). Esta premisa es válida para todas las funcionalidades de la aplicación.

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

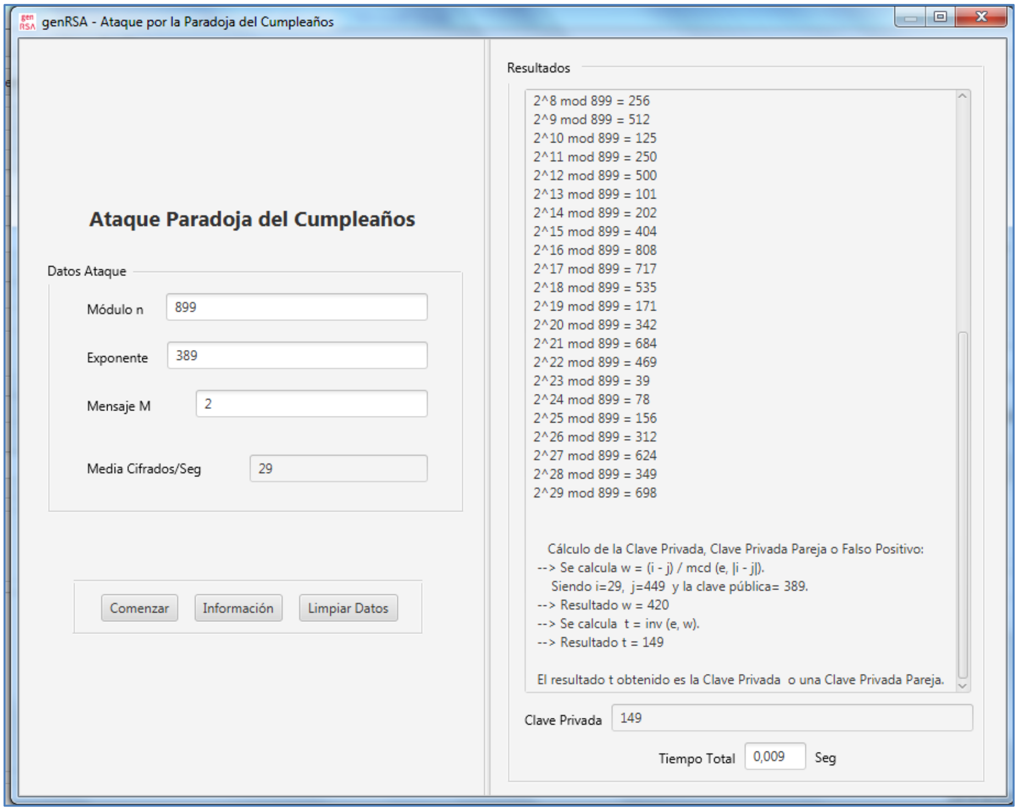
Ataque Paradoja del Cumpleaños

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

Se procede a comprobar que el resultado del siguiente ataque 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 esta primera imagen podemos ver los primeros valores del ataque como son la columna I y el primer valor de la columna J. A continuación se va a proceder a realizar cifrados sucesivos a la columna I hasta encontrar un valor que sea igual al cifrado de la columna J.



Como vemos en esta otra imagen, en la vuelta 29 se encuentra el valor 698 que corresponde con el cifrado de la columna J. Ahora se va a 42comprobar que el cálculo de la clave obtenida sea el correcto.

- w =(i – j) / mcd (e, |i – j|) tal que i=29, j=449 y e=389.

  w = (29 - 449) / mcd (389, |29 – 449|) =

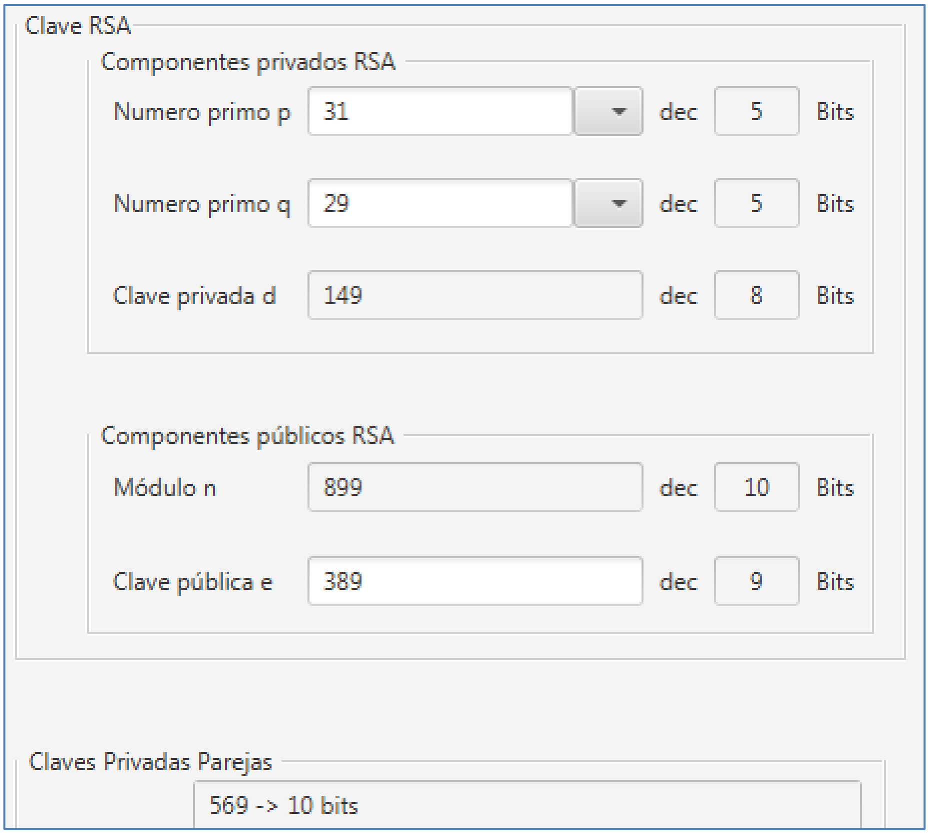
  = -420 / mcd(389, 420) = -420 / 1 = -420

- t = inv(e, w) = 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} mod 889 = 689

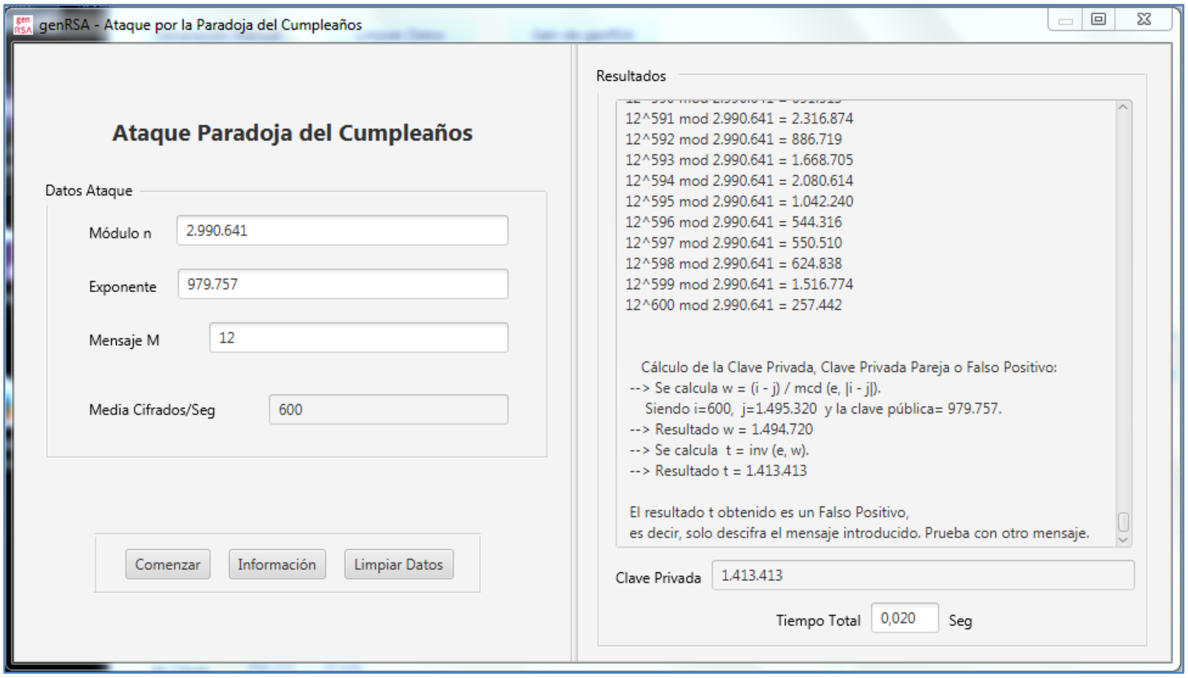
  Descifrado de 689: 689^{149} mod 889 = 5

Queda comprobado que el valor t obtenido es una clave privada pareja o la propia clave privada. Si lo comparamos con la clave que se había generado, podemos observar que el resultado obtenido es la clave privada.



Por otro lado, al realizar el ataque por la paradoja del cumpleaños puede darse el caso de obtener un falso positivo. Este es el caso de la clave formada por los siguientes componentes: primo p=3.889, primo q=769 y clave pública=979.757.

Con dichos datos de clave y el mensaje M = 12, 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.

  $318.239^{979.757}$ mod 2.990.641 = 2.670.055

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

  $2.670.055^{1.413.413}$ mod 2.990.641 = 2.165.631 ERROR!

- Se comprueba que la clave privada si lo descifra.

  $2.670.055^{2.536.613}$ mod 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 dividirá en números de menor o igual valor que el módulo. Si los datos introducidos son texto, se codificarán en ASCII y se cifrarán/firmaran 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 dividirá en números de menor o igual valor que el módulo. 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.

Comprobación:

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

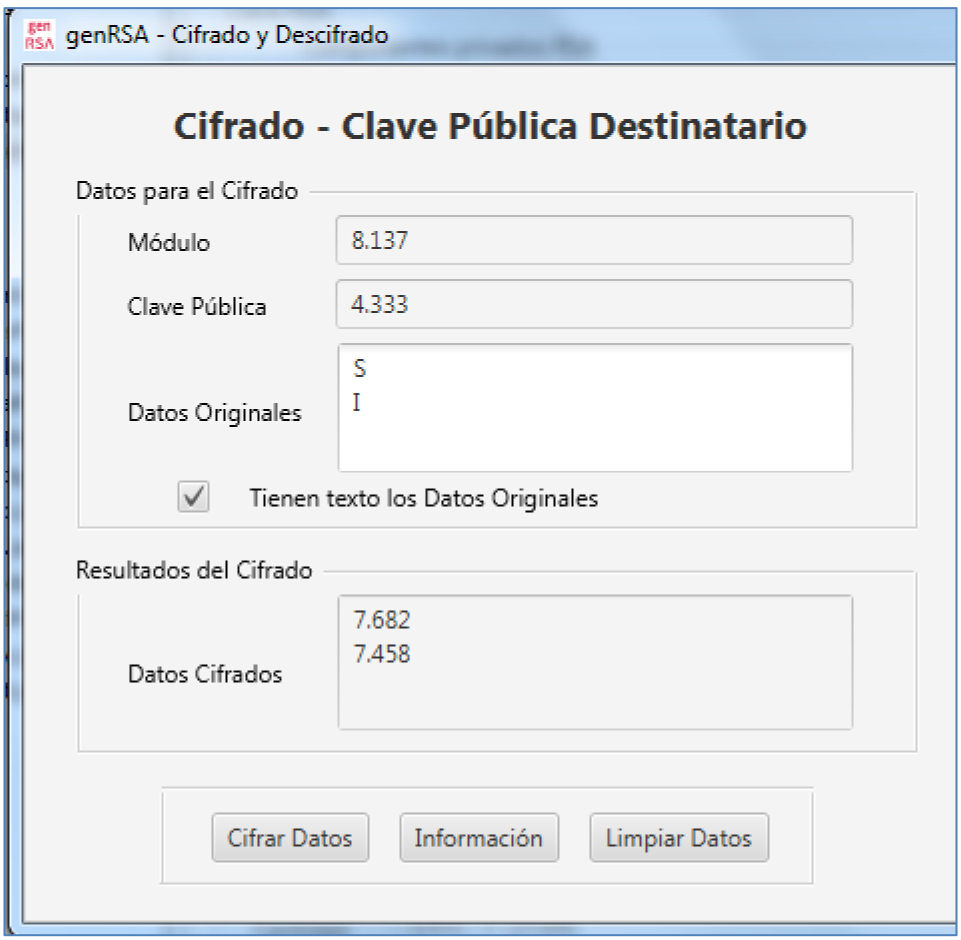
$1.223^{4.333} \bmod 8.137 = 3.644$

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

- <u>Segunda prueba</u>: 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 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 se pulsa aceptar se puede ver como los datos se han fragmentado formando números inferiores al módulo pero lo más cercanos posible.

- <u>Tercera prueba</u>: cifrar dos caracteres introduciendo cada uno en una línea.



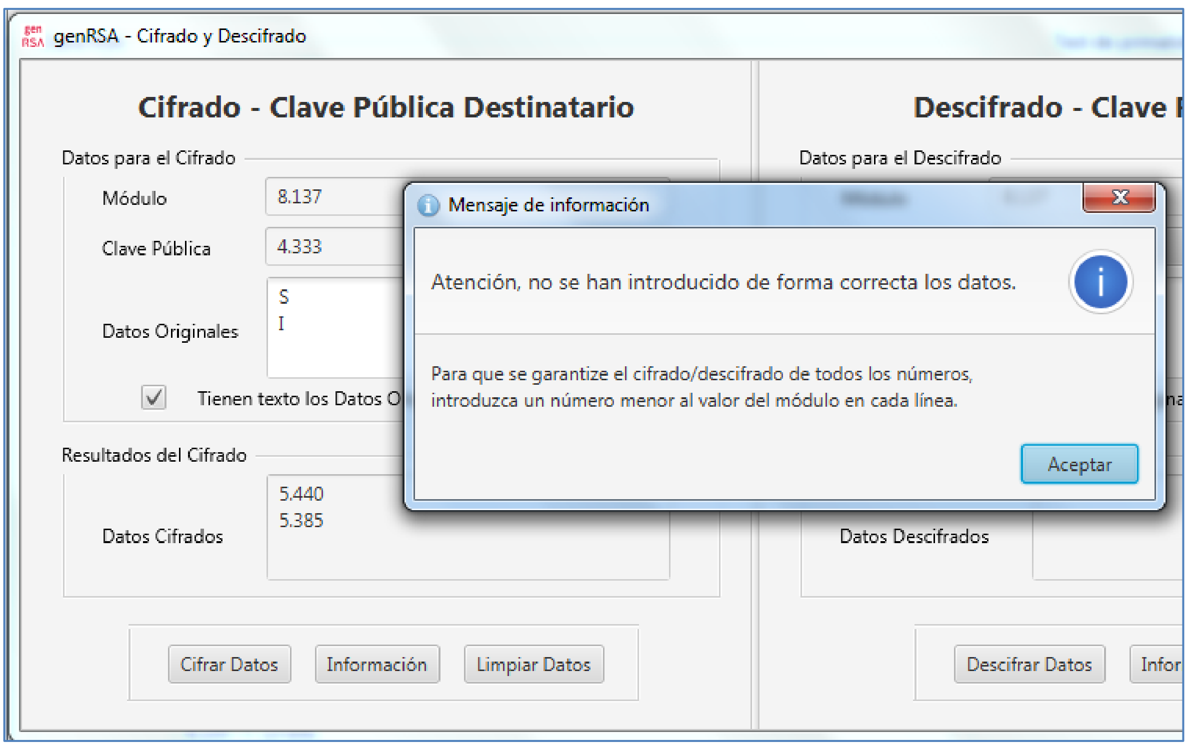
A continuación se procede a comprobar que los datos son correctos. Para ello primero se transformarán a ASCII los caracteres: S = 83 e I =73.

$83^{4.333}$ mod 8.137 = 7.682
$73^{4.333}$ mod 8.137 = 7.458

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

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

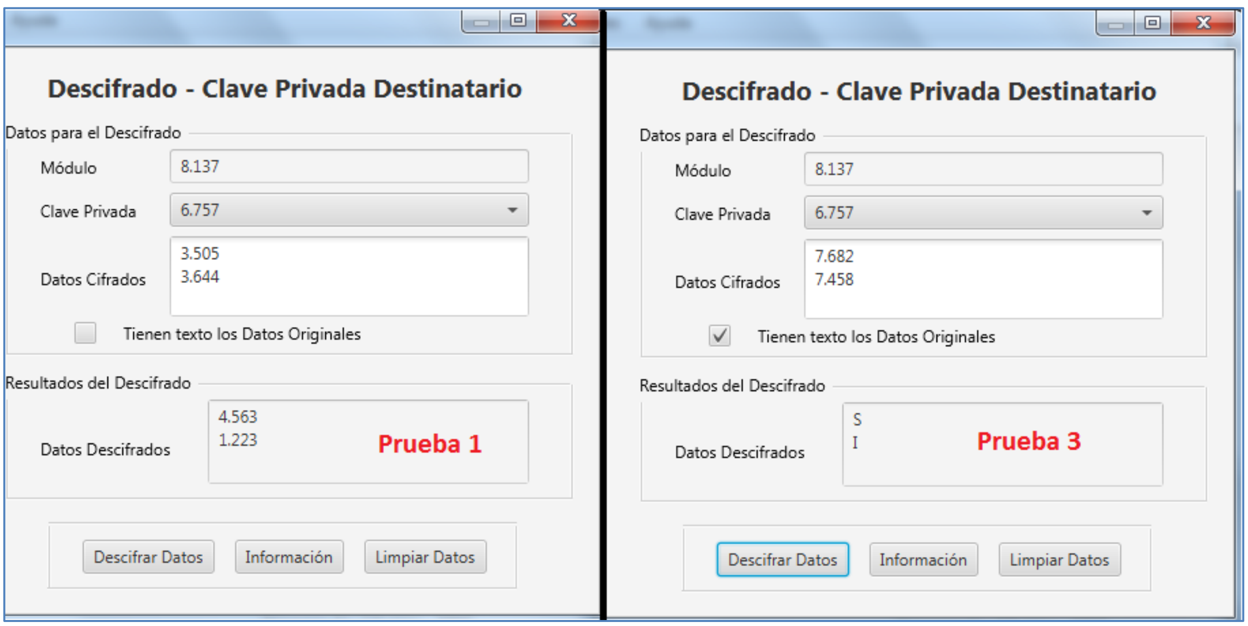
  Al intentar ejecutar la operación de cifra se muestra un mensaje por pantalla indicando que no se han introducido de forma correcta los datos. Una vez mostrado el mensaje se puede ver como los datos se han fragmentado quedando cada carácter en una línea. Esto es debido a que cada carácter ocupa 8 bits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

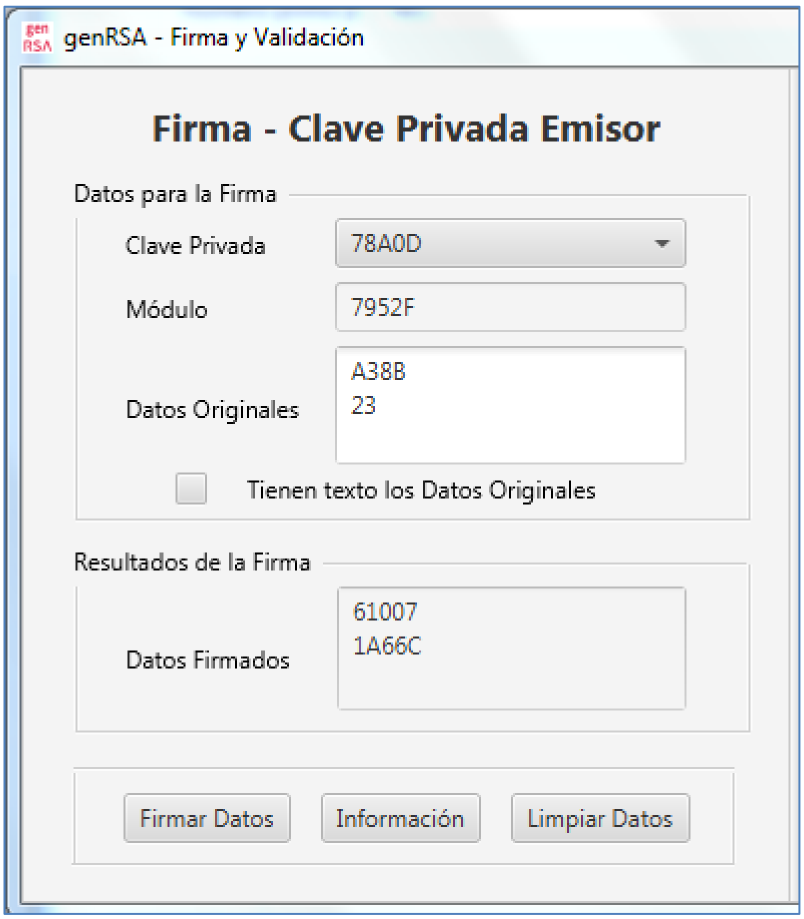


Firma

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 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.



$A38B^{78A0D}$ mod 7952F = 61007
23^{78A0D} mod 7952F = 1A66C

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.

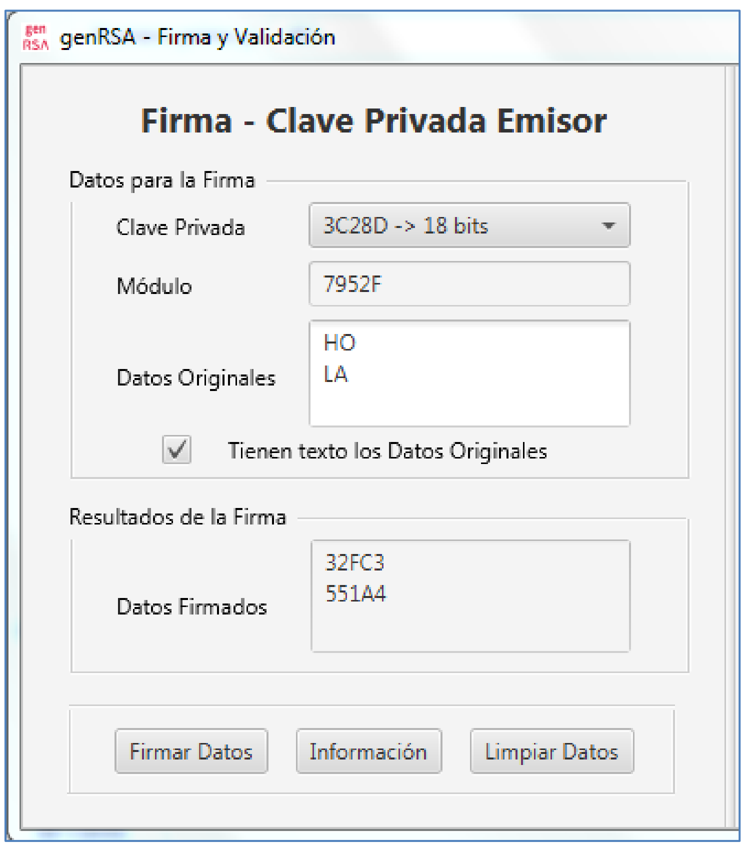
  El mensaje HOLA al codificarlo en ASCII se obtiene 4 bytes. Pero dado que la clave es de solo 19 bits, no es posible firmar este mensaje sin dividirlo. Con 19 bits, la firma se va a realizar en bloques de 2 Bytes. De este modo el mensaje, a cifrar quedará así: "HO" "LA".

  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 = $686F^{3C28D}$ mod 7952F = 32FC3

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

  La imagen siguiente verifica que los resultados son correctos.



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 podemos 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.