8636 Criptografía y Seg Informática

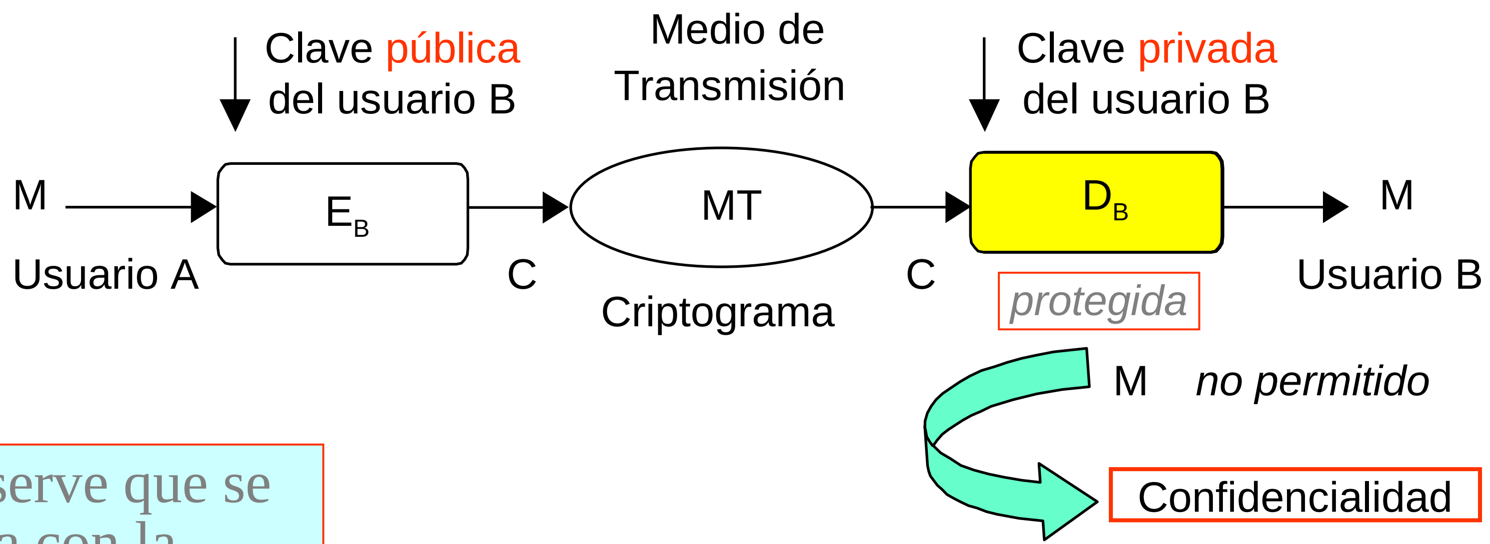
Cifradores Asimétricos

Criptosistemas asimétricos

- Cada usuario crea un par de claves, una privada y otra pública, inversas dentro de un cuerpo finito.
- Lo que se cifra en emisión con una clave, se descifra en recepción con la clave inversa.
- La seguridad del sistema reside en la dificultad computacional de descubrir la clave privada a partir de la pública.

Criptosistemas asimétricos (parte 1)

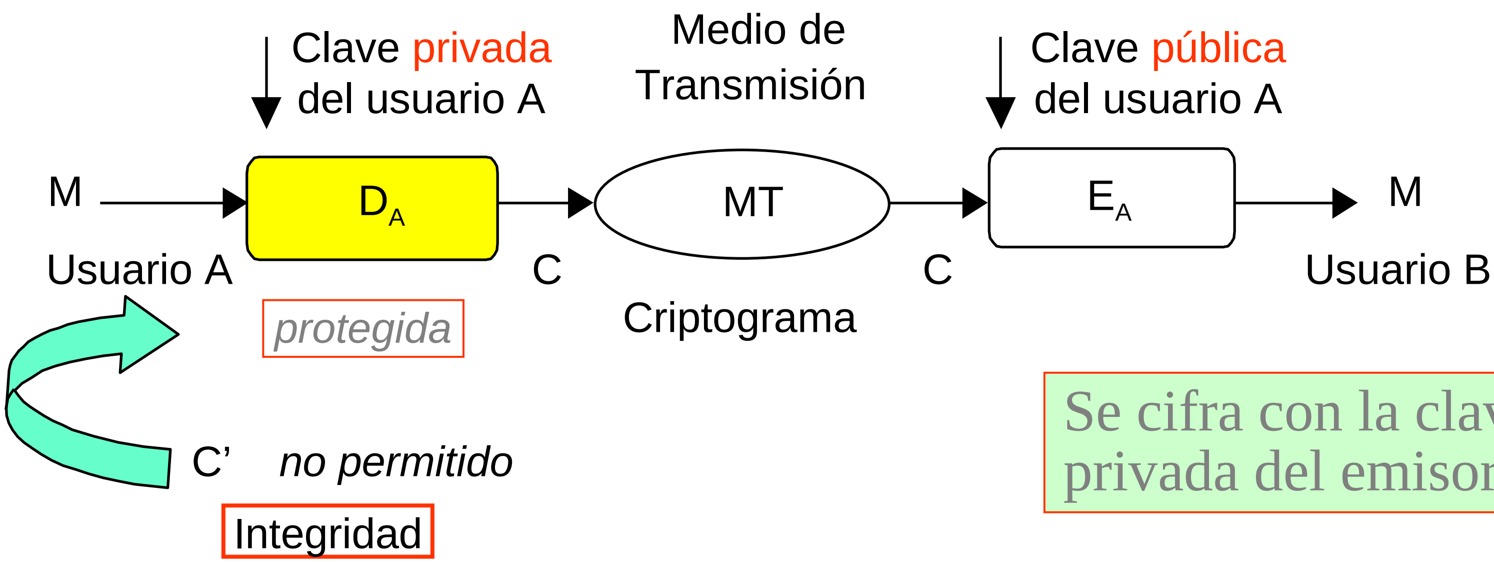
Cifrado con clave pública del receptor



Observe que se cifra con la clave pública del destinatario.

Criptosistemas asimétricos (parte 2)

Cifrado con clave privada del emisor
Firma digital RSA



Se cifra con la clave privada del emisor.

Diffie-Hellman

Sistemas de Clave Pública

Introducción

Pregunta:

¿Es posible que dos partes de una comunicación que nunca han intercambiado información previamente establezcan un canal seguro?

Raiz Primitiva. Definición

Se dice que a es raiz primitiva de p

si las potencias de a generan todos los enteros desde 1 hasta $p-1$

$$\left.\begin{array}{l} a \bmod p \\ a^2 \bmod p \\ \ldots \\ a^{p-1} \bmod p \end{array}\right\}$$

con alguna permutación.

Diffie Hellman

Elementos públicos:

- Un número primo **q**
- Un número **α** tal que α raíz primitiva de q

Generación de claves de usuario:

Usuario A
Selecciona clave privada

$$X_A < q$$

Calcula clave pública

$$Y_A = \alpha^{X_A} \bmod q$$

Usuario B
Selecciona clave privada

$$X_B < q$$

Calcula clave pública

$$Y_B = \alpha^{X_B} \bmod q$$

Diffie Hellman

Intercambio:

El usuario A transmite al usuario B su clave pública

El usuario B le envía su clave pública al usuario A

$$Y_A \longleftrightarrow Y_B$$

Usuario A
Calcula

$$K = (Y_B)^{X_A} \bmod q$$

Usuario B
Calcula

$$K = (Y_A)^{X_B} \bmod q$$

Diffie Hellman

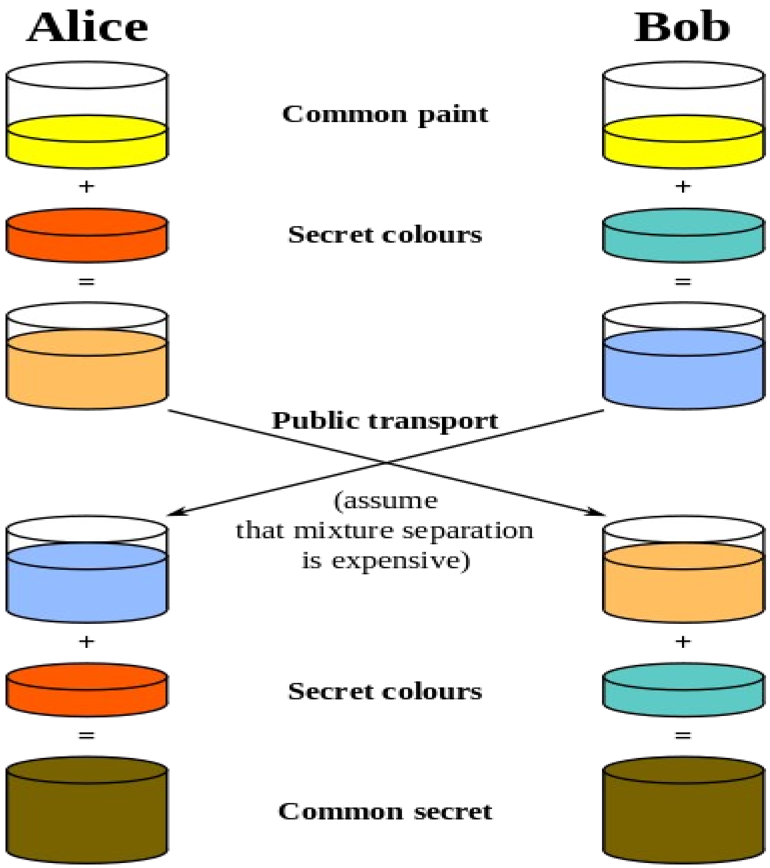
Demostración:

Se verifica que:

$$\left(Y_B\right)^{X_A} \bmod q = \left(\alpha^{X_B}\right)^{X_A} \bmod q = \left(\alpha^{X_B \cdot X_A}\right) = \left(\alpha^{X_A}\right)^{X_B} \bmod q = \left(Y_A\right)^{X_B} \bmod q$$

Y, por lo tanto, ambas claves K resultan idénticas.

Ejemplo Gráfico



http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Ejemplo Numérico

Parámetros públicos:

- $q = 541$
- $\alpha = 10$ (raíz primitiva)

Generación de claves de usuario:

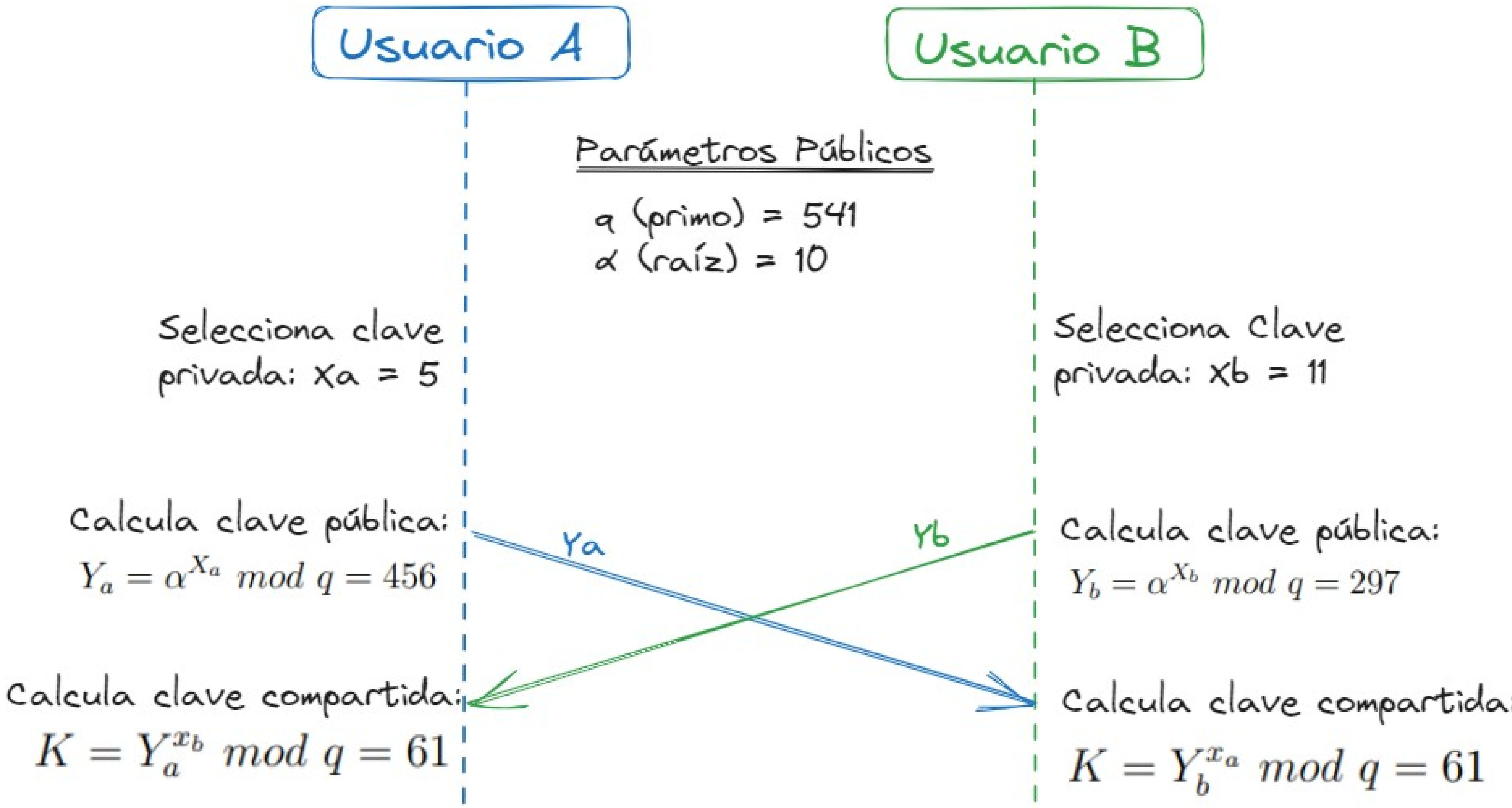
- $Xa = 5$ ➔ $Ya = \alpha^{Xa} \bmod q = 10^5 \bmod 541 = 456$
- $Xb = 11$ ➔ $Yb = \alpha^{Xb} \bmod q = 10^{11} \bmod 541 = 297$

Intercambio:

$key_a = B^{a} \bmod p = 297^5 \bmod 541 = 61$

$key_b = A^{B} \bmod p = 456^{11} \bmod 541 = 61$

Ejemplo Numérico

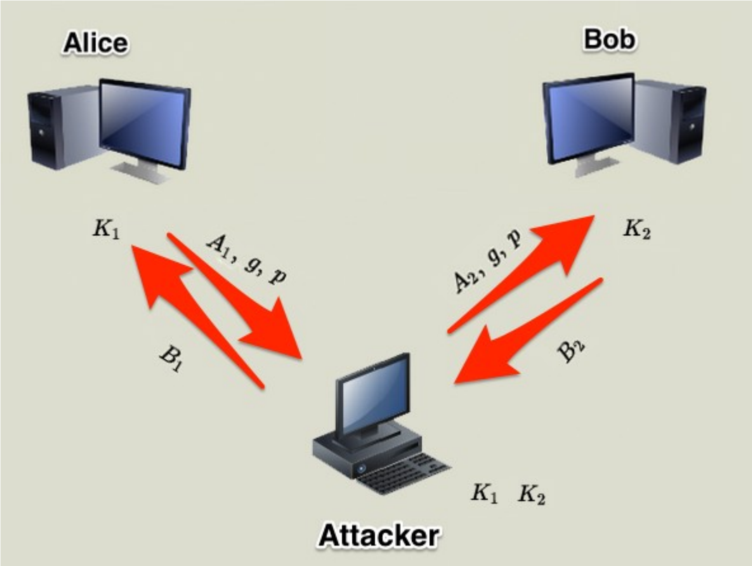


Seguridad del intercambio DH

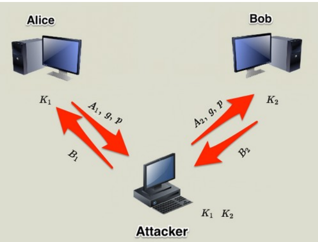
- Radica en la imposibilidad al tener que resolver el problema del logaritmo discreto para encontrar la clave privada que se encuentra en el exponente de la expresión $\alpha^i \bmod p = C$.
- Como p y α serán públicos, al capturar el valor Y el atacante deberá resolver $i = \log_\alpha Y \bmod p$, un problema no polinomial
- Vemos que se envían los parámetros públicos sin autenticar… Que problema puede traer esto?
- El algoritmo es vulnerable ante un ataque del tipo “man in the middle”.

¿Es vulnerable el protocolo de DH?

Man in the midle



¿Es vulnerable el protocolo de DH?



- A elige un número a con 1 < a < p-1 y envía a B α^a mod p
- C intercepta este valor, elige un número c con 1 < c < p-1 y envía a B α^c mod p
- B elige un número b con 1 < b < p-1 y envía a A α^b mod p
- C intercepta este valor y envía a A α^c mod p (valor anterior)
- A y B calculan sus claves $k_A = (\alpha^c)^a$ mod p, $k_B = (\alpha^c)^b$ mod p
- C calcula también las claves:
  - $k_{CA} = (\alpha^a)^c$ mod p
  - $k_{CB} = (\alpha^b)^c$ mod p

¿Cómo se puede solucionar?

Por lo tanto, a partir de ahora C puede interceptar todos los mensajes que se intercambian A y B.

RSA

RSA

- Rivest, Shamir & Adleman del MIT en 1977
- Patentado en 1983
- Estándar
- Usa enteros grandes (ej. 2048 bits, ~10^{300})
- Seguridad: Costo de factorizar números grandes
- Hoy RSA Security, parte de EMC

https://people.csail.mit.edu/rivest/pubs/RSA78.pdf
https://www.youtube.com/watch?v=v4qaxSKTVlA

- Clifford Cocks, un matemático británico que trabajaba para la agencia de inteligencia británica Government Communications Headquarters GCHQ, había descrito un sistema equivalente en un documento interno en 1973 (cinco años antes) que RSA
- su descubrimiento no fue revelado hasta 1997 ya que se trataba de material confidencial (entorno militar), por lo que se supone que Rivest, Shamir y Adleman desarrollaron el algoritmo RSA de forma independiente

RSA: Generación de clave

Seleccionar	p, q	*ambos primos*
Calcular	$n=p*q$	
Calcular	$\phi(n) = \phi(p \ast q) = (p-1)(q-1)$	
Seleccionar la clave publica e		
	$mcd(\phi(n), e) = 1$	$1<e<\phi(n)$
Calcular d	$d=e^{-1} \bmod \phi(n)$	
Clave Publica	$KU = \{e,n\}$	
Clave Privada	$KR = \{d,n\}$	

RSA: Cifrado

CIFRADO	
Texto claro	$M<n$
Cifrado	$C = M^e \bmod n$

DESCIFRADO	
Texto claro	C
Cifrado	$M = C^d \bmod n$

Ejemplo

Seleccionar números primos P y Q:

- $p = 11$
- $q = 23$

Calcular n y Φ(n)

- $n = p * q = 11 * 23 = 253$
- $\Phi(n) = (p-1) * (q-1) = 10 * 22 = 220$

Seleccionar número e / MCD (Φ(n), e) =1

- $e = 3$, MCD (220, 3) = 1

Ejemplo

Calcular exponente privado

- $d = inv(e) \bmod \Phi(n)) = 147$

Clave Pública y Privada

- KU = {e,n} = {3, 253}
- KR = {d,n} = {147, 253}

Cifrado

- M = 18 => C = $18^3 = 5832 \bmod 253 = 13$

Descifrado

- C = 13 => M = $13^{147} \bmod 253 = 18$

Demostración

Teorema de Euler:

- $a^{ø(n)} \bmod N = 1$ donde $gcd(a,N)=1$

En rsa

- $N=p.q$
- $ø(N)=(p-1)(q-1)$
- e y d inversas $\bmod ø(N)$ => $e.d=1+k.ø(N)$
- Entonces:

  $C^d = (M^e)^d = M^{1+k.ø(N)} = M^1.(M^{ø(N)})^k = M^1.(1)^k$

  $= M^1 = M \bmod N$

Elección del valor de la clave pública

¿Qué valor usamos para e?

•la clave pública e deberá ser un valor bajo para que, la clave privada d sea un valor muy alto, cercano a φ(n), y no sea adivinable por fuerza bruta.

•mundialmente se usa el valor F4 (número 4 de Fermat) como clave pública estándar

- F4 = 2^(2^4) + 1 = 2^16 + 1 = 65.537 (un número primo de 17 bits)
- 65.537 en hexadecimal representado con 3 bytes es 0x 010001

about:certificate?cert=MIIFLjCCBBagAwIBAgISA27HsdHjVNrVTnvOaLG2MmoZMA0GCSqGSIb3DQEBCwUAMDIxCzAJBgNVBAYT   mailchimpapp.net

Certificate

campusgrado.fi.uba.ar | R3 | ISRG Root X1

Subject Name

Common Name	campusgrado.fi.uba.ar

Issuer Name

Country	US
Organization	Let's Encrypt
Common Name	R3

Validity

Not Before	Fri, 12 Aug 2022 16:29:14 GMT
Not After	Thu, 10 Nov 2022 16:29:13 GMT

Subject Alt Names

DNS Name	campusgrado.fi.uba.ar

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	BD:15:C5:D9:B7:BE:E2:CC:F6:12:A3:11:60:17:9F:3E:74:2A:CB:94:0A:3D:D9:...

La seguridad

- Se basa en la dificultad computacional de factorizar n que para valores de mil bits se vuelve intratable
- Actualmente n sigue siendo de 2048 bits con p y q de 1024 bits
- Ya se sugiere usar 3 kbits o 4kbits

¿Cómo guardamos la clave privada d?

- Nunca se debería guardar en texto plano
- Habrá que cifrarla de forma local con un algoritmo simétrico.
  - Por ejemplo con AES 256
  - guardarla como un archivo en un dispositivo de almacenamiento (disco rigido, pendrive, etc.)

Casos de uso básicos

Usuario A

- K_{RA}: Clave Privada de A
- K_{UA}: Clave Pública de A

Usuario B

- K_{RB}: Clave Privada de B
- K_{UB}: Clave Pública de B

$$C_{CONF} = E_{K_{UB}}[M]$$

$$C_{AUTH} = E_{K_{RA}}[M]$$

$$C_{CONF+AUTH} = E_{K_{UB}}\left[E_{K_{RA}}[M]\right]$$

Alg. Acelerado para el Calculo de Potencias

A elevado a la K Modulo

n	101	Modulo
a	11	
K	78	

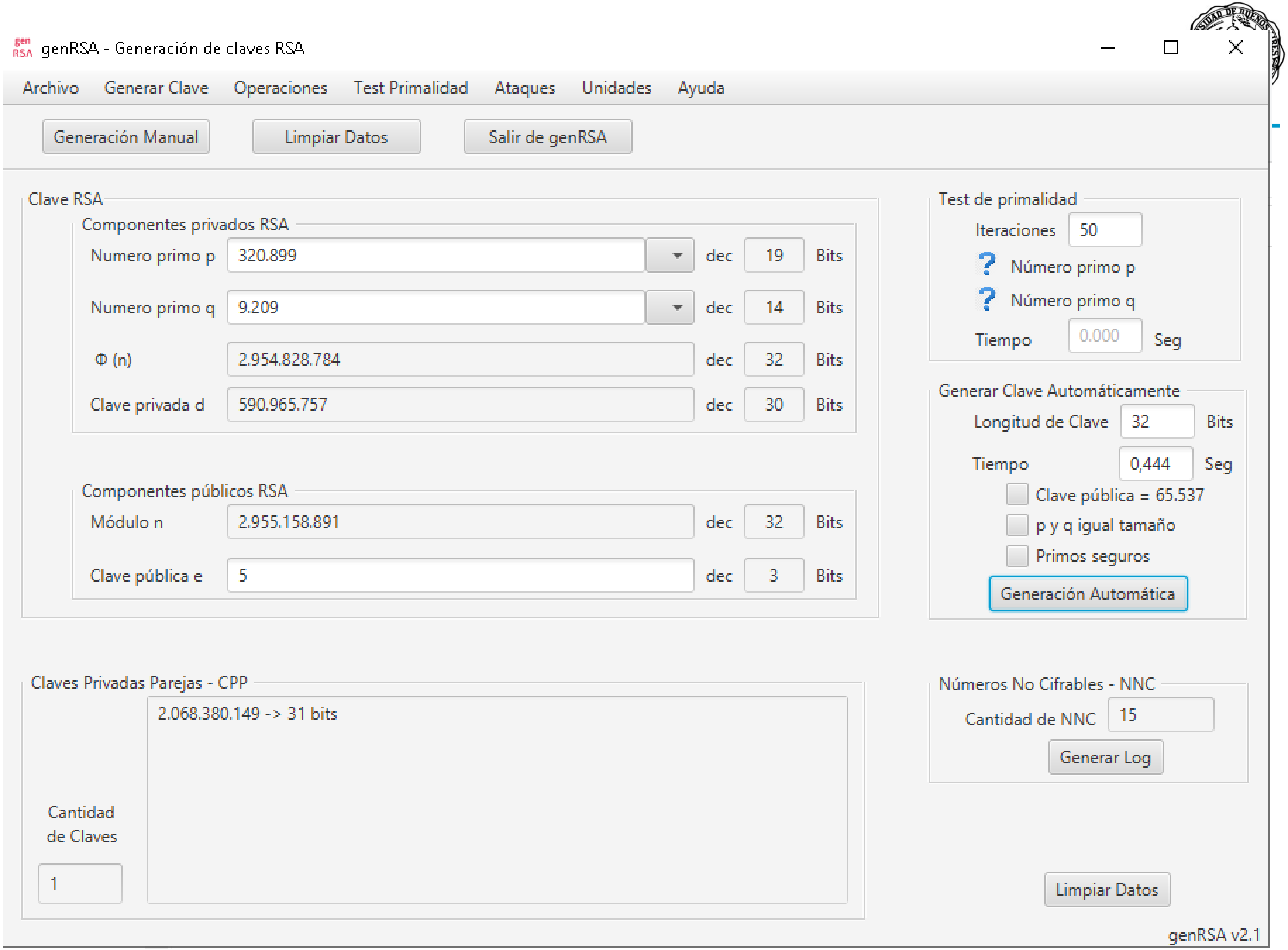
PASO	**A**	**K**	**IMPAR**	1
1	11	78	0	1
2	20	39	1	20
3	97	19	1	21
4	16	9	1	33
5	54	4	0	33
6	88	2	0	33
7	68	1	1	22

a^{2^1} a^{2^2} a^{2^3} a^{2^6}

1. 78 en binario 1001110
2. $A^{78} = a^{2^6} * a^{2^3} * a^{2^2} * a^{2^1}$
3. primero elevo al cuadrado el a y k lo disminuyo a la mitad.

- Si el k es impar multiplico c por el valor de a
- Todas las cuentas modulo n

➔ **11^{78} mod (101) = 22**


genRSA - Generación de claves RSA
Archivo
Generar Clave
Operaciones
Test Primalidad
Ataques
Unidades
Ayuda
Generación Manual
Limpiar Datos
Salir de genRSA
Clave RSA
Componentes privados RSA
Numero primo p
320.899
dec
19
Bits
Numero primo q
9.209
dec
14
Bits
Φ (n)
2.954.828.784
dec
32
Bits
Clave privada d
590.965.757
dec
30
Bits
Componentes públicos RSA
Módulo n
2.955.158.891
dec
32
Bits
Clave pública e
5
dec
3
Bits
Test de primalidad
Iteraciones
50
Número primo p
Número primo q
Tiempo
0.000
Seg
Generar Clave Automáticamente
Longitud de Clave
32
Bits
Tiempo
0,444
Seg
Clave pública = 65.537
p y q igual tamaño
Primos seguros
Generación Automática
Claves Privadas Parejas - CPP
2.068.380.149 -> 31 bits
Cantidad de Claves
1
Números No Cifrables - NNC
Cantidad de NNC
15
Generar Log
Limpiar Datos
genRSA v2.1

Evitando algunas debilidades…

Padding

- Es necesario rellenar de alguna manera el valor de M, para evitar casos triviales como M=0 o M=1.
- OAEP (Optimal Asymetric Encryption Padding) / SAEP+ (Simplified Asymmetric Encryption Padding) / REACT / PSS

Ataques por textos conocidos

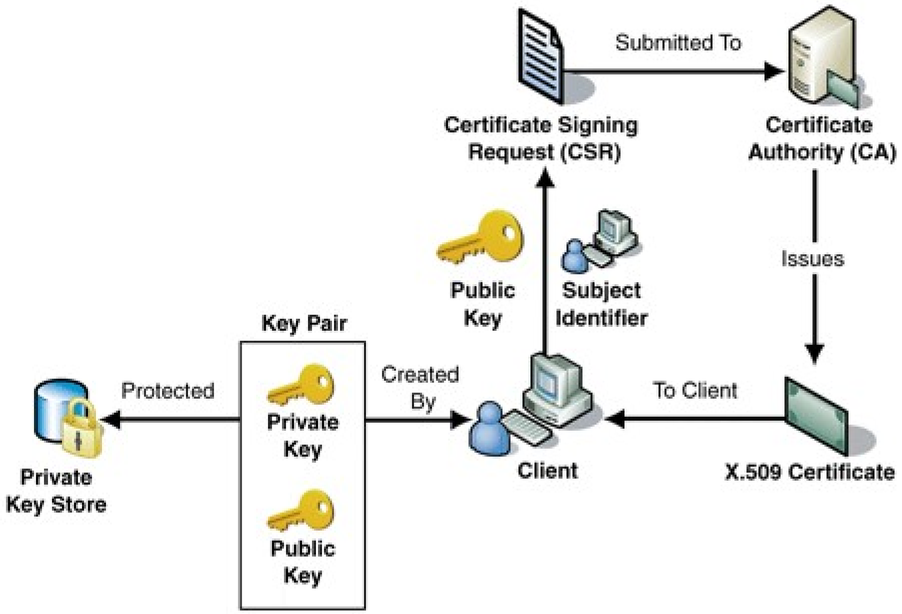
- Dado que para una clave siempre genera el mismo resultado, podría generar un diccionario de textos conocidos M probables, y buscar una colisión.

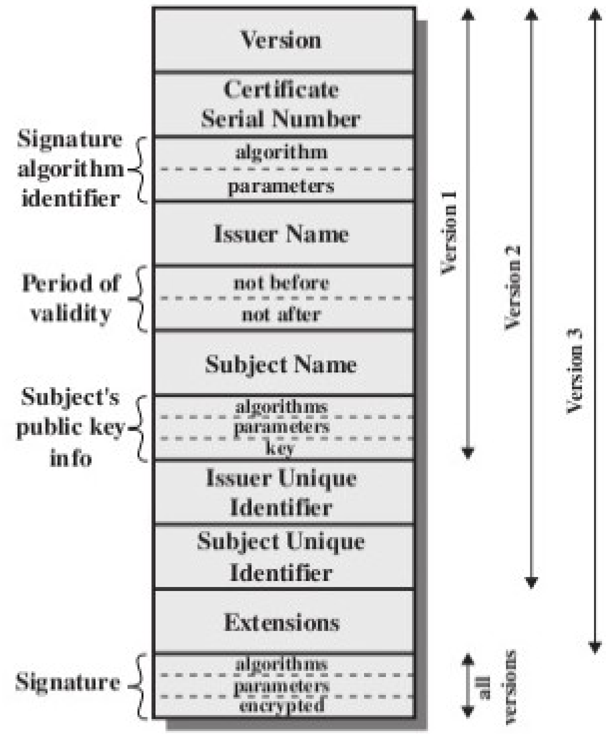
Desafío RSA Labs

Número RSA	Cifras decimales	Cifras binarias	Premio ofrecido	Factorizado en	Factorizado por
RSA-100	100	330		Abril 1991	Arjen K. Lenstra
RSA-129	129	426	$100 USD	Abril 1994	Arjen K. Lenstra et al.
RSA-576	174	576	$10,000 USD	Diciembre 2003	Jens Franke et al., Universidad de Bonn
RSA-640	193	640	$20,000 USD	Noviembre 2005	Jens Franke et al., Universidad de Bonn
RSA-704	212	704	$30,000 USD	*abierto*	
RSA-768	232	768	$50,000 USD	Diciembre 2009	A six-institution research team led by T. Kleinjung
RSA-896	270	896	$75,000 USD	*abierto*	
RSA-1024	309	1024	$100,000 USD	*abierto*	
RSA-1536	463	1536	$150,000 USD	*abierto*	
RSA-2048	617	2048	$200,000 USD	*abierto*	

Casos de uso: Autenticación

Casos de uso: Certificados





(a) X.509 Certificate