RSA - encriptar

Parámetros: $N = pq \approx 1024$ bits, $p, q \approx 512$ bits

1-1 función $RSA(M) = M^e$, e $\gcd(e, \varphi(N)) = 1$

$\varphi(N) = (p-1)(q-1)$

Trapdoor d potencia de desencriptar:

$\Rightarrow ed = 1 \pmod{\varphi(N)}$

Inversión $RSA(M)^d = M^{ed} = M^{k\varphi(N)+1} = M \pmod N$

Problema de RSA calcular $x^{1/e}$

RSA assumption: El problema de RSA es difícil cuando

1) N es grande

2) N es aleatorio

3) $C \neq M \leq N-1$ y también C, entonces es aleatorio

Strong RSA assumption: El problema

dado N y C, calcular cualquier M y e (impar) tq

$M = C^e \pmod N$

es difícil

$$L(f, 1/2, \chi_D) = \frac{k_f \, \ell_{|D|}(f)^2}{\sqrt{N(D)}}$$

$N(D) \approx |D|^2$ probably false

$N(D)$ |

$$\sum 2^{\omega(\phi)} \sim \log(N(D))$$

RSA del texto es inseguro:

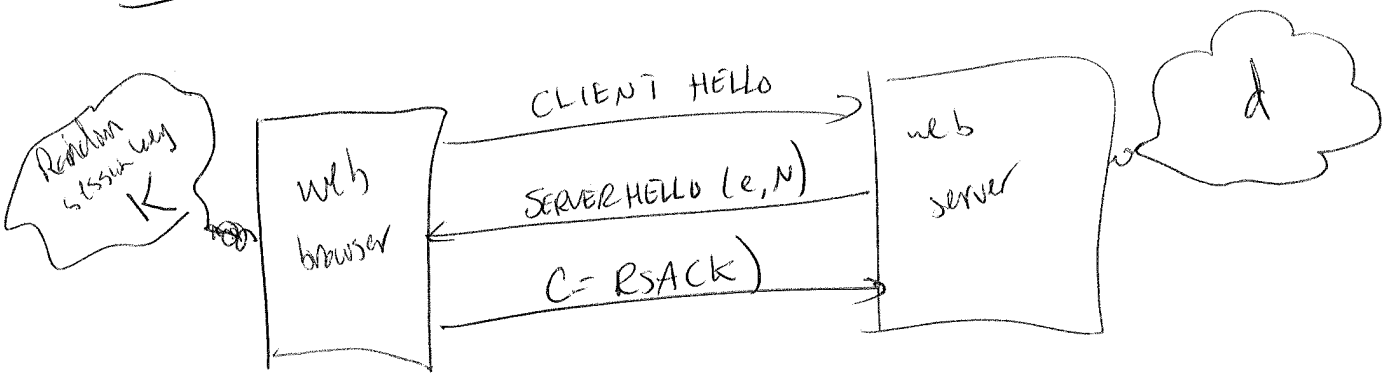
~~RSA~~ textbook RSA:

- clave pública (N, e) Enc: $C = M^e (N)$
- clave privada: d Dec: $C^d = M (N)$

Completamente inseguro

El trapdoor no es un criptosistema.

Un ataque:



- Session key K es ≈ 64 bits, $K \in \{0, 2^{64}-1\}$

  Eve ve: $C = K^e \pmod N$

- Supongamos $K = K_1 \cdot K_2$ donde $K_1, K_2 < 2^{34}$ (prob $\approx 20\%$)

  $\Rightarrow C/K_1^e = K_e^e \pmod N$

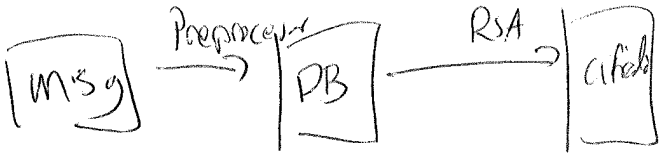
- construir tabla $C/1^e, C/2^e, \ldots, C/2^{34e}$ tiempo 2^{34}

  ver si para $K_2 = 0, \ldots, 2^{34}$ K_2^{*e} está en la tabla tiempo $2^{34} \cdot 34$

- tiempo total $2^{34} + 2^{34} \cdot 34 \approx 2^{40} \approx 2^{64}$

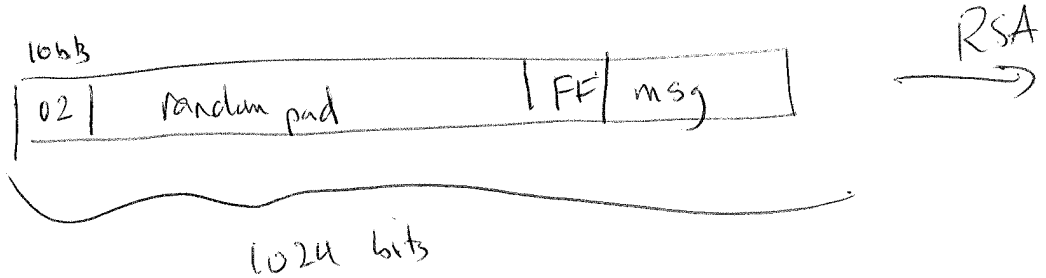
Encriptar con RSA:

- ¡Nunca usar textbook RSA!
- RSA en la práctica:



¿Cómo se hace el preprocesar?

PKCS1 V1.5



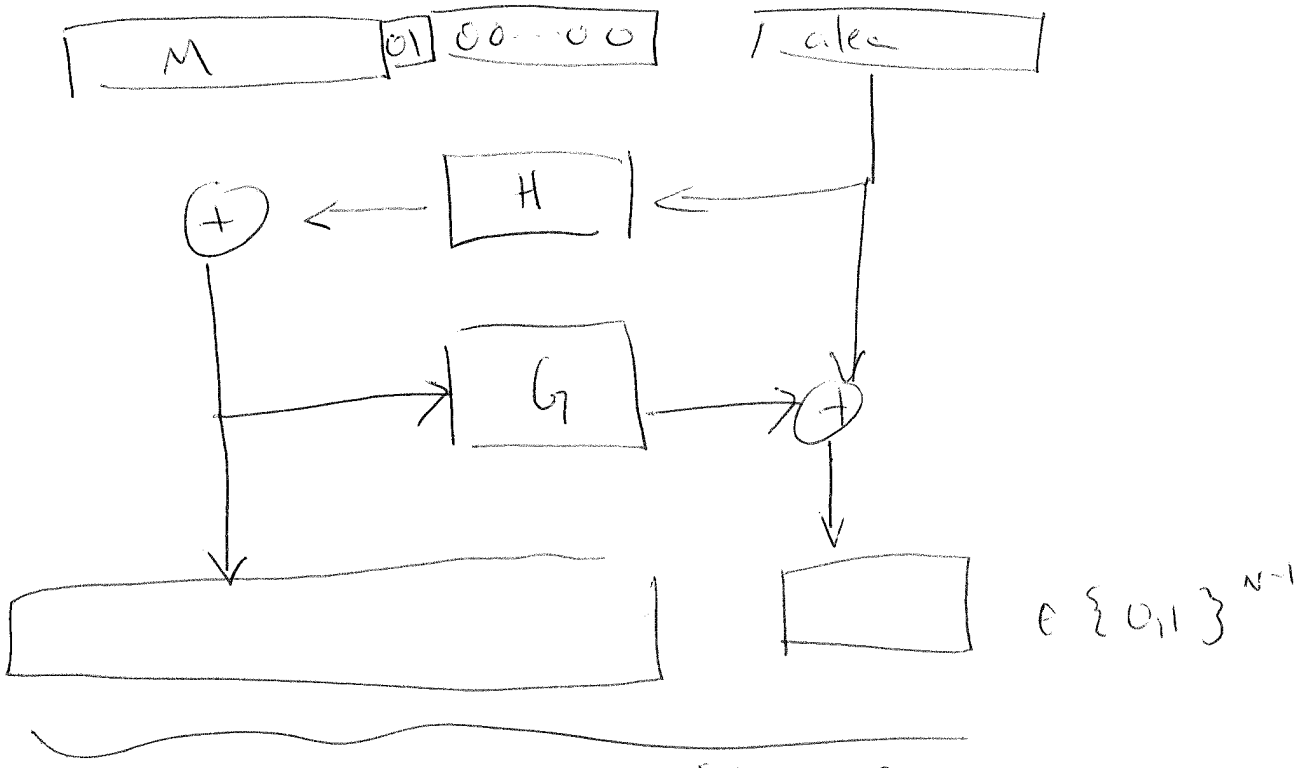
No se ha investigado si este protocolo es seguro aunque se usa en muchos browser y servers.

Ataque contra PKCS1 v 1.5

Bleichenbacher (1998): texto cifrado elegido

Un buen sistema debería resistir un ataque de este tipo

PKCS1 V2.0 OAEP



~~para cada~~ se encripta con RSA

Teor: OAEP es seguro si G y H son oráculos aleatorios

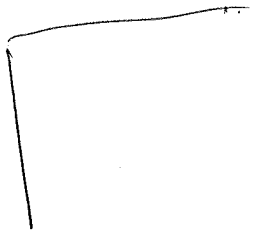
en la práctica se usan funciones hash como SHA-1

Sutileza en la implementación:

ataques de timing

Un sistema debería tener chosen ciphertext security.

Ningún adversario puede ganar este juego.



Las especificaciones:

RSAES-OAEP-Encrypt ((n,e), M, L, Hash, MGF)
mask generating fun.

Input:
(n,e) clave pública, k largo de n en octets

M mensaje Cadena de octets de largo mLen

$mLen \leq k - 2\,hLen - 2$
hLen: largo del Hash

L label : por default ""

Output:
C: una cadena de octets de largo k.

Assert (n,e) válida.

Pasos: 1 Chequear largos:
- Si largo de L es demasiado grande, tirar un error
- Si largo de M es "

2 EME-OAEP
↓
optimal asymmetric encryption padding

a) Si L = "", seguir, si no hashear lo,

lHash = Hash(L) de largo hLen

b) PS (padded string) ~~constante~~ que consiste de

k - mLen - 2hLen - 2 octets de ceros

(puede ser de largo 0)

c) DB = lHash || PS || 0x01 || M

DB → data block

↑ este octet nos deja ~~deducir~~ separar el mensaje del resto

d) generar una cadena ~~de~~ aleatoria de octets ; un seed de largo hLen

e) dbMask = MGF(seed, k - hLen - 1)

f) maskedDB = DB ⊕ dbMask

g) seedMask = MGF(maskedDB, hLen)

h) maskedSeed = seed ⊕ seedMask

i) EM = 0x00 || maskedSeed || maskedDB

EM → encoded message

3) RSA Encrypt:

m = OS2IP(EM)

c = RSAEP((n,e), m)

C = I2OSP(c, k)

↑ largo en octets de nuestro texto cifrado