El sistema criptográfico RSA

Félix Delgado - Ana Núñez

Universidad de Valladolid

Curso 2020 - 2021

RSA

- En 1975 Walter Diffie y Martin Hellman en *New directions in cryptography* sientan las bases de la Criptografía de clave pública.
- Sin embargo no se conoce ningún sistema criptográfico de clave pública hasta que en 1978 Ronald Rivest, Len Adleman y Adi Shamir (MIT) proponen el sistema RSA
- El algoritmo fue patentado por el MIT en 1983 en Estados Unidos. La patente expiró el 21 de septiembre de 2000.
- En 1997 se hizo público que Clifford Cocks, un matemático británico que trabajaba para la agencia de inteligencia británica GCHQ, había descrito un sistema equivalente en un documento interno en 1973.
- Seguía ideas desarrolladas por James H. Ellis a finales de los 60.

Generación de claves

Cada usuario hace las siguientes operaciones:

1. Elige números primos p, q.
2. Calcula $n = pq$ y $\varphi(n) = (p-1)(q-1)$.[(1)]
3. Elige arbitrariamente un entero e tal que $0 < e < \varphi(n) = (p-1)(q-1)$ y $\text{mcd}\,(e, \varphi(n)) = 1$
4. Calcula el inverso modular de e módulo $\varphi(n)$, $d \equiv e^{-1} \mod \varphi(n)$.

La clave pública es el par (n, e) y la privada es d.

(1) $\varphi(n)$ es el indicador de Euler:

$$\varphi(n) = \#\{m \in \mathbb{N} \mid 1 \leq m < n \text{ y } \text{mcd}\,(n, m) = 1\}.$$

$\varphi(p) = p - 1$; $\varphi(pq) = (p-1)(q-1)$.

Cifrado y descifrado RSA (1)

Cifrado:

$$\begin{array}{rccl} E_{(n,e)} & \mathcal{M} = \mathbb{Z}_n & \to & \mathbb{Z}_n = \mathcal{C} \\ & M & \mapsto & M^e \mod n \end{array}$$

Desifrado:

$$\begin{array}{rccl} D_d & \mathbb{Z}_n & \to & \mathbb{Z}_n \\ & C & \mapsto & C^d \mod n \end{array}$$

Extensión inocente a mensajes arbitrarios $M = M_1 M_2 \dots M_r$ con $M_i \in \Sigma = \mathbb{Z}_N$ (e.g. $N = 256$):

- Para $i = 1, \dots, r$ hacemos $C_i := M_i^e \mod n \in \mathbb{Z}_n$.
- Mensaje cifrado $C := (C_1, \dots, C_r)$.

Inconvenientes: El mensaje cifrado tiene salida en $\mathbb{Z}_n$ y no en $\mathbb{Z}_N$. Puesto que $n >> N$ además de usar otro alfabeto en la salida estariamos utilizando mucho espacio.

Cifrado con RSA (2)

Existe una forma mejor, que consiste en entender el RSA como un cifrado por bloques, y que explicamos a continuación.

- Tamaño de los bloques, k, es el entero tal que $N^k \leq n < N^{k+1}$).
- Bloque de longitud k: $M = M_1 \dots M_k$ con $M_i \in \mathbb{Z}_N$, $(0 \leq M_i < N)$.
- Calculamos $m = M_1 \cdot N^{k-1} + M_2 \cdot N^{k-2} + \dots + M_k < N^k \leq n$. (Interpretamos M como la escritura en base N del entero m.)
- Calculamos el cifrado de m: $c = m^e \bmod n < n < N^{k+1}$.
- Calculamos la expresión de c en base N:

$$c = C_1 \cdot N^k + C_2 \cdot N^{k-1} + \dots + C_{k+1}.$$

  Nótese que la longitud de c es $k+1$.
- Cifrado del bloque M: $C = C_1 C_2 \dots C_{k+1}$.

Cifrado con RSA (2)

El protocolo anterior es un cifrado en bloques:

$$E_{(n,e)} : (\mathbb{Z}_N)^k \to (\mathbb{Z}_N)^{k+1}.$$

Se implementa en mensajes arbitrarios en modo ECB:

- Se calcula k tal que $N^k \leq n < N^{k+1}$.
- Se divide M en bloques de longitud k (usando algún procedimiento que complete el último de ser necesario), $M = B_1 \dots B_r$.
- Se cifra por el procedimiento anterior cada bloque B_i, $C_i = c(B_i)$ (bloques de longitud $k+1$).
- El cifrado de M es $c(M) = C_1 C_2 \dots C_r$. Obsérvese que el número de símbolos ha aumentado en uno por bloque.

Se podría también aplicar una versión ligeramente modificada del modo CBC. No tienen sentido los modos CFB u OFB.

Descifrado con RSA (2)

El descifrado (ya descrito en modo ECB), $D_d : (\mathbb{Z}_N)^{k+1} \to (\mathbb{Z}_N)^k$:

1. Se calcula k tal que $N^k \leq n < N^{k+1}$.
2. Se divide C en bloques de longitud $k+1$, $C = C_1 \dots C_r$.
3. Para cada bloque $C_i = C_1^i \dots C_{k+1}^i$, calculamos el entero $c_i = C_1^i \cdot N^k + C_2^i \cdot N^{k-1} + \dots + C_{k+1}^i < n$.
4. Calculamos $b_i = c_i^d \bmod n$.
5. Escribimos b_i en base N con longitud k:

$$b_i = B_1^i \cdot N^{k-1} + B_2^i \cdot N^{k-2} + \dots + B_k^i$$

   El descifrado de C_i es $B_i = B_1^i B_2^i \dots B_k^i$.
6. El mensaje descifrado es $M = B_1 \dots B_r$.

¿Por qué funciona?

Puesto que $ed \equiv 1 \mod \varphi(n)$, tendremos

$$D_d(E_{n,e}(M)) = M^{1+k\varphi(n)} = M(M^{\varphi(n)})^k \equiv M \mod n$$

SIEMPRE que $\text{mcd}\,(M, n) = 1$.

Teorema de Euler: Si m, n son enteros y $\text{mcd}\,(m, n) = 1$. Entonces

$$m^{\varphi(n)} \equiv 1 \mod n\,.$$

Más adelante justificaremos la validez del Teorema de Euler y comprobaremos que de hecho la última igualdad es cierta aún en el caso en que $\text{mcd}\,(M, n) \neq 1$.

Diffie-Hellman para RSA (1)

Comencemos por las operaciones que deben ser computacionalmente sencillas.

1. Buscar primos "grandes" p, q.
   ¿Es posible encontrar primos grandes de forma eficiente?
2. Calcular $n = pq$ y $\varphi(n) = (p-1)(q-1)$.
   ¿Es realmente eficiente la multiplicación?
3. Buscar $e < \varphi(n)$ tal que $\text{mcd}\,(e, \varphi(n)) = 1$. Calcular $d \equiv e^{-1} \mod \varphi(n)$.
   Algoritmo de Euclides extendido. ¿cómo es de rápido?
4. Cifrar y descifrar: Calcular $M^e \mod n$ y $C^d \mod n$.
   ¿Hay algoritmos eficientes de exponenciación modular?

El *supuesto* problema intratable en el que se basa el RSA es el problema de la factorización.
¿No se trata del mismo problema que saber si un número es primo?

Diffie-Hellman para RSA (2): Problema de primalidad

La búsqueda de primos grandes (150, 200 cifras decimales) es, sin duda, el apartado más complicado.
Un método sencillo de proceder:

Problema de primalidad: Disponer de un algoritmo eficiente A de manera que, dado un entero w, nos permita saber si es o no primo.

Posteriormente, podemos probar con números elegidos al azar de cierto tamaño mínimo y someterlos al algoritmo A hasta dar con un número primo.

Este método funcionaría siempre que haya muchos primos de un cierto tamaño.
Para estimar la eficiencia del método (probabilista) necesitamos conocer la DENSIDAD del conjunto de números primos en el conjunto de los números naturales.

Diffie-Hellman para RSA (3)

El *supuesto* problema intratable en el que se basa el RSA es el problema de la factorización.

¿Hay bases para pensar que es un problema intratable?

¿Qué algoritmos de factorización se conocen?

Evidentemente, si conocemos los primos p y q con $n = pq$, podemos calcular $\varphi(n)$ y d a partir de (n, e). Pero ...

¿Es equivalente conocer p y q a conocer $\varphi(n)$?

¿Es imprescindible conocer $\varphi(n)$ para calcular d a partir de (n, e)?

¿Es posible recuperar M a partir de $M^e \mod n$ sin conocer d?