2008 Argentine Congress on Computer Science
(Congreso Argentino en Ciencias de la Computación - CACIC 2008)

Factorización de N: recuperación de factores primos a partir de las claves pública y privada

Marcelo José Cipriano
Laboratorio de Investigación en Técnicas Criptográficas y Seguridad Teleinformática
Escuela Superior Técnica – Facultad de Ingeniería del Ejército
marcelocipriano@iese.edu.ar

Abstract

Se puede factorizar N y hallar sus factores primos a partir del conocimiento de la clave pública e y la clave privada d en un criptosistema RSA. Se analiza y resuelve el problema en forma matemática y luego se muestra un algoritmo para su implementación computacional. La aritmética de punto flotante no aporta la exactitud necesaria para las actuales longitudes de las claves. Luego el algoritmo presentado prescinde de ella, utilizando el método de Newton-Raphson para hallar uno de los factores. Se podrá así analizar la solidez de su esquema RSA y cambiarlo en caso de considerarse que no es lo suficientemente seguro.

KEYWORDS: RSA, clave pública, clave privada, primos fuertes.

1. INTRODUCCIÓN.

RSA[1] [01] [02] [03] es un procedimiento criptográfico para cifrar y descifrar mensajes. El mismo también se usa en las aplicaciones de firma digital. Fue patentado en 1983 con el número de patente 4.405.829[2], cuya fecha de expiración ha sido el 21 de Septiembre de 2000.

Emplea una clave para cifrar y otra diferente para descifrar. Esto representó un avance en la criptografía, pues fue el primer de ***criptosistema de clave pública* o *asimétrico*** [05]. Actualmente es uno de los sistemas criptográficos más empleados por su demostrada robustez. [06].

Se puede usar además con otros sistemas criptográficos utilizando un servicio mixto, conocido por el nombre de *envoltura digital RSA*: enviar con RSA la clave simétrica de un esquema DES o triple DES al destinatario. Así la criptografía asimétrica brinda servicios a la criptografía simétrica al encargarse de la distribución de la clave [07].

Hasta ahora el esquema RSA ha resistido en forma eficaz los ataques a los que se ha enfrentado. Es también cierto, que la seguridad que ofrece es ***computacional***, y por lo tanto, a medida que aumenta la potencia y velocidad de los ordenadores, también aumenta el tamaño de las claves que es necesario emplear.

---

[1] El acrónimo RSA, corresponde a los apellidos de los científicos que lo crearon: *Ron RIVEST*, *Adi SHAMIR* y *Len ADLEMAN* [04].

[2] El número 4.405.829 es un número primo.

RSA - Laboratories ofrecía interesantes premios al que lograba factorizar los números propuestos. Así buscaban demostrar la robustez del sistema e incentivar la búsqueda de algoritmos de factorización. ***El desafío RSA*** o como se conoce en inglés ***The RSA Factoring Challenge*** ya ha sido quitado de la Web, y sólo se listan los números ya factorizados y los premios entregados. Ello es posible, a que durante 2007 ningún desafío ha sido resuelto [08].

RSA estaría a buen resguardo detrás de lo que se conoce como el ***problema de la factorización*** pues es extremadamente arduo factorizar números compuestos muy grandes. Sin embargo aún no se ha demostrado que romper RSA signifique factorizar N.

2. BASES MATEMÁTICAS DE RSA.

2.1. Teoremas de Euler y Fermat: pilares de RSA.

RSA cifra y descifra mensajes que por medio de exponenciación modular. Se basa íntegramente en los teoremas de Euler y Fermat [11] [12].

Teorema de Euler

$$\text{Si } \operatorname{mcd}(a\,;n)=1 \Rightarrow a^{\Phi(n)} \equiv 1 \pmod{n}$$

Teorema de Fermat

$$\text{Si p primo}, \Phi(p)=p-1 \Rightarrow a^{\Phi(n)} = a^{p-1} \equiv 1 \pmod{n}$$

Corolario de Euler-Fermat:

$$\text{Si } p, q \text{ primos} / n = p\,q;\ k \text{ y } m \in Z^{+} / m < n \Rightarrow m^{k\Phi(n)+1} = m^{k(p-1)(q-1)} \equiv m \pmod{n}$$

Expresión RSA

$$\text{Sean } e,\ d \in Z^{+} / e\,d \equiv 1 \pmod{\Phi(n)} \Rightarrow m^{e\,d} \equiv m \pmod{n}$$

Llamamos al número ***e clave pública,*** y a ***d clave privada;*** entonces, para cifrar un emisor debe conocerse el par **(*N; e*)** y el receptor para descifrar el par **(N; d)** [09] [10].

Sin embargo, a fin de evitar que cada persona tenga una clave pública diferente y se convierta en un problema almacenar tantas claves para cifrar, se usan las mismas claves: el número 65537 ($2^{16}+1$).

2.3. Elección de los factores primos de N: primos fuertes

Los factores primos de **n, p** y **q** deben ser **primos fuertes** [13] [14], cumpliendo las siguientes condiciones:

1. **p y q** deben diferir en pocos dígitos, aunque no muy cercanos.

   Si ignoramos la primera condición entonces nuestro esquema en particular estaría debilitado porque sucumbiría al método de Fermat (método para factorizar un número compuesto partiendo de la raíz cuadrada del mismo y buscando hallar dos números cuadrados perfectos), probando con número mayores que la raíz cuadrada de N.

2. **(*p-1*)** y **(*q-1*)** deben contener factores primos grandes.

   Caso contrario, tampoco **Φ(N)** los tendría, pudiéndose probar todas las potencias modulares del criptograma ***(v+1)/e*** variando ***v*** en busca que dicho cociente sea entero hasta obtener el texto en claro.

3. **mcd (*p-1; q-1*)** debe ser pequeño, de ser posible 2.

   Si **mcd (*p-1; q-1*)** fuese grande, entonces, **mcm (*p-1; q-1*)** sería pequeño comparado con ***ϕ* (N).** Luego cualquier inverso de ***e*** módulo m podría descifrar el mensaje.

En la práctica se elige primo grande, llamado ***r*** y con él hallamos a los otros, de manera que ***p*** y ***q*** sean también primos:

$$p = 2r + 1 \wedge q = 2p + 1 \Rightarrow N = pq \qquad (1)$$

Seguidamente se evalúa la función de Euler

$$\Phi(\mathrm{N}) = (\mathrm{p}-1)(\mathrm{q}-1) \Rightarrow \mathrm{p} + \mathrm{q} = \mathrm{n} + 1 - \Phi(\mathrm{N}) \ . \qquad (2)$$

3. FACTORIZACIÓN DE *N.*

3.1 Polinomio-RSA.

Definición: llamaremos Polinomio-RSA al polinomio entero y mónico de grado 2 que tiene a los primos **p** y **q** como sus raíces.

$$P(x) = (x - p)(x - q) = x^2 - (p + q)x + N \, . \qquad (3)$$

$$p \ ; q = \frac{(p+q) \pm \sqrt{(p+q)^2 - 4N}}{2} \, . \qquad (4)$$

Reemplazando en (2)

$$p; q = \frac{\left[\, N + 1 - \phi(N) \,\right] \pm \sqrt{\left[N + 1 - \phi(N)\right]^2 - 4N}}{2} \, . \qquad (5)$$

Como $p \neq q$ entonces

$$\Delta = \left[N + 1 - \phi(N)\right]^2 - 4N > 0 \, . \qquad (6)$$

Despejando ***Φ(N)***:

$$N + 1 - 2\sqrt{N} > \phi(N) \, . \qquad (7)$$

Sea

$$ed \equiv 1 \bmod \phi(N) \Rightarrow \exists k \in Z^+ / ed = \phi(N)k + 1 \, . \qquad (8)$$

(9)

$$\frac{ed-1}{k} = \phi(N)$$

Por (7) (9) tenemos

$$\frac{ed-1}{N+1-2\sqrt{N}} < k\,. \qquad (10)$$

El error que se introduce en el cálculo de ***k*** al usar esta aproximación es muy pequeño: sea la expresión (7) vemos que ambos miembros de la desigualdad tienen un orden de magnitud semejante. Sea ***m*** el orden de magnitud de *N*, entonces

$$N+1-2\sqrt{N} > \phi(N) \approx 2^m\,. \qquad (11)$$

Luego

$$\frac{1}{N+1-2\sqrt{N}} < \frac{1}{\phi(N)} \approx 2^{-m}\,. \qquad (12)$$

Tomando un entorno de centro en el ***k*** hallado podemos hallar el valor de ***Φ(N)*** y con él podemos usar la expresión (10) para hallar ***p*** y ***q***.

$$\frac{\left[N+1-\phi(N)\right] \pm \sqrt{\left[N+1-\phi(N)\right]^2 - 4N}}{2} = (p;q)\,. \qquad (5)$$

De esta manera puede hallarse los factores primos ***p*** y ***q*** a partir del conocimiento del módulo, clave pública y privada en un esquema RSA.

4. DIFICULTADES COMPUTACIONALES.

4.1 Aritmética de punto flotante y Norma IEEE 754.

Para realizar en forma efectiva todos los procedimientos y cálculos indicados, formalmente en los apartados anteriores debemos tener el auxilio de un ordenador.

Dado que la seguridad proviene del tamaño de los números en cuestión, prontamente hallaremos dificultades importantes en la etapa de implementación por las limitaciones de la aritmética de punto flotante.

Las expresiones (5) y (10) necesitan procesar operaciones de división y extracción de raíces. Estas operaciones devuelven números reales como resultado.

La norma **IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985)** [15] o más conocida como **IEEE 754** [3]es el estándar más ampliamente usado para las operaciones de punto flotante.

---

[3] También conocida como IEC 60559:1989. Binary floating-point arithmetic for microprocessor systems (originalmente el número de referencia era IEC 559:1989).

La representación de números reales es de la forma $\mathbf{r = m\ b^e}$, donde ***m*** es la mantisa, ***b*** la base y ***e*** el exponente.

La base se elige de acuerdo al sistema numérico de representación, es decir: octal, decimal, binaria, etc. La representación de doble precisión, por ejemplo, asigna 64 bits a cada número real. Un bit para el signo, los 11 siguientes para el exponente y los restantes para la mantisa.

Con claves RSA del orden de 1024 bits e inclusive más, se está al límite de la precisión requerida para realizar cálculos. Para recuperar la precisión perdida, se emplearán herramientas de Cálculo Numérico como es el método de Newton-Raphson [16].

5. ALGORITMO PARA LA FACTORIZACIÓN DE *N*.

5.1. Generalidades.

Se presenta una solución algorítmica expresada en seudo-código que hace uso del método de Newton-Raphson con el cual se puede hallar una de las raíces del polinomio RSA.

Al trabajar en aritmética entera, este método es muy veloz y confiable, dejando de lado las dificultades de precisión computacional.

Una de las entradas del algoritmo es un valor inicial aproximado a la raíz buscada y cuanto más cercana sea a la raíz, mejor será el desempeño del método.

Sin embargo, se opta asumir por defecto esta entrada eligiendo la semisuma de las raíces, es decir $\mathbf{(N + 1 - \phi(N))}$.

En las pruebas realizadas, el rendimiento del algoritmo halló la raíz buscada en muy pocas iteraciones, dependiendo de la magnitud de *N*, obviamente.

5.2 Observación para programadores de Python o VBScript.

Estos lenguajes de programación no requieren declarar tipo de las variables porque lo asumen cuando se les asigna por primera vez el contenido.

Para trabajar con **Python**[4] en particular, al final de todas las entradas se agrega la letra **L** indicando que el contenido es un entero grande y no tiene limitación en su almacenamiento.

5.3. Seudo-código del algoritmo.

ENTRADA N: módulo RSA
e: clave pública
d: clave privada
m: máximo de iteraciones

4 Este lenguaje de programación permite, entre otras cosas, trabajar con numero esteros muy grandes (de 1.000 o más bits).

SALIDA los factores primos p y q del número N
Paso 1 inicializar N, f, k, a, p, s, i
Paso 2 tome k = entero [(e * d – 1) / (N + 1 – 2 * N ^ 0.5)]
Paso 3 tome f = entero [(e * d – 1) / k]
Paso 4 tome s = N + 1 - f
Paso 5 tome i = 1
Paso 6 tome a = (N + 1 – f) / 2
Paso 7 mientras i <= m haga pasos 8-11
Paso 8 tome p = a – entero [(a ^ 2 – a * s + n) / (2 * a – s)]
Paso 9 Si valor_absoluto (a – p) = 0 entonces
tome q = N / p
SALIDA p; q
PARE
Paso 10 Tome i = i + 1
Paso 11 Tome a = p
PASO 12 SALIDA ("no se alcanza a factorizar N")
SALIDA ("aumente el máximo de iteraciones")
PARE

6. CONCLUSIONES.

El procedimiento matemático desarrollado permite recuperar **p** y **q** a partir de la información del trío **(N, e, d)**.

Futuros estudios podrían permitir estudiar las propiedades del polinomio-RSA.

La aritmética de punto flotante que se norma a través de IEEE 754 tiene sus limitaciones para abordar con precisión la operatoria con números tan grandes. Cambios de paradigmas criptográficos llevan a concluir que las claves irán en aumento.

Tal vez se desarrollarán lenguajes y normas con mayores niveles de precisión en sus representaciones para números reales. Este trabajo permite estimar que esto no sería necesario. Por el contrario, algoritmos como el presentado evitarían estas dificultades.

La aplicación práctica desarrollada permite comprobar si implementaciones del esquema RSA tienen vulnerabilidades ocultas debajo del **N** elegido para recuperar la seguridad perdida.

7. BIBLIOGRAFÍA.

[01] RSA Laboratories. "Public-Key Cryptography Standards (PKCS) #1 v2.1", 2002. Estándar para la aplicación del esquema RSA
[02] IEEE Std 1363-2000: Standard Specifications for Public Key Cryptography. IEEE, August 2000.
[03] IEEE P1363 working group. IEEE P1363a D10: *Standard Specifications for Public Key Cryptography: Additional Techniques*. November 1, 2001. http://grouper.ieee.org/groups/1363.
[04] R. Rivest, A. Shamir, L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, Vol. 21 (2). 1978. pp.120–126.

[05] Diffie, W. y M.E.Hellman. "*New directions in cryptography"*, IEEE Transactions on Information Theory 22. 1976, pp. 644-654.
[06] E.Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. *RSA-OAEP is Secure under the RSA Assumption.* In J. Kilian, editor, Advances in Cryptology – Crypto 2001, volume 2139 of Lecture Notes in Computer Science, pp. 260 – 274. Springer Verlag, 20011
[07] MENEZES, A y otros. *Handbook of applied cryptography*. CRC PRESS, 1997.
[08] Http://www.rsa.com/rsalabs/node.asp?id=2092.
[09] M. Bellare and P. Rogaway. *Optimal Asymmetric Encryption – How to Encrypt with RSA*. In A. De Santis, editor, Advances in Cryptology – Eurocrypt '94, volume 950 of Lecture Notes in Computer Science, pp. 92 – 111. Springer Verlag, 1995.
[10] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations *Among Notions of Security for Public-Key Encryption Schemes*. In H. Krawczyk, editor, *Advances in Cryptology – Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pp. 26 – 45. Springer Verlag, 1998.
[11] KOBLITZ, Neal. *A Course in Number Theory and Cryptography*, Graduate Texts in Math. No. 114, Springer-Verlag, New York, 1987. Second edition, 1994.
[12] KOBLITZ, Neal. *Algebraic Aspects of Cryptography, Algorithms and Computation in Mathematics* Vol. 3, Springer-Verlag, New York, 1998.
[13] R.L. RIVEST, "Are 'strong' primes needed for RSA?", unpublished manuscript, 1991.
[14] RAMIÓ AGUIRRE, Jorge. *Vulnerabilidades y Ataques al Sistema de Cifra RSA.* Madrid, 2005. Diapositiva 8 de su presentación.
[15] GOLDBERG, David, *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, Computing Surveys., Association for Computing Machinery, Inc., March, 1991.
[16] BURDEN, R y FAIRES, D. *Análisis Numérico* (Sexta Edición). Internacional Thomson Editores. México, 1998.