UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

RSA y Ataques

Alan Reyes-Figueroa

Criptografía y Cifrado de Información (Aula 18) 19.octubre.2021

RSA

RSA de libro de texto:

- clave pública: (N, e) encripción: $\mathbf{c} = \mathbf{m}^e \pmod N$,
- clave secreta: (N, d) decripción: $\mathbf{m} = \mathbf{c}^d \pmod N$.

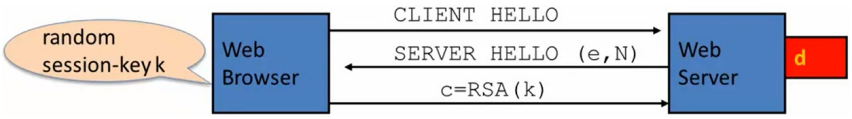
Este método de cifrado es inseguro!!

RSA de libro de texto:

- No es semánticamente seguro.
- La función *trapdoor* de permutación de RSA, por sì sola no define un esquema de cifrado. Falta la parte aleatoria.

RSA

Átaques: Ejemplo de un ataque simple al RSA de libro de texto.



Suponga que se tiene un servidor web, con clave secreta (N, d).

- El cliente establece una comunicación inicial.
- El servidor responde con la clave pública (N, e) para que el cliente pueda comunicarse.
- El cliente genera una clave secreta $\mathbf{k}$ (*premaster secret key*), la cifra como $\mathbf{c} = RSA(\mathbf{k})$, y la envía al servidor.

Vamos a suponer que $\mathbf{k}$ es de 64 bits, esto es $\mathbf{k} \in \{0, 1, \ldots, 2^{64} - 1\}$. Veremos qué ocurre si un atacante lee $\mathbf{c} = \mathbf{k}^e \pmod N$.

RSA

Primero supongamos que $\mathbf{k}$ se factora como el producto de dos enteros, de tamaños similares. Escribimos $\mathbf{k} = \mathbf{k}_1 \cdot \mathbf{k}_2$, con $\mathbf{k}_1, \mathbf{k}_2 < 2^{34}$ (prob. $\approx 20\%$).

Entonces, $\mathbf{c} \equiv \mathbf{k}_1^e \cdot \mathbf{k}_2^e \pmod{N}$, y pasando el primer factor del lado derecho, resulta $\frac{\mathbf{c}}{\mathbf{k}_1^e} \equiv \mathbf{k}_2^e \pmod{N}$.

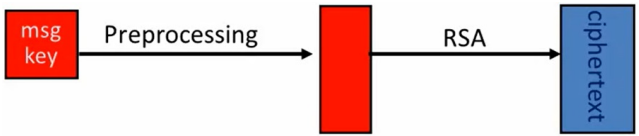
Pero ahora, el atacante conoce $\mathbf{c}$, conoce e, y conoce N, y las únicas incógnitas de esta ecuación $\mathbf{k}_1$ y $\mathbf{k}_2$ están separadas. Podemos hacer un ataque del tipo *meet in the middle*.

- Elaboramos una tabla de valores $\frac{\mathbf{c}}{1^e}, \frac{\mathbf{c}}{2^e}, \frac{\mathbf{c}}{3^e}, \ldots, \frac{\mathbf{c}}{2^{34e}}$ (para $\mathbf{c}/\mathbf{k}_1^e$) $O(2^{34})$.
- Elaboramos una tabla de valores para $1^e, 2^e, 3^e, \ldots, 2^{34e}$ (para $\mathbf{k}_2^e$) $O(2^{34})$.
- Vemos si hay colisiones. Una colisión corresponde a una solución $\mathbf{k}_1, \mathbf{k}_2$.

En general, podríamos probar cada una de las claves en $O(2^{64})$.
Con este método el tiempo se reduce considerablemente a $O(2^{34})$.

RSA en la práctica

Nunca se debe usar esta versión de RSA libro de texto. No encriptarlo directamente.



Siempre se le debe hacer algo al mensaje, antes de la encripción. Por ejemplo, el ISO estándar, generamos una cadena aleatoria **x**, la encriptamos usando RSA, y de ésta derivamos un método de cifrado simétrico.

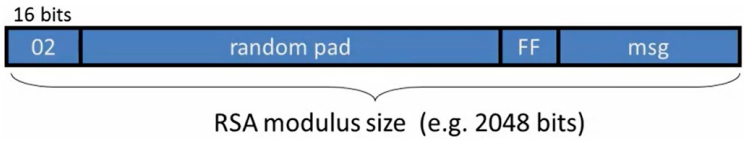
En la práctica, no se suele utilizar el ISO estándar para RSA. En cambio:

Tomamos un cifrado simétricos, por ejemplo AES (de 128 bits), y lo expandimos mediante algún procedimiento (digamos, hasta 2048 bits). Luego aplicamos RSA a esta expansión.

- ¿Cómo se debe hacer este preprocesamiento?
- ¿Es este método seguro?

RSA en la práctica

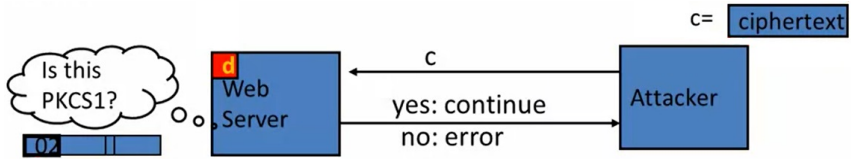
PKCS1: mode 2 = encription, mode 1 = signature.
Es el primero de una familia de estándares denominados *Public-Key Cryptography Standards* (PKCS), publicados por RSA Laboratories. Proporciona las definiciones básicas y las recomendaciones para implementar el algoritmo RSA para la criptografía de clave pública.



- El valor resultante de este *padding* se cifra mediante RSA.
- Ampliamente implementado, por ejemplo, se usa en varios protocolos web.

RSA en la práctica

Ataque a PKCs1: (Bleichenbacher, 1998).



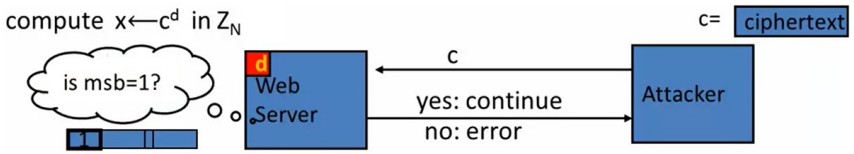
- El atacante intercepta un mensaje cifrado **c**, que quiere descifrar.
- El atacante envía el mensaje **c** directamente al servidor. Este verifica primero si se trata del protocolo PKCS1 (ve los primos dos bytes), y si es "02", continua con el protocolo y lo descifra.

Con esto, el atacante tiene un mecanismo para saber si $msb(\mathbf{c}) = 02$. Si el atacante quiere descifrar un mensaje cifrado **c**:

- Elegir un número aleatorio $\mathbf{r} \in \mathbb{Z}/N\mathbb{Z}$, y calcular $\mathbf{c}' = \mathbf{r}^e \cdot \mathbf{c} (\equiv \mathbf{r}^e\mathbf{m}^e) \pmod{N}$.
- Usar la respuesta que devuelva el servidor ($\mathbf{d}' = (\mathbf{c}')^d = \mathbf{rm} \pmod{N} \Rightarrow \mathbf{m} = \mathbf{r}^{-1}\mathbf{d}' \pmod{N}$).

RSA en la práctica

Para explicar de mejor forma, consideramos una simplificación del ataque anterior.
Ataque: Baby-Bleichenbacher.



Suponga que $N = 2^n$ (este es un módulo inválido, pero nos servirá para explicar).

- Enviar $\mathbf{c}$ revela $\text{msb}(\mathbf{x})$.
- Enviar $2^e \cdot \mathbf{c} \pmod N$ revela $\text{msb}(2\mathbf{x} \pmod N) = \text{msb}_2(\mathbf{x})$.
- Enviar $4^e \cdot \mathbf{c} \pmod N$ revela $\text{msb}(4\mathbf{x} \pmod N) = \text{msb}_3(\mathbf{x})$.
- Enviar $8^e \cdot \mathbf{c} \pmod N$ revela $\text{msb}(8\mathbf{x} \pmod N) = \text{msb}_4(\mathbf{x})$.

Continuando de esta forma, se puede conocer todo $\mathbf{x}$.

RSA en la práctica

Defensas:

Los ataques descubiertos por Bleichenbacher y Klima et al., se pueden evitar mediante un tratamiento y formateo adecuado de los bloques de mensaje.

Podemos hacer lo siguiente:

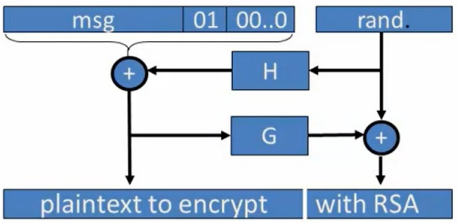
- Generar una cadena R de 46 bytes aleatorios.
- Decriptar el mensaje para recuperar el texto plano **m**.
- Si el padding del PKCS1 no es correcto:
  se establece una pre-master-secret-key $= R$.
  se continúa el protocolo.

En otras palabras, si no es correcto, no se le dice al usuario (o atacante) que no es correcto, o que si la cadena comienza con "02" o no. Smplemente pretendemos que la cadena es un valor aleatorio R.

Defensas

OAEP: Nueva función de pre-proceamiento.

check pad
on decryption.
reject CT if invalid.



Thm [FOPS'01]: RSA is a trap-door permutation $\Rightarrow$
RSA-OAEP is CCA secure when H,G are *random oracles*

in practice: use SHA-256 for H and G

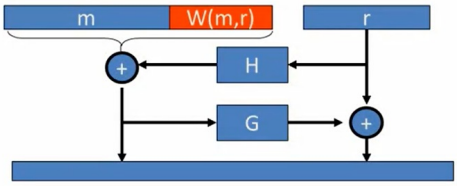
Defensas

Variantes y mejoras: OAEP+

OAEP+: [Shoup'01]

∀ trap-door permutation F
F-OAEP+ is CCA secure when
H,G,W are *random oracles*.

During decryption validate W(m,r) field.



Defensas

Variantes y mejoras: SAEP+

SAEP+: [B'01]

RSA (e=3) is a trap-door perm $\Rightarrow$
RSA-SAEP+ is CCA secure when
H,W are *random oracle*.