Proyecto CLCript
Cuadernos de Laboratorio de Criptografía. Entrega nº 4. Última actualización 06/05/19
Autor: Dr. Jorge Ramió Aguirre (@criptored)
Prácticas con el algoritmo RSA: Generación de claves RSA con genRSA v2.1

- Software genRSA v2.1: https://www.criptored.es/software/sw_m001d.htm
- Win64 OpenSSL: https://slproweb.com/products/Win32OpenSSL.html
- Lectura de interés MOOC Crypt4you: https://www.criptored.es/crypt4you/temas/RSA/leccion1/leccion01.html
- Lectura de interés: https://crypto.stackexchange.com/questions/33523/how-common-are-weak-rsa-keys?rq=1

Objetivos:

1. Comprobar con el software de laboratorio genRSA v2.1 cómo se genera una clave RSA.
2. Generar claves RSA en formato decimal y hexadecimal, de diferentes tamaños.
3. Observar los valores típicos de los parámetros en claves RSA reales.
4. Comprobar la velocidad con la que OpenSSL genera claves RSA con números muy grandes.

I. Generación Manual de claves RSA

Ejercicio 1)

1.1. Abre el programa genRSA v2.1 e introduce estos valores p = 23, q = 31, e = 7. Hecho esto, haz clic en Generación Manual. Observa la clave generada de 10 bits.

1.2. Cambia el valor de clave pública e por 3 y 5 e intenta generar la clave.

1.3. Limpia los datos y genera ahora también manualmente una clave decimal de 1.024 bits con los valores que se indican y guarda la clave con el nombre claveRSA1024.

p =
9.440.469.914.135.934.299.457.675.921.772.190.556.429.153.648.419.122.174.874.593.399.042.603.083.702.603.846.552.111.558.461.309.607.579.885.353.619.983.077.642.978.381.204.743.684.714.325.557.481.453.657
q =
10.438.975.648.308.876.092.455.705.738.242.513.907.974.629.456.946.735.483.937.053.392.466.809.015.976.578.419.637.376.280.051.988.143.679.917.308.773.810.078.515.426.953.565.865.379.314.380.791.393.061.019
e = 65.537

1.4. Limpia los datos y desde Unidades del Menú, cambia a hexadecimal. Introduce los valores que se indican, genera manualmente la clave y guárdala como claveRSA2048.

p =
9ECA36CD01D588CB45147764B2B7F575F4DF317526290BBABAFA8A0767475BD9E05143B1AA0E0283695EF2A03346E956EC7551FAC6ABD9838CFF31E8DEE8844C2383133F8F47A4565BFCB09DC463C12B55E343F89858FB26E8E6B6BBEF2837070B3EAB49AA1B7FD768ADB78979157DE358B0B666A68937C33929C708D347204F
q =
D57C4E13F42810CE1BF47117490E4E0B3CFA0ECD7E7FBD890F3B40C9C6AAB8989D114D9D5BBEB9729ACF73BA6D054A989DCF30A5FD5086BB45DA7A15E470505ABC008E88E4F97A8D9137023EC4DBF338237C8AC808F3FF8F8B12A143BFF02DD06812C13191A2FF225E6BB3E6E49EE9DDCC9E0FB94D336E718D027754151E58BB
e = 10001

1.5. Por último, genera manualmente una clave hexadecimal de 4.096 bits con los valores que se indican y guárdala como claveRSA4096.

p =
CCD819C7D848C42810E8751F3F3C4831F7FEE865FD9C52E452A2C1D7261A1F7725929855F5E10B41DAAE3B55E454F3FC88B86BEB0A5E8B4854AFB6FC96AE77BF4443FC518848C726128EE0F52E0237C8D12BAD8CDAEDDCC5B9A944698A7B6C02EB162922BF73C5A10CBF4E10595681D19D438E3FD113C8D9BAC8CC492C77D4436059026326FBDFDC829C687A502CD0AA20B621502A7E99EBD84FAA9D44E74776C8A3649268EC57F72198A2EC7D9EFAB72FAE92B86A4AE241F1795963FB08EDB16AE655F158BEB81EC6DCC4AA709BBF3AA03BBBF3602BAC7EC0C46F21B9C51932948D35EF2F2B03939EC7C4291DBE111FA88EDFC986A5772297C402F67800241F
q =
ACFD271D5520781588012053DCC7B910FEE2FE63095300D8FC17C67B22402025C75ED955DE68C702AD62285423BF5B40C3C5CAE7B4827483D2D72319928D11A5AFC5B9B1849108FA3323019B536FCB8BDB421B684CA081E1CEDC9864D18D3CF87A029EE3815B9D27FBC67C6124E9D149E52F49BC1C0CDF429E8CCD4E0D25F9B5B0B171E12723C932F33025FC8E6C54845800169A3F1E7A576A4A60CF63158563EF859D65B04FE382C6C12E9EF4A9D8E4482E8FBA0263E035BF95289DC9786444996E546A061728DE47D5C1E7BAB15463F100F934296505BB2A81DF4EFBAF0BB59845263DC23AE31164FC358D12D78533BF5D80D6164D17DAC7EF46BD23E43607

e = 10001

1.6. Observa que en las tres últimas claves con valores reales de 1.024 (2^{10}) bits, 2.048 (2^{11}) bits y 4.096 (2^{12}) bits, se ha usado como clave pública e el valor estándar, que es el número 4 de Fermat. Observa, además, los valores que se obtienen de clave privada d, muy cercanos a $\phi(n)$.

1.7. Basándote en el concepto de los inversos multiplicativos, explica por qué en los tres casos la clave privada d es un valor muy cercano a $\phi(n)$.

Comprueba tu trabajo:

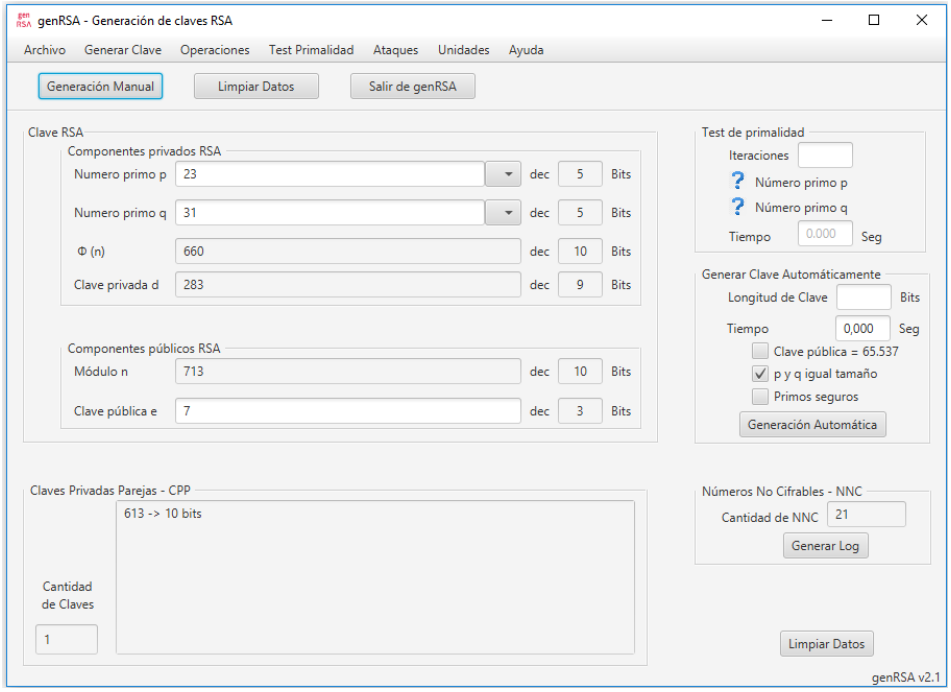

Figura 1. Clave RSA de 10 bits

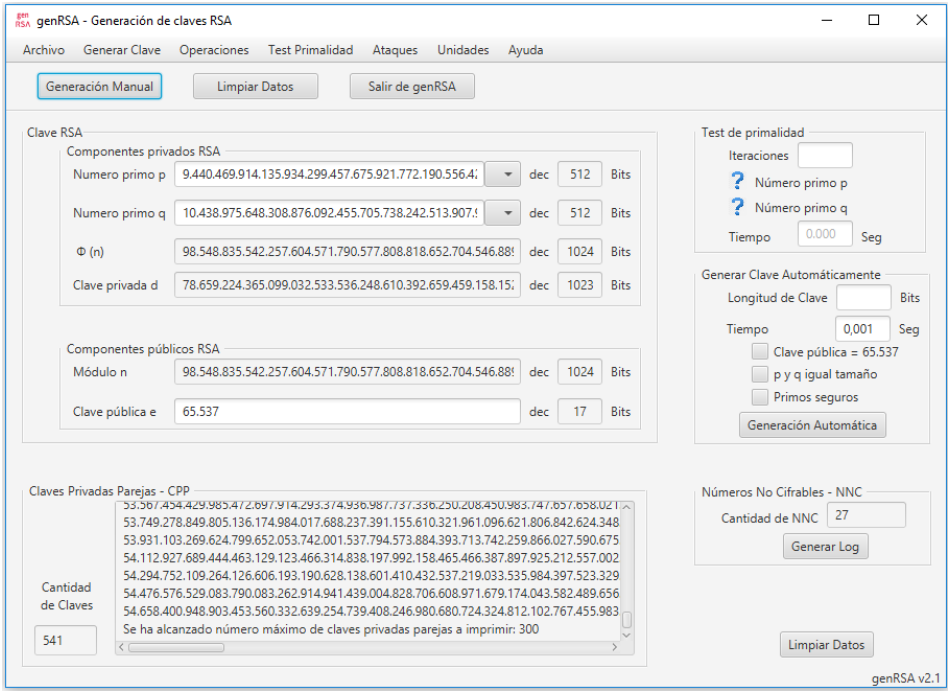

Figura 2. Clave RSA de 1.024 bits

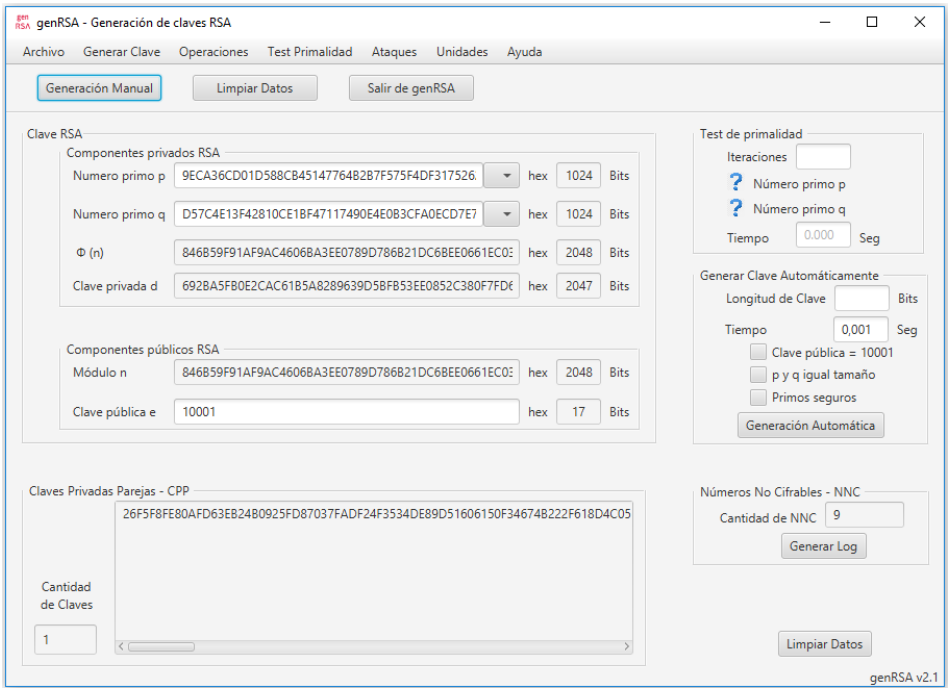

Figura 3. Clave RSA de 2.048 bits

II. Generación Automática de claves RSA

Ejercicio 2)

2.1. Limpia los datos y cambia las Unidades a Decimal. En Generar Clave Automáticamente introduce como longitud de clave 1.024 bits. Genera una decena de claves pulsando el botón Generación Automática. Apunta el tiempo medio que ha tardado el programa en generar esas claves.

2.2. Introduce ahora como longitud de clave 2.048 bits y repite el ejercicio anterior.

2.3. Genera ahora un par de claves de 3.000 y de 4.096 bits. Esta última clave puede tardar bastantes segundos en generarse.

2.4. Saca conclusiones de los tiempos medios de generación de clave encontrados en los apartados anteriores.

2.5. ¿Es muy costoso en cómputo generar una clave actual de 2.048 bits?

2.6. ¿Por qué aquí ha sido costoso en tiempo de cómputo generar una clave de 4.096 bits y en el ejercicio 1 no?

2.7. Repite los ejercicios anteriores activando primero la opción Clave pública = 65.537, después p y q de igual tamaño, o bien ambas.

2.8. Activa la opción Primos seguros y genera media docena de claves de 1.024 bits.

2.9. Intenta generar ahora con Primos seguros una clave de 1.280 bits. Nota: puede tardar más de 2 minutos.

2.10. ¿Qué conclusiones sacas de lo observado en los apartados 2.8 y 2.9?

2.11. Busca en Internet la diferencia que existe entre primos seguros (*safe primes*) y primos fuertes (*strong primes*).

III. Comprobación de la primalidad de p y q

Ejercicio 3)

3.1. Genera automáticamente diversas claves de longitudes 1.024, 2.048 y 4.096 bits, con opción Clave pública = 65.537, p y q igual tamaño, pero sin activar Primos seguros. En cada uno de los casos, una vez creada la clave, comprueba que p y q son primos con la opción del menú Test Primalidad, usando Miller-Rabin y Fermat. Puedes modificar si lo deseas el número de iteraciones, que por defecto son 50.

3.2. ¿Es una tarea computacionalmente costosa saber si un número de hasta unos 2.048 bits es o no primo?

3.3. Limpia la clave e introduce como primos p y q estos valores de 4.096 bits de una clave RSA de 8.192 (2^{13}) bits. Haz un Test de Primalidad para comprobar cuánto tarda el programa en comprobar que son primos p y q. Saca conclusiones.

Primo p

664.215.146.591.313.167.758.655.648.182.226.542.587.758.978.366.447.446.865.296.746.475.472.404.692.191.923.572
.481.304.021.217.063.545.039.330.843.562.663.662.679.639.490.885.193.674.704.154.222.999.106.627.211.409.114.78
0.085.109.241.229.679.588.465.932.690.750.545.801.831.161.763.112.657.279.027.814.082.563.499.580.623.843.105.0
85.337.277.484.097.802.713.489.716.457.087.828.974.099.104.424.520.003.498.999.153.019.589.070.604.733.826.304.
607.088.778.070.691.328.648.895.922.985.476.513.557.392.740.468.019.501.271.554.169.770.792.028.456.063.436.154
.210.398.253.848.564.551.835.507.175.807.925.340.464.486.779.757.609.758.694.528.161.149.634.325.375.681.675.69
6.751.010.623.640.593.762.337.786.092.128.204.806.204.644.171.246.616.397.647.035.948.280.367.280.166.346.468.2
09.020.864.523.126.676.529.525.900.640.103.283.335.444.821.626.275.827.435.367.135.380.561.219.616.734.154.815.
430.869.826.927.080.133.622.327.373.519.417.712.839.842.040.926.372.981.371.325.174.143.102.554.943.582.866.936
.484.400.969.595.772.166.464.301.350.376.584.787.853.287.684.481.438.210.529.209.141.212.078.127.502.696.551.54
4.730.364.994.283.349.880.619.204.004.185.384.749.444.051.219.280.951.297.642.639.968.236.587.882.455.980.943.2
73.760.427.869.878.392.227.475.848.257.267.445.045.489.264.482.815.310.780.705.360.773.708.002.772.772.551.940.
580.026.127.244.105.825.199.859.634.474.176.399.934.032.974.596.261.191.903.370.098.900.700.822.123.281.344.588
.022.616.777.589.556.749.440.594.826.113.286.601.986.759.309.923.877.814.299.856.007.192.207.305.249.017.749.68
1.526.274.161.054.833.302.958.027.178.647.964.708.489.389.138.531.491.344.266.120.451.851

Primo q

967.276.417.214.798.374.983.112.605.629.366.519.257.652.416.915.590.818.342.092.536.501.548.274.503.556.517.642
.910.702.023.798.495.086.779.793.269.041.611.772.099.312.076.312.524.469.149.727.339.088.386.165.611.497.831.61
3.226.845.754.661.133.997.311.958.520.780.975.678.914.330.268.763.548.545.157.804.318.487.800.754.376.190.960.5
36.497.790.717.027.546.799.294.830.490.547.023.656.633.664.216.305.645.015.654.741.398.894.813.977.732.786.195.
079.542.880.949.791.264.767.074.947.938.460.782.541.174.753.186.195.435.994.774.447.231.028.304.606.841.248.261
.719.657.196.482.551.598.494.448.883.758.321.630.414.930.435.568.861.098.085.561.516.876.729.233.956.555.596.12
9.467.597.581.612.467.089.460.875.039.669.797.925.864.087.372.442.307.258.521.137.690.029.744.897.307.540.449.9
40.011.202.135.023.140.011.906.888.187.052.672.461.794.345.625.386.397.990.739.044.663.699.925.927.633.442.163.
566.058.087.054.521.872.915.856.033.933.854.453.134.510.937.258.586.527.645.392.175.875.185.421.079.682.446.946
.335.958.066.435.166.932.011.819.093.556.272.694.984.106.168.757.467.720.100.316.415.888.272.068.702.798.683.36
8.946.168.620.398.177.735.794.303.303.597.149.461.252.847.313.749.120.285.259.842.907.697.397.932.876.651.778.0
63.919.968.330.172.020.500.216.608.943.267.032.082.175.933.494.533.388.096.515.655.712.558.548.897.262.675.552.
835.721.370.215.684.799.491.387.673.872.727.630.094.361.155.871.067.088.874.162.627.102.783.071.832.461.757.416
.719.647.502.109.112.596.882.322.230.809.610.703.820.776.087.846.560.740.022.004.197.138.782.890.466.966.982.65
6.862.987.057.465.482.159.577.558.588.323.754.578.803.987.895.354.552.510.285.245.164.387

3.4. Genera la clave anterior Manualmente, introduciendo los valores de p y q y e = 65.537.

3.5. Si tienes más de 5 minutos, genera de forma Automática con genRSA v2.1 una clave de 8.192 bits con p y q igual tamaño. Aunque tarda mucho tiempo en encontrar los dos primos, recuerda que genRSA v2.1 es un software educativo y de prácticas y que éste es el tamaño mayor de claves que genera.

Comprueba tu trabajo:

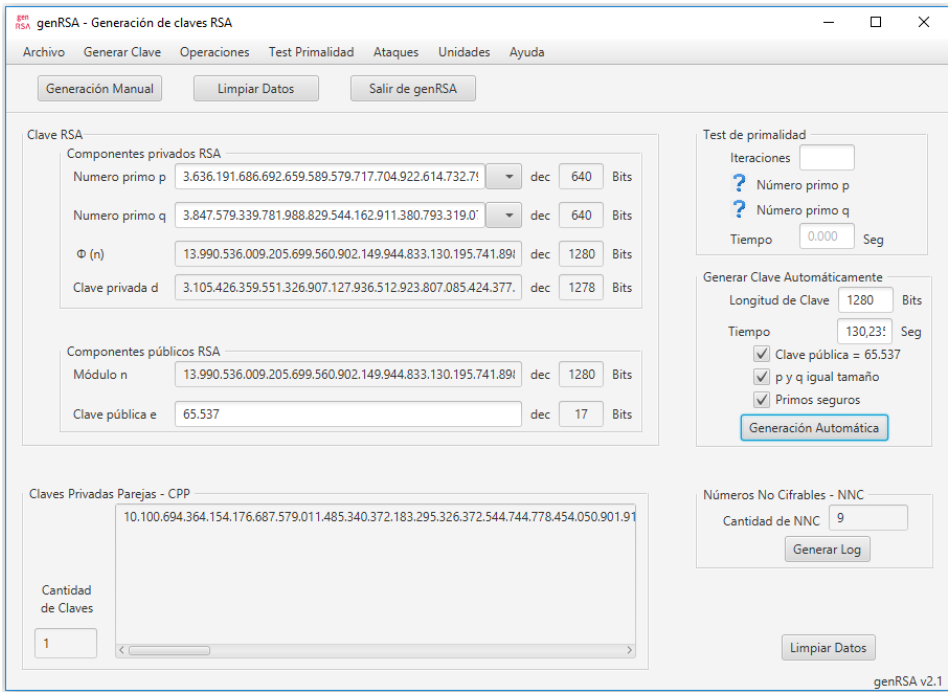

Figura 4. Clave RSA de 1.280 bits generada automáticamente.

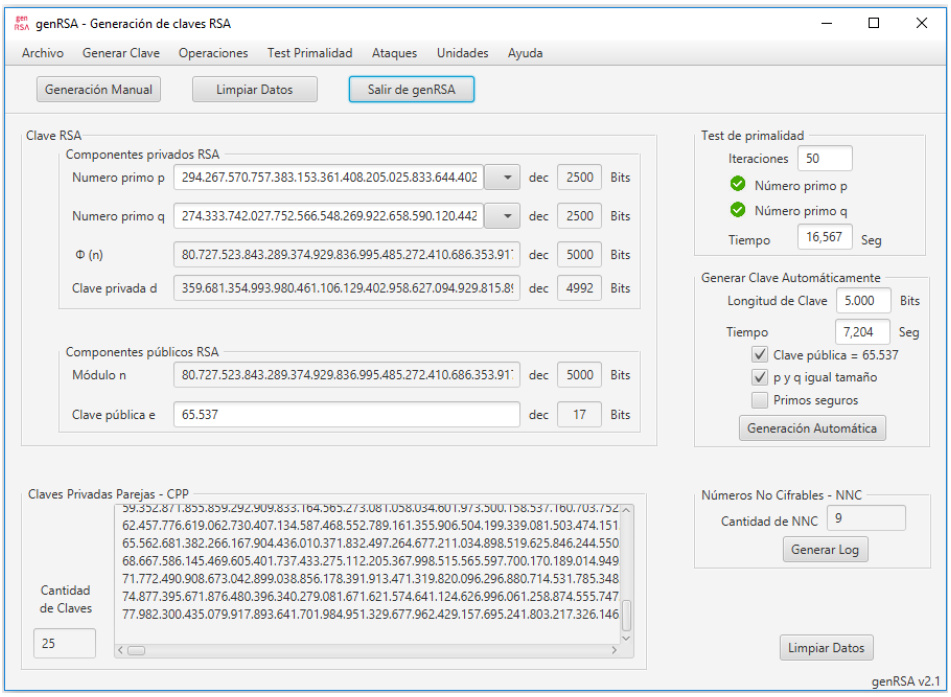

Figura 5. Clave RSA de 5.000 bits generada automáticamente.

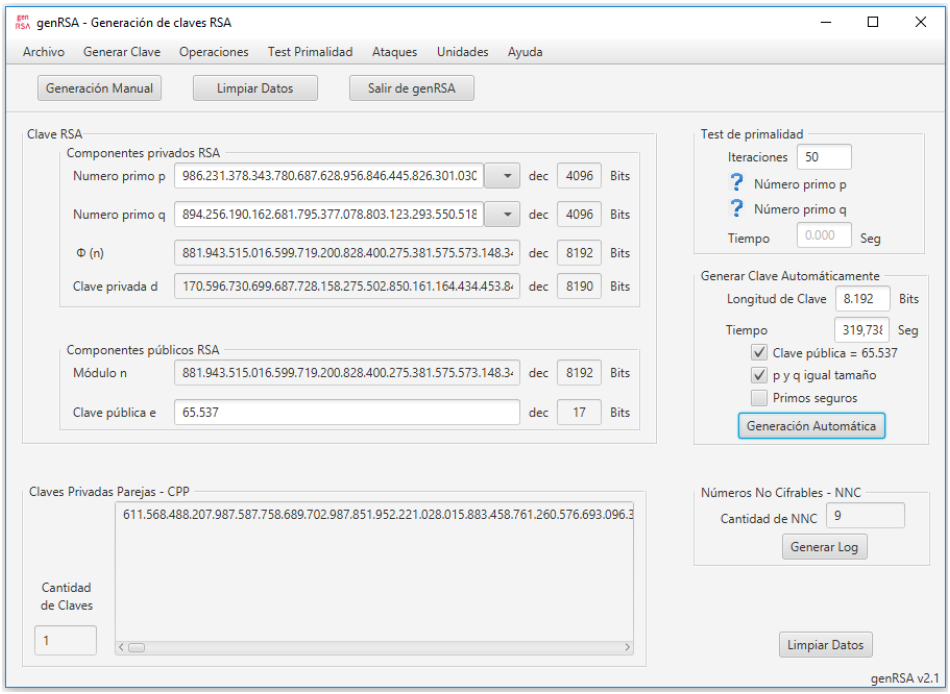

Figura 8. Clave RSA de 8.192 bits generada automáticamente.

IV. Introducción a la generación de claves RSA con OpenSSL

Ejercicio 4)

4.1. Con OpenSSL puedes generar claves RSA de 8.192 bits (2^{13}) en muy poco tiempo. La generación de claves RSA con OpenSSL y su conversión a formato texto lo veremos en detalle en un próximo cuaderno de laboratorio.

4.2. Para ello, descarga desde la Web indicada el programa Win64 OpenSSL v1.1.0h o superior e instálalo en tu PC. También puedes descargarlo para máquinas de 32 bits.

4.3. Desde el modo comando (Inicio → cmd → Enter), accede a la carpeta del programa C:\OpenSSL-Win64\bin>.

4.4. Ejecuta este comando: openssl genrsa -out MiClaveGrande 8192

4.5. Genera un par de claves y observa el tiempo que tarda OpenSSL en generarla.

4.6. ¿Qué son significan esos puntos que van apareciendo y que terminan en ++?

4.7. Observa que, por defecto, la clave pública es el número 4 de Fermat. En un próximo cuaderno de laboratorio veremos y analizaremos este tema en detalle.

4.8. Genera ahora un par de claves de 16.384 bits (2^{14}). Observa cómo baja la velocidad con la que ahora se mueve el cursor marcando los puntos durante su generación.

4.9. Genera una clave de 32.768 (2^{15}) bits e intenta crear otra clave de 65.536 (2^{16}) bits para comprobar la dificultad con la que OpenSSL genera esas claves con primos tan grandes. Ten presente que la primera operación puede tardar varios minutos y la última muchas horas. Puedes abortar la operación en cualquier momento, pulsando las teclas Ctrl + C.

4.10. ¿Sería recomendable trabajar hoy con claves RSA mayores que 2^{15}? Saca conclusiones.

Comprueba tu trabajo:

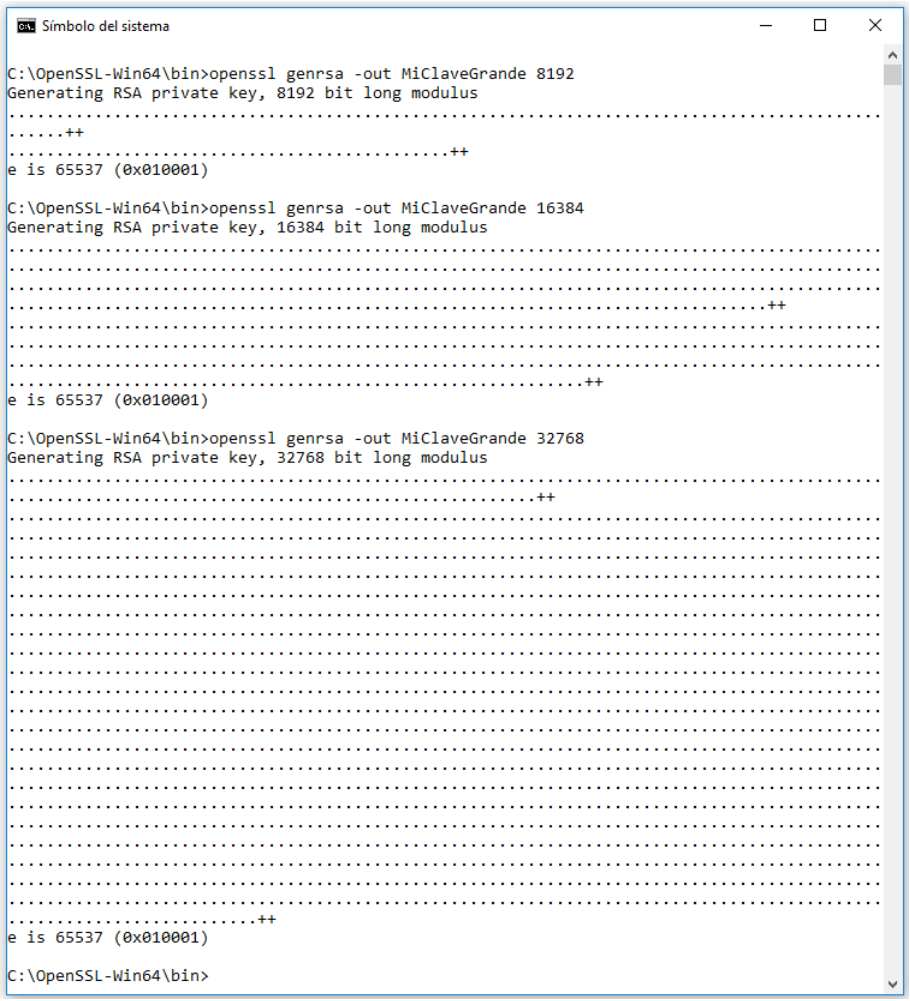

Figura 9. Generación de claves RSA de gran tamaño con OpenSSL.

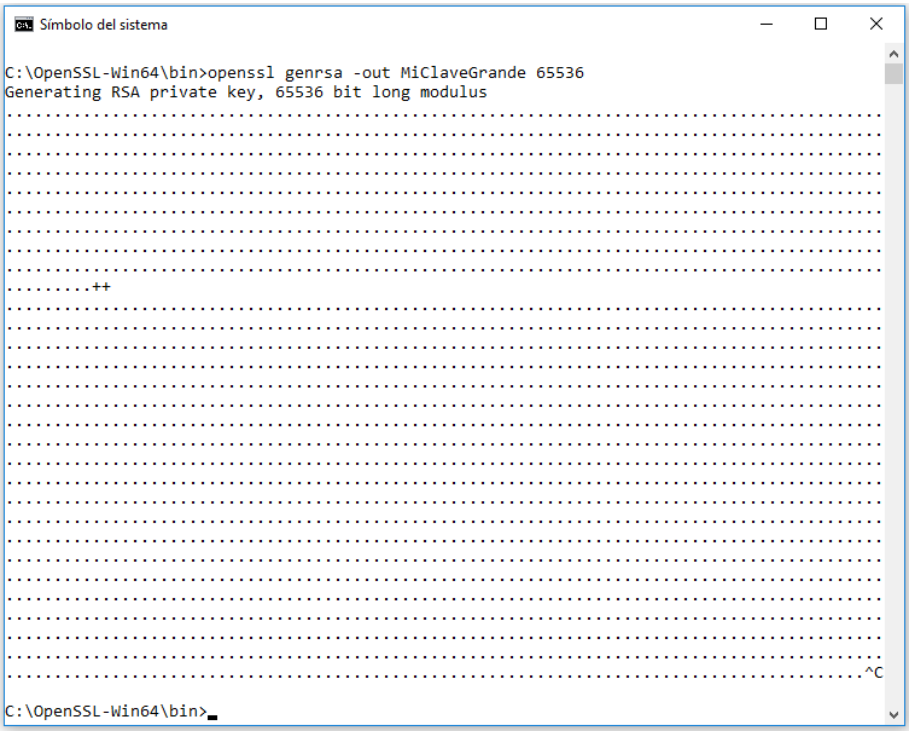

Figura 10. Generación de clave RSA de 65.536 bits, abortada después de 8 horas calculando.

Madrid, 6 de mayo de 2019
Dr. Jorge Ramió Aguirre