Tema 2 - Seguridad y criptografía

Hoja de ejercicios
Tema 2

7 de abril de 2019

1. Criptosistema RSA

Problema 1

Alicia quiere mandar un mensaje a Benito a través de un canal no seguro. Para ello, deciden acordar que el algoritmo criptográfico a utilizar será RSA. Alicia elige los números primos $p = 3$ y $q = 7$. Tras los cálculos pertinentes, Alicia obtiene que $e = 5$. Contesta a las siguientes preguntas:

1. Comprobar si el número e que ha generado el algoritmo cumple todas las condiciones necesarias para poder ser utilizado en RSA.

2. Calcular el inverso multipicativo, d, necesario para poder descifrar el mensaje enviado por Alicia.

3. Calcular clave pública y privada para que Benito pueda descifrar el criptograma enviado por Alicia.

4. Alicia quiere mandar el texto plano $M = 9$ a Benito. ¿Cuál será el criptograma generado por Alicia al emplear RSA con los datos anteriores?

5. Comprueba que, al descifrar el criptograma anterior, se recupera el mensaje en claro M.

Solución

1. La condición fundamental que e tiene que cumplir es que e y n sean primos entre sí. Al ser $n = pq$, con p y q primos y $p \neq e$ y $q \neq e$, e es un parámetro válido.

2. e es conocido como el *exponente público* y necesita su contraparte, d, que deben cumplir $ed = 1 \mod (\phi(n))$. En este caso $5d = 1 \mod (12)$, por lo que tenemos lo que se llama **congruencia lineal** o ecuación de congruencia lineal. En su forma general una ecuación de este tipo tiene la expresión:

$$ax = b \mod m$$

Esta expresión es equivalente a $ax = b + km$, para valores sucesivos de $k = 0, 1, 2....$ Este tipo de expresiones suelen resolverse con el algoritmo extendido de Euclides que, además de encontrar el mcd de dos números a y b, permite encontrar también la mínima combinación lineal de ambos. Pero como en el caso de RSA, siempre será $b = 1$, podemos utilizar un "truco" para valores pequeños de e, d y m, que consiste simplemente en reescribir la expresión:

$$\begin{aligned} 5d &= 1 + 12k \\ d &= \frac{(1 + 12k)}{5} \end{aligned} \tag{1}$$

Ahora, se trata de ir probando valores sucesivos de k hasta encontrar el mínimo valor que hace d entero. En nuestro caso, $k = 2$ hace que $d = 5$.

3. La clave pública y privada son, simplemente, los pares $e, n = 5, 21$ y $d, n = 5, 21$ (que sean iguales es mera casualidad, debido a los pequeños valores con los que estamos trabajando).

4. El criptograma se obtiene con la expresión $C = m^e \mod (n) = 9^5 \mod (21) = 18$.

5. Para recuperar M, simplemente hay que utilizar el exponente privado: $M = C^d \mod (n) = 18^5 \mod (21) = 9$

Problema 2

Considera ahora los siguientes parámetros para RSA, $p = 17$, $q = 43$, $e = 101$. Calcula:

1. $\phi(n)$.

2. Clave pública y clave privada.

3. Si Bernardo quiere enviar el mensaje "SI" a Alicia empleando el alfabeto que se muestras más abajo, ¿cuál será el criptograma generado por RSA y enviado a Alicia?

A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	2	3	4	5	6	7	8	9	10	11	12	13	14
Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	
15	16	17	18	19	20	21	22	23	24	25	26	0	

2. Políticas de contraseñas

Problema 4

Asume un sistema de gestión de contraseñas sencillo, que solo utiliza letras minúsculas, mayúsculas y números (es decir, un alfabeto de 62 caracteres), y que usa SHA256 como función hash. Las contraseñas tienen una longitud de entre 1 y 6 caracteres.

Imagina ahora que un atacante desea pre-computar tablas que transformen cada posible contraseña en su correspondiente valor hash, de forma que puedo luego buscarse éste fácilmente. Si cada carácter almacenado en la tabla requiere un byte, ¿cuáles serían los tamaños finales de las tablas en el caso de:

- contraseñas almacenadas sin salt?
- contraseñas almacenadas con un salt de 8 caracteres de longitud elegidos aleatoriamente de un alfabeto de 64?

Solución

Sin salt El número total de contraseñas es claramente $P = \sum_{k=1}^{6} 62^k \approx 5{,}7731 \cdot 10^{10}$. Almacenar todas estas contraseñas requerirá, por tanto, un total de $S = \sum_{k=1}^{6} k \cdot 62^k$ bytes. Por otro lado, los valores hash ocupan siempre lo mismo, 32 bytes en este caso. Almacenarlos todos requiere, entonces, $H = 32 \cdot P$ bytes, que son aproximadamente 1,84 TB. Normalmente es necesario al menos un carácter separador entre contraseña y hash, para saber dónde empieza y termina cada uno, lo que añade P bytes más. Sumando todas las cantidades, obtenemos un total de $T_{unsalted} = S + H + separador \approx 12{,}96$ TB.

Con salt Comencemos calculando el número total de valores distintos de salt posibles, $n_{salts} = 64^8$. En este caso, es necesario tener el valor hash correspondiente de cada posible combinación de contraseña-salt, puesto que éste último es elegido al azar. Por tanto, esto incrementa el valor de almacenamiento calculado anteriormente T en, al menos, un factor $64^8 = 2^{48}$, dado que el atacante necesitará crear una tabla para cada posible valor de salt.

Por tanto, el valor total $T_{salted} = T_{unsalted} \cdot 64^8 = 3{,}6 \cdot 10^{15}$ TB, que está totalmente fuera de las posibilidades de cualquier atacante actual.