UNIVERSIDADE DA CORUÑA

Certificados digitales Autoridades Certificación

LSI - 2016/2017

José Manuel Vázquez Naya
jose@udc.es

Contenidos

- Conceptos de cifrado
  - Cifrado simétrico
  - Cifrado asimétrico
  - Funciones hash
  - Firma Digital
- Certificados digitales
- Autoridades de certificación
- Protocolos seguros

Cifrado

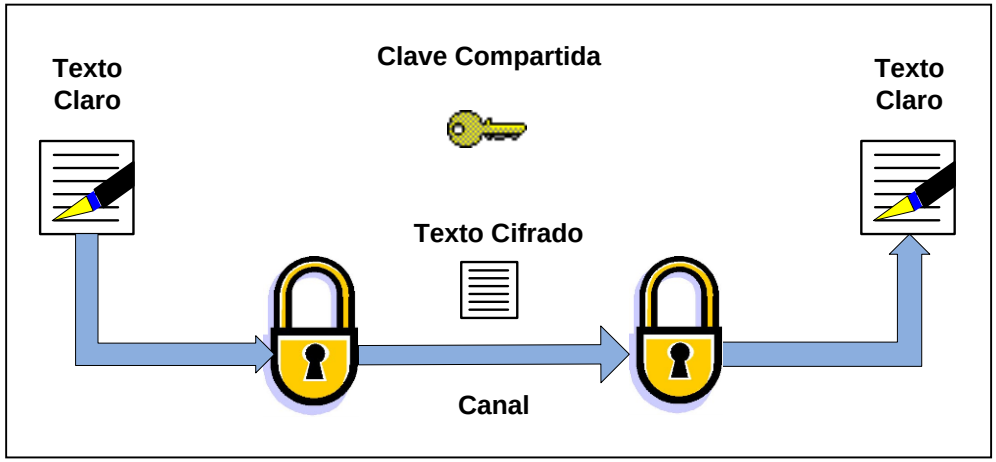
- Criptografía:
  - *Kriptos* (secreto) y *Graphos* (escritura)
  - Forma de escribir ocultando el significado

- Caracterización sistemas de cifrado
  - ¿Cuántas claves se usan para el cifrado/descifrado?
    - Cifrado Simétrico
      - Misma clave se usa para el cifrado y descifrado
    - Cifrado Asimétrico
      - Claves diferentes para el cifrado y el descifrado

CIFRADO SIMÉTRICO

Cifrado simétrico

Cifrado Simétrico



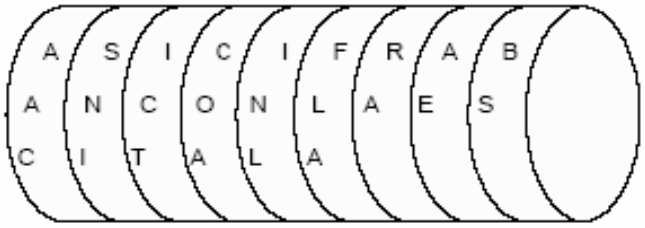
Cifrado simétrico

- **Criptografía clásica**

- Criptografía moderna

Escítala (Scytale)

- Siglo V a.C.
- Pueblo griego de los lacedemonios
- Bastón en el que se enrollaba una cinta de cuero y luego se escribía en ella el mensaje de forma longitudinal




```
M = asicifrabanconlaescitala

C = AACSNIICTCOAINLFLARAAEBS
```

Escítala (Scytale)

- Para descifrar el criptograma y recuperar el mensaje en claro habrá que enrollar dicha cinta en un bastón con el **mismo diámetro** que el usado en el extremo emisor y leer el mensaje de forma longitudinal
- La clave del sistema se encuentra en el diámetro del bastón
- Cifrado por **transposición**

César

- Siglo I a.C.
- Cada carácter en el texto plano se sustituye por el carácter situado 3 posiciones a la derecha en el alfabeto (la A por la D, la B por la E, ...)
  - Desplazamiento, b = 3
- Ejemplo:

```
plain  = a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher = D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Ejemplo:
        M = bomba
        C = ERPED
```

- Para una clave dada, cada letra siempre se cifra igual -> cifrado por **sustitución monoalfabeto**

Cifrado simétrico

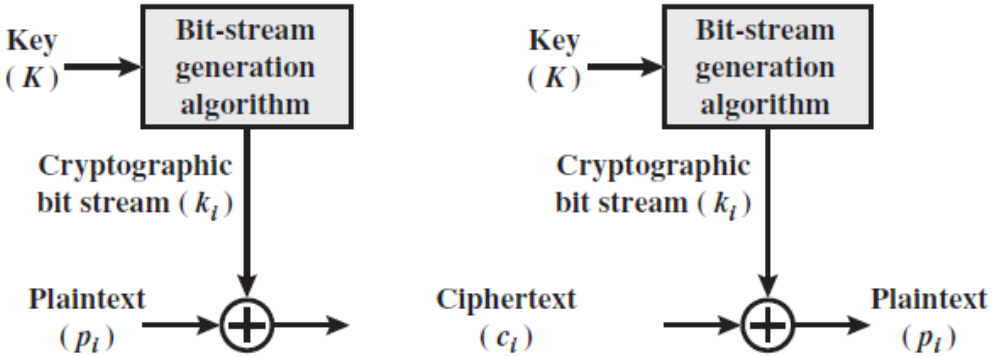
- Criptografía clásica
  - Sustitución
    - Monoalfabeto (ej: César)
    - Polialfabeto (ej: Vigenère, Máquinas de rotor, ...)
  - Transposición (ej: Escítala, Rail Fence, ...)

Cifrado simétrico

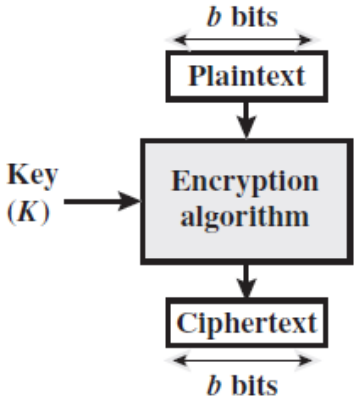
- Criptografía clásica

- **Criptografía moderna**

Cifradores de flujo y cifradores de bloque



(a) Stream cipher using algorithmic bit-stream generator



(b) Block cipher

Figure 3.1 Stream Cipher and Block Cipher

Extraído de (Stallings, 2011)

Data Encryption Stantard (DES)

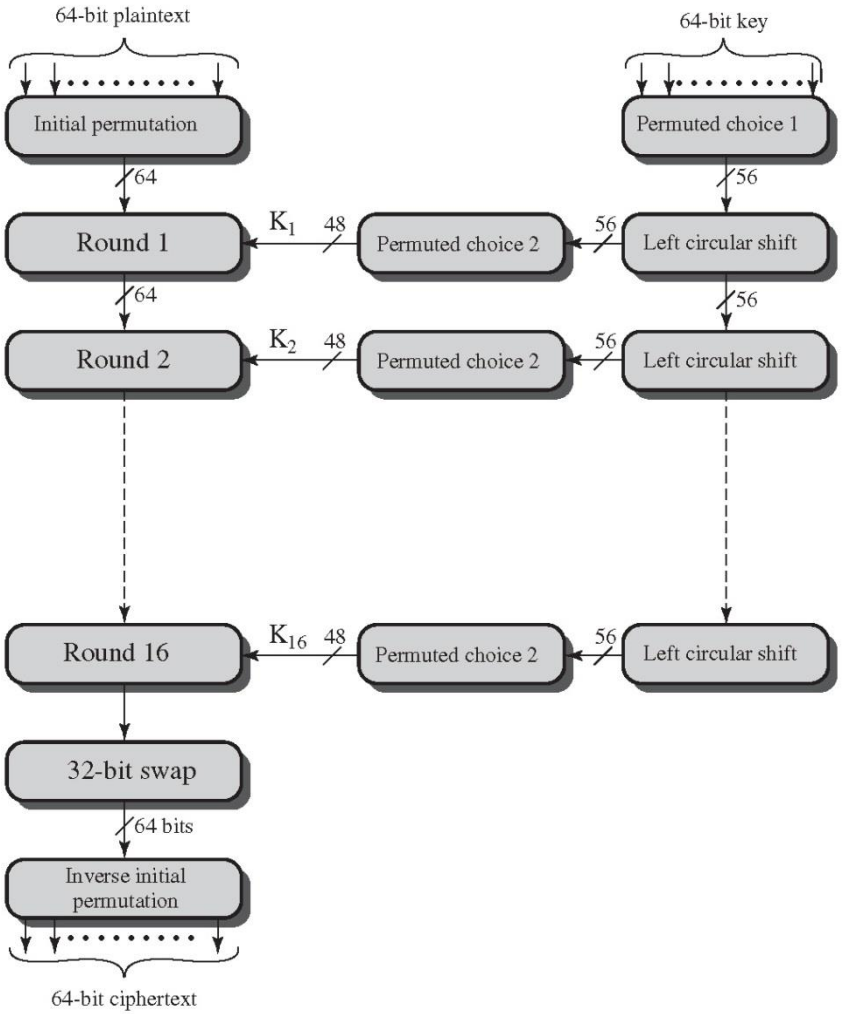


Figure 3.5 General Depiction of DES Encryption Algorithm

Data Encryption Stantard (DES)

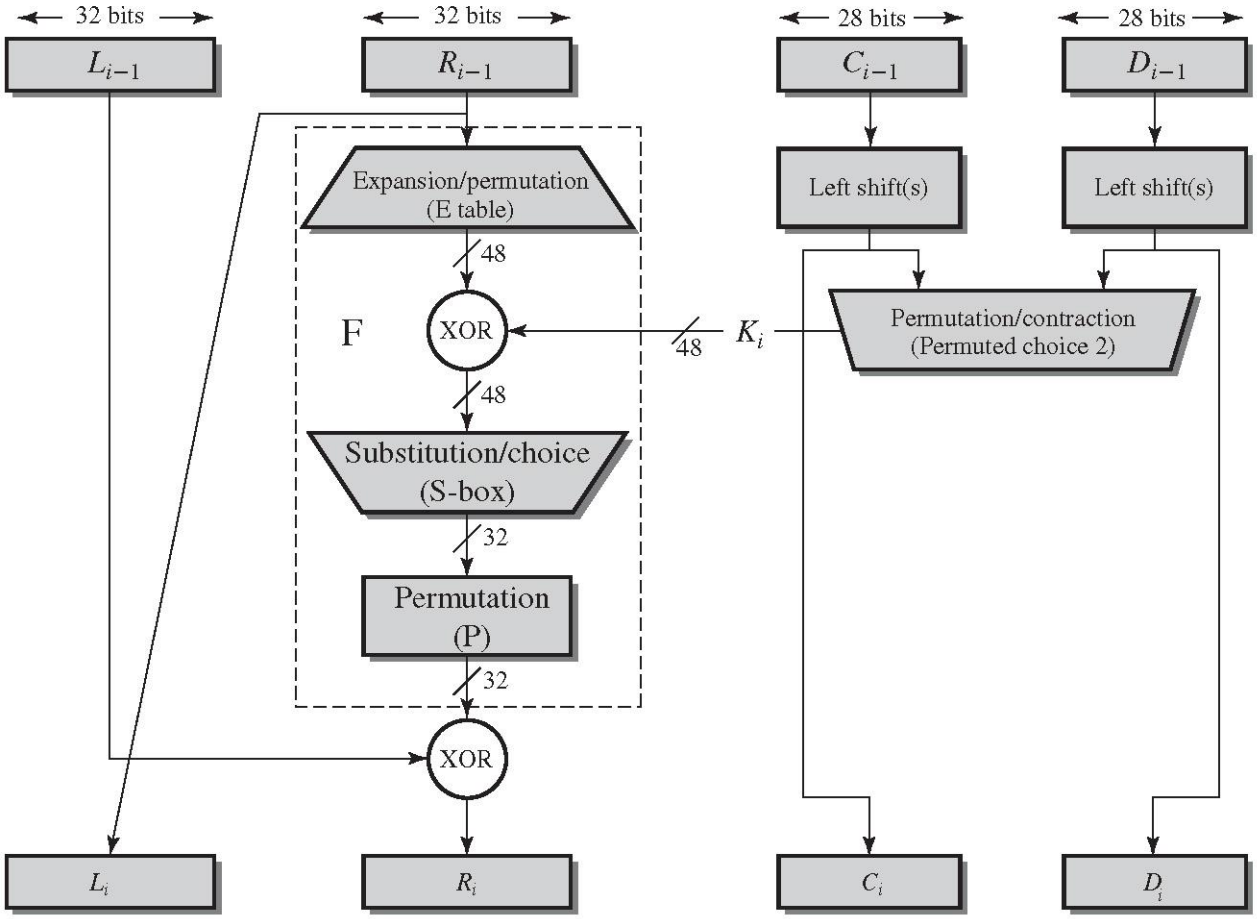


Figure 3.6 Single Round of DES Algorithm

Cifrado simétrico

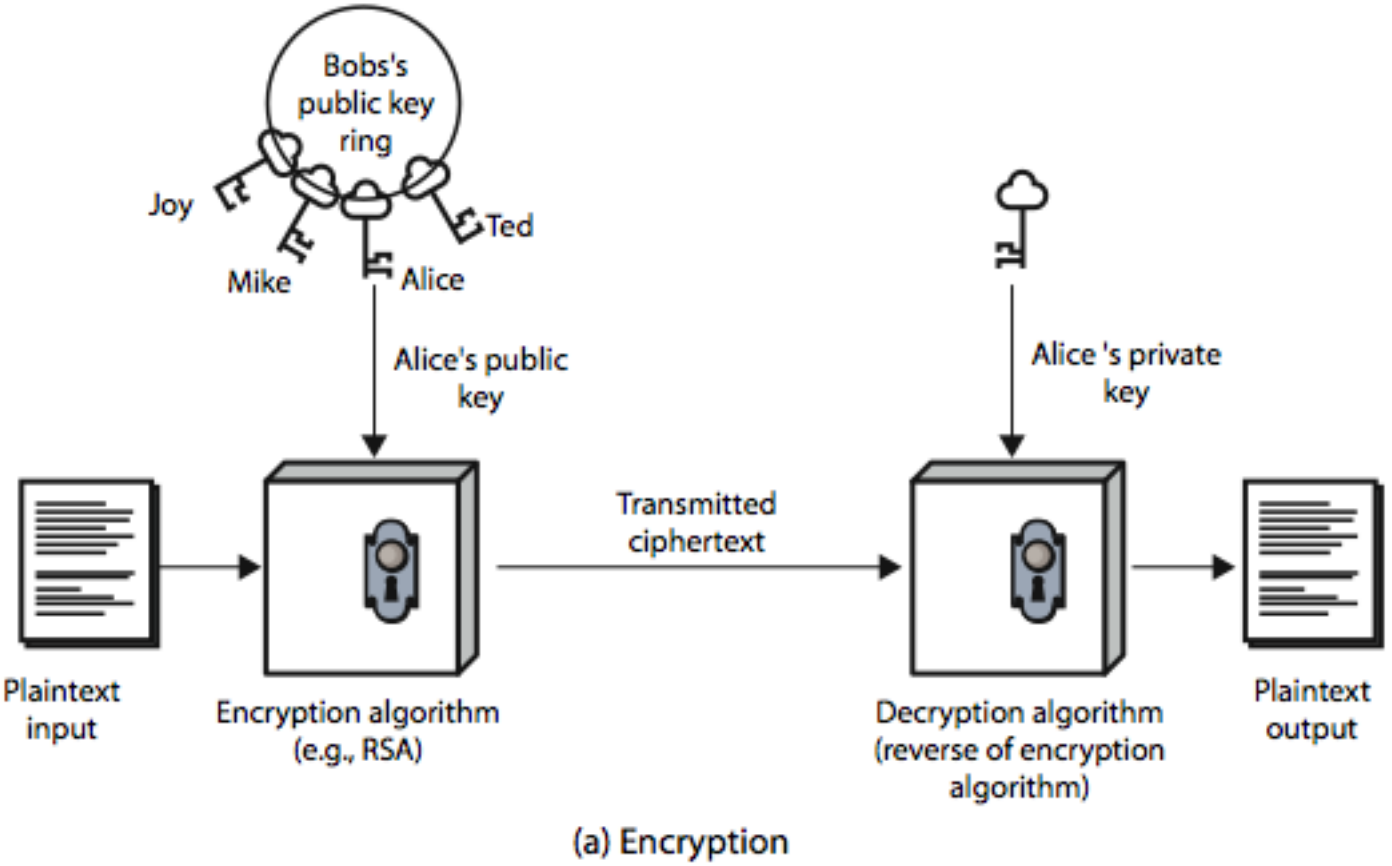
- Criptografía moderna
  - Cifradores de bloque
    - Varias rondas de procesamiento
    - En cada ronda se efectúan operaciones de sustitución y transposición
    - ej: DES, 3DES, AES, ...
  - Cifradores de flujo
    - ej: RC4, Turing, SEAL, ...

CIFRADO ASIMÉTRICO

Cifrado asimétrico

- Dos claves:
  - Pública (K_{Ua})
    - Conocida por todo el mundo
    - Usada para cifrar mensajes y verificar la firma de un mensaje
  - Privada (K_{RA})
    - Conocida únicamente por el propietario
    - Usada para descifrar mensajes y para firmar mensajes
- Asimétrica: las partes no son iguales
- Funcionamiento basado la Teoría de Números

Cifrado asimétrico



Cifrado con clave pública de destino

- Sólo destinatario podrá descifrar el mensaje (PR_B)

- Proporciona:
  - *confidencialidad*

Cifrado con clave privada de origen

- Cualquier usuario podrá descifrar el mensaje (PU_A)
- No proporciona *confidencialidad*
- Sí proporciona:
  - *integridad*
    - si el mensaje es alterado no se podrá descifrar
  - *autenticidad del emisor*
    - Sólo el emisor puede haber cifrado el mensaje con su clave privada, ya que sólo él tiene esa clave
  - *no repudio*
    - el emisor no puede negar que ha sido el quien cifró el mensaje
- ¡Es el mecanismo que hace posible la **firma digital**!

Entonces…
¿ya no necesitamos
el cifrado simétrico?

Principales algoritmos de clave pública

- RSA
  - Ronald Rivest, Adi Shamir y Leonard Adleman.
  - Cifrado, intercambio de claves y firma digital.

- Diffie-Hellman
  - Whitfield Diffie y Martin Hellman.
  - Intercambio de claves.

El algoritmo de clave pública RSA

- Desarrollado por Rivest, Shamir & Adleman en el MIT en **1977**. Se publicó en 1978.
  - Ha sido desde entonces el enfoque más aceptado e implementado para el cifrado de clave pública.
- Es un **cifrado de bloque** en el que el texto claro y el texto cifrado son enteros entre 0 y n-1 para algún n.
  - Tamaño típico de n: 1024 bits (309 dígitos decimales)
- Basado en exponenciación.
- Seguridad basada en el coste de factorizar números grandes.

Generación de claves RSA

- Seleccionar dos números primos grandes, p y q.
- Calcular **n** = p·q
- Calcular φ (n)=(p-1)(q-1)
- Seleccionar clave de cifrado **e**

  1<e< φ(n), mcd(e, φ(n))=1
- Obtener clave de descifrado **d**

  e·d=1 mod φ(n) y 0≤d≤n

  d se puede calcular usando el alg. Euclides extendido
- Clave Pública: PU={**e**, **n**}
- Clave Privada: PR={**d**, **n**}
- Guardar en secreto o destruir p, q y φ(n)

- **p=17** y **q=11**
  (estos no son grandes ;-)
- **n =** p·q = 17×11 = **187**
- **φ(n) =** 16×10 = **160**
- **e = 7** ya que

  mcd(7, 160) = 1
- **d = 23** ya que

  23 · 7 = 161 = (1 · 160) + 1
- PU={**7**, **187**}
- PR={**23**, **187**}

Cifrado y Descifrado en RSA

- Cifrado
  - Texto plano: $M < n$
  - Texto cifrado: $C = M^e \bmod n$

Recordar:

PU={**e**, **n**}
PR={**d**, **n**}

- Descifrado
  - Texto cifrado: C
  - Texto plano: $M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

- Requisitos
  - $\exists$ e, d, n / $M^{ed} \bmod n = M \quad \forall M < n$
  - $(M^e \bmod n)$ y $(C^d \bmod n)$ sean fáciles de calcular
  - Conocidos e y n sea imposible calcular d

Cifrado y Descifrado en RSA

- Un ejemplo:

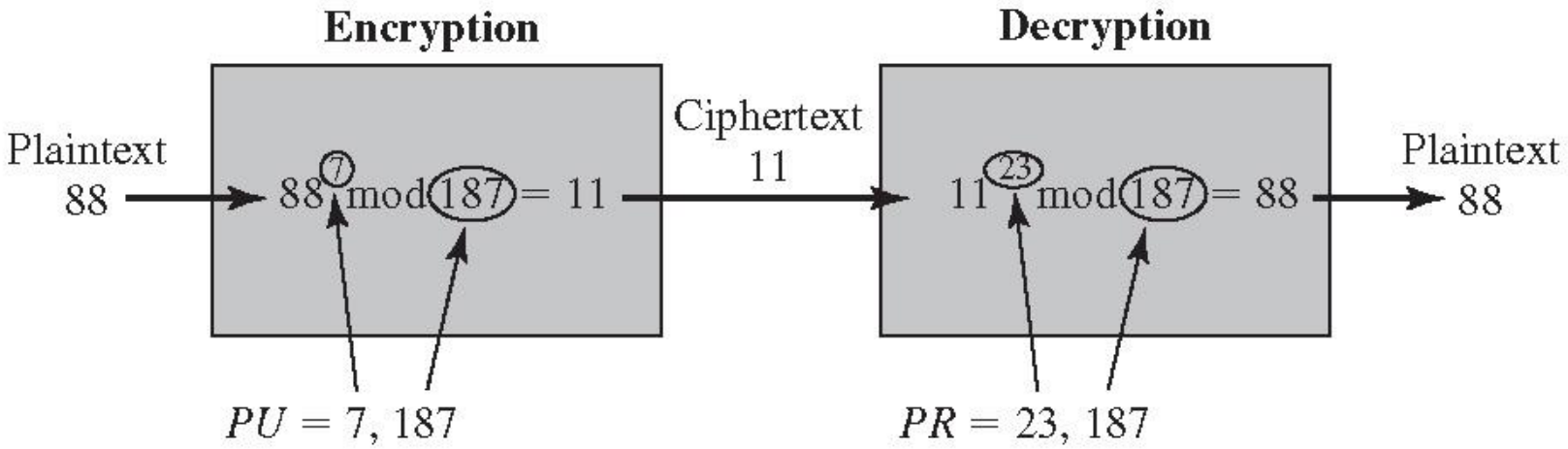


Figure 9.6 Example of RSA Algorithm

Intercambio de claves Diffie-Hellman

- Primer algoritmo de clave pública publicado (Diffie & Hellman, 1976)
- Muchos productos comerciales utilizan este algoritmo
- El propósito del algoritmo es permitir a dos usuarios intercambiar una clave secreta de forma segura que luego pueda ser usada para el cifrado posterior de mensajes.
- Se usa en protocolos como SSH, SSL, TLS, …
- Su seguridad radica en la extrema dificultad (conjeturada, no demostrada) de calcular logaritmos discretos en un cuerpo finito

Conceptos previos

- Problema del **logaritmo discreto**

Fácil →

$3^{29} \bmod 17 = 12$

Difícil ←

$3^{?} \bmod 17 = 12$

$? = \text{dlog}_{3,17}(12)$

Conceptos previos

- Logaritmo discreto

$$b = \mathrm{dlog}_{g,p}(B)$$

  - Logaritmo discreto de B en base g módulo p es el número al que hay que elevar g para obtener B módulo p, es decir, es el número que cumple

$$B = g^b \bmod p$$

Diffie-Hellman Key Exchange

- Parámetros públicos:
  - Número primo: **p** (ej: 23)
  - Generador: **g** (ej: 5)
    - Número natural
    - Preferiblemente distinto de 0 y 1
    - g < p
    - raíz primitiva módulo p

Diffie-Hellman Key Exchange

- Alice elije su secreto: a
  - 1 < a < p (ej. 8)

- Bob elije su secreto: b
  - 1 < b < p (ej. 5)

Diffie-Hellman Key Exchange

1. Alice genera su clave compartida: A

$$A = g^a \bmod p$$

Y se la envía a Bob

$$5^8 \bmod 23 = 16$$

2. Bob genera su clave compartida : B

$$B = g^b \bmod p$$

Y se la envía a Alice

$$5^5 \bmod 23 = 20$$

3. Alice recibe B y calcula la clave de sesión: S

- $S = B^a \bmod p$

$$20^8 \bmod 23 = 6$$

4. Bob recibe A y calcula la clave de sesión: S

- $S = A^b \bmod p$

$$16^5 \bmod 23 = 6$$

Ambos obtienen el mismo valor sin haberlo enviado por la red!

Diffie-Hellman Key Exchange

- ¿Por qué funciona?

S (calculado en Alice) $= B^a \pmod p$

$= g^{b^a} \pmod p$

$= g^{(ba)} \pmod p$

$= g^{(ab)} \pmod p$

$= g^{a^b} \pmod p$

$= A^b \pmod p = S$ (calculado en Bob)

- Ningún intermediario puede conocer la clave de sesión S, puesto que desconoce los secretos a y b

Diffie-Hellman Key Exchange

- Un adversario, para calcular el secreto de Bob, b, tiene

  p, g, *A* y *B (no son datos secretos)*

- y sabe que

  $B = g^b \bmod p$

- luego tendría que resolver:

  $b = \text{dlog}_{g,p}(B)$

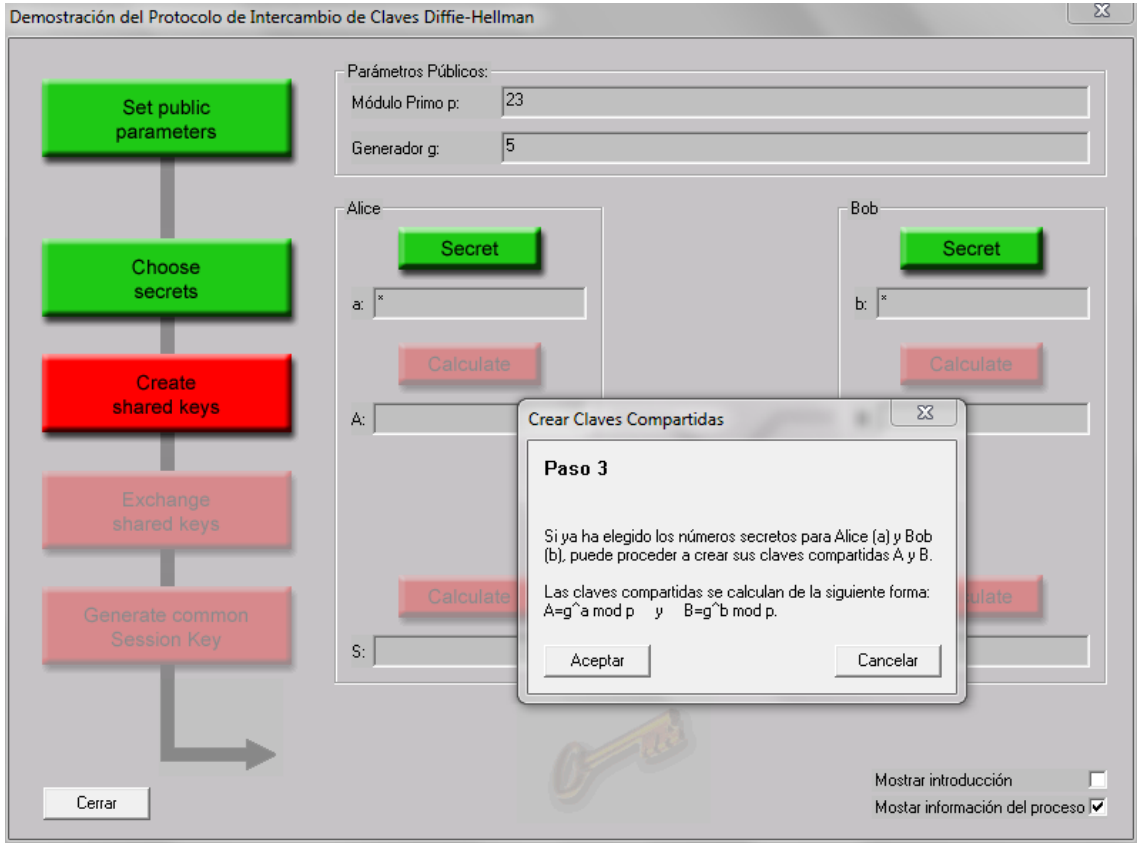
- Pero, mientras es relativamente sencillo calcular exponenciales módulo un primo, es muy difícil calcular logaritmos discretos. Para números primos grandes se considera imposible.

Diffie-Hellman

- Un intercambio de claves con este protocolo es vulnerable a ataques MitM
  - Esto se debe a que D-H no autentica a los participantes
  - Una posible solución es incluir el uso de firma digital y certificados de clave pública

Diffie-Hellman

- Cryptool > Procedimientos Indiv. > Protocolos > Demostración Diffie-Hellman.



FUNCIONES HASH Y FIRMA DIGITAL

Funciones hash

- Una función hash acepta un mensaje de tamaño variable, M, como entrada y produce un resumen del mensaje de tamaño fijo H(M) como salida.

- Usos:
  - Integridad de archivos
  - Contraseñas
  - ...

Funciones Hash. Una aproximación

- Agrupación de texto en bloques
  - Representación de cada carácter por su código ASCII
  - Tamaño bloque: 3
- Función matemática sobre elementos del bloque
  - P. ej.: (A – B) * C
  - Primer Bloque: (69 – 110) * 32 = -1312
- Valor Hash a partir de valores parciales
  - Ejemplo: suma de todos los resultados intermedios

E	n		u	n		r	i	n	c	ó	n		d	e	
69	110	32	117	110	32	114	105	110	99	243	110	32	100	101	
		-1312			224			990			-15840			-6868	-22806

	l	a		M	a	n	c	h	a		d	e		c	
32	108	97	32	77	97	110	99	104	97	32	100	101	32	99	
		-7372			-4365			1144			6500			6831	2738

u	y	o		n	o	m	b	r	e		n	o		q	
117	121	111	32	110	111	109	98	114	101	32	110	111	32	113	
		-444			-8658			1254			7590			8927	8669
															-11399

Funciones Hash. Una aproximación

- Cualquier cambio mínimo en el texto produce un cambio radical en el resultado de la función Hash
  - Así, si cambiamos rincón por rincon, el valor de la función Hash pasa de -11.399 a 3.121

E	n		u	n		r	i	n	c	o	n		d	e	
69	110	32	117	110	32	114	105	110	99	111	110	32	100	101	
		-1312			224			990			-1320			-6868	-8286

	l	a		M	a	n	c	h	a		d	e		c	
32	108	97	32	77	97	110	99	104	97	32	100	101	32	99	
		-7372			-4365			1144			6500			6831	2738

u	y	o		n	o	m	b	r	e		n	o		q	
117	121	111	32	110	111	109	98	114	101	32	110	111	32	113	
		-444			-8658			1254			7590			8927	8669
															3121

Una función hash simple

- Para generar un hash de n bits, se puede dividir el archivo a procesar en m bloques de n bits cada uno y calcular el XOR de dichos bloques. El resultado será el hash del archivo.

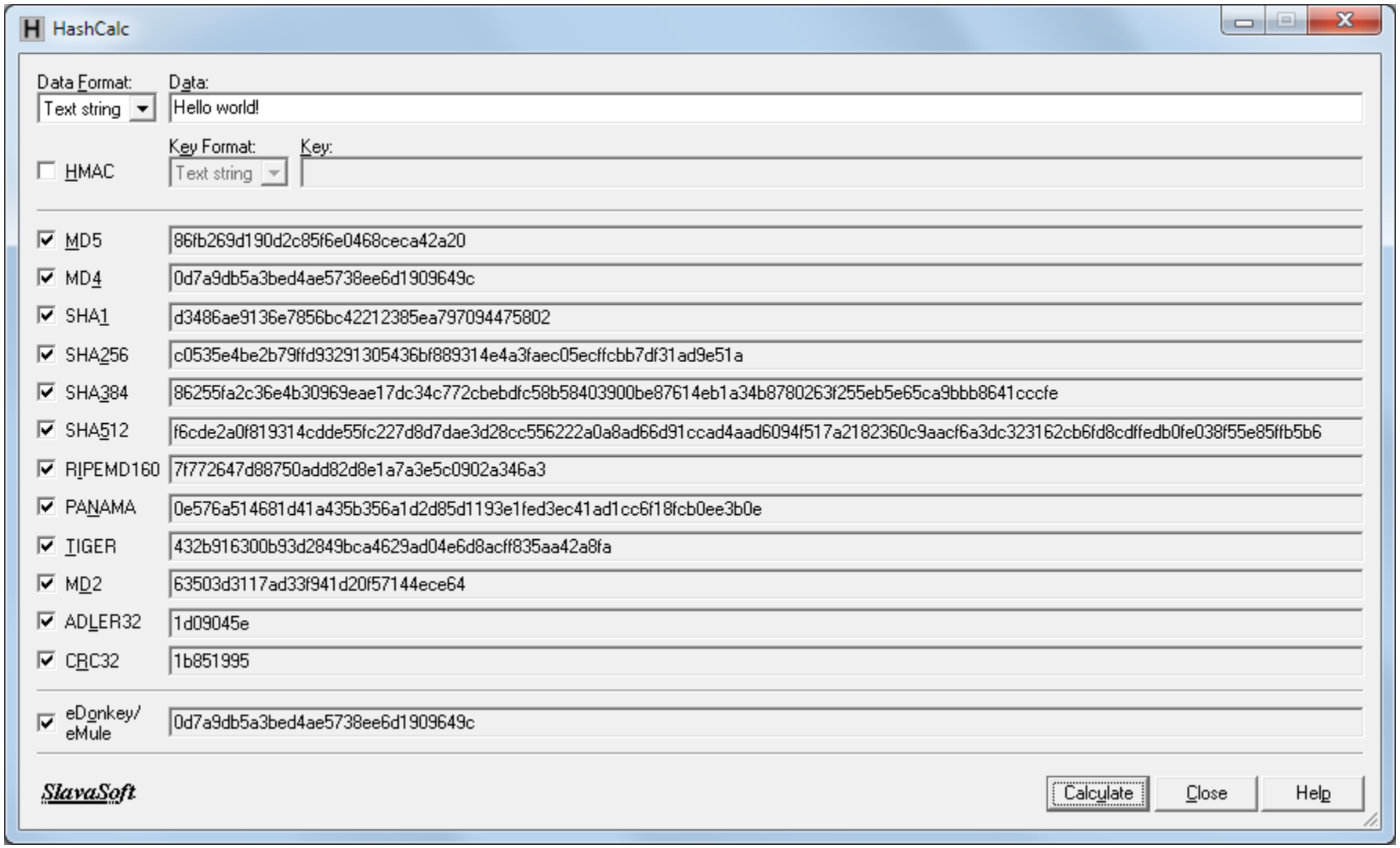
	Bit 1	Bit 2	• • •	Bit *n*
Bloque 1	b_{11}	b_{21}		b_{n1}
Bloque 2	b_{12}	b_{22}		b_{n2}
	•	•	•	•
	•	•	•	•
	•	•	•	•
Bloque *m*	b_{1m}	b_{2m}		b_{nm}
Código *hash*	C_1	C_2		C_n

Función hash simple mediante XOR bit a bit (Stallings, 2004)

Requisitos de las funciones hash

- Para que resulte útil a la autenticación de mensajes, una función hash H debe poseer las siguientes propiedades (Stallings, 2011):
  - H puede aplicarse a un bloque de datos de cualquier tamaño
  - H produce una salida de tamaño fijo
  - H(x) es relativamente fácil de computar para cualquier x dado
  - Para cualquier valor h dado, es imposible desde el punto de vista computacional encontrar x tal que H(x)=h (**propiedad unidireccional**)
  - Para cualquier bloque dado x, es imposible desde el punto de vista computacional, encontrar $y \neq x$ con $H(y) = H(x)$ (**resistencia débil a la colisión**)
  - Es imposible desde el punto de vista computacional encontrar un par (x, y) tal que $H(x) = H(y)$ (**resistencia fuerte a la colisión**)

Funciones hash



Firma digital

- 1ª aproximación:
  - Documento se cifra con PR_{orig}
    - Sólo el poseedor de la clave privada puede haberlo hecho
    - PU_{orig} permite comprobar la validez del documento
  - Problema: ineficiencia cifrado
    - Cifrado asimétrico tiene un coste computacional muy alto

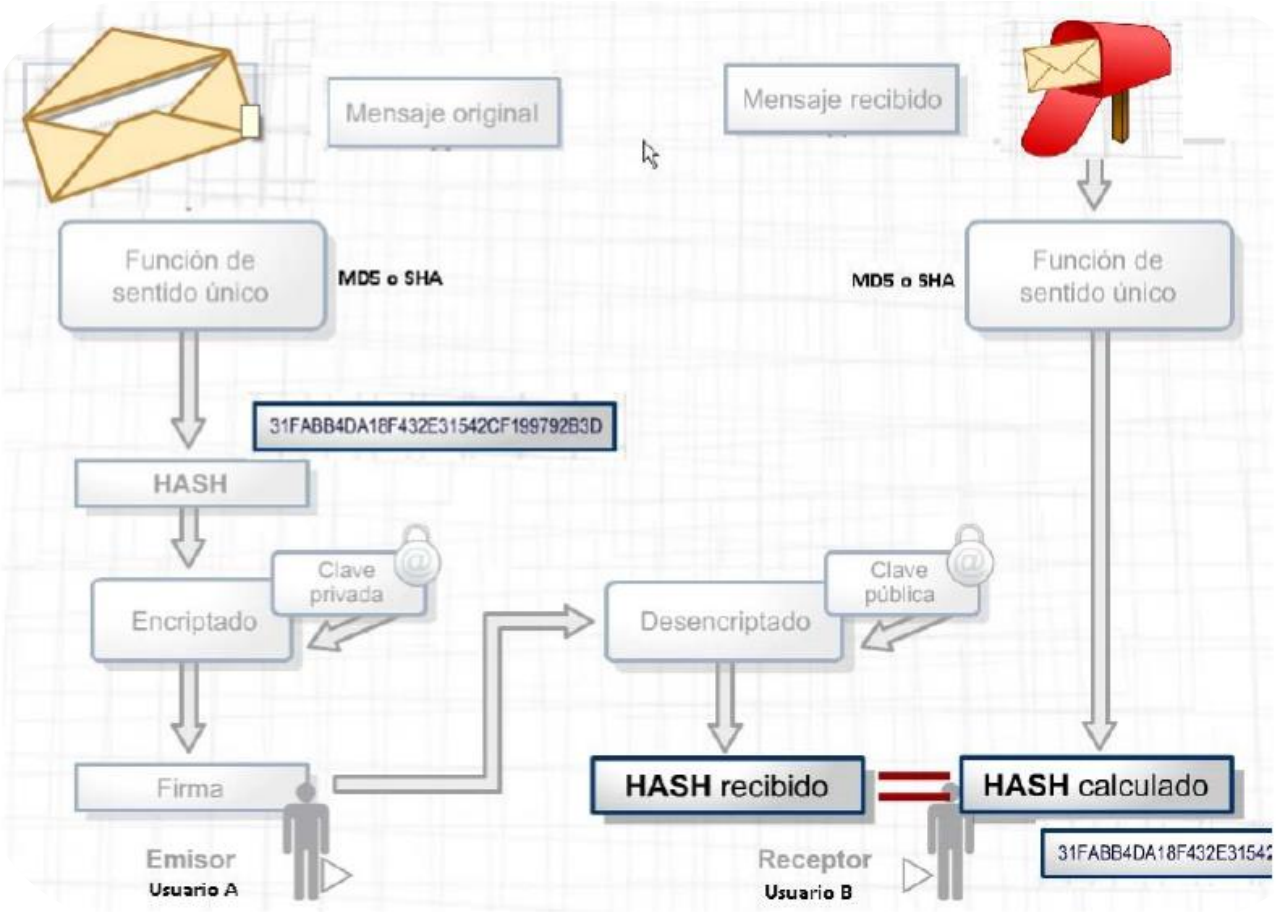
Firma digital

- 2ª aproximación (real):
  - Se calcula un resumen (hash) del documento
  - Se cifra dicho resumen con PR_{orig}
    - Dadas las propiedades de las funciones hash, esto es equivalente a la cifra del documento original
    - Ventajas
      - Más rápido (los resúmenes tienen tamaño fijo y pequeño)
      - No es necesario aplicar ninguna función para acceder al texto original

Firma digital

- Protocolo:
  1. Generar resumen del documento utilizando una función conocida (MD5, SHA-1, SHA-2, ...)
  2. Cifrar resumen con clave privada emisor
  3. Enviar documento junto con resumen firmado al receptor
  4. Receptor genera un resumen del documento recibido, usando la misma función que el origen
  5. Receptor descifra con la clave pública del emisor el resumen cifrado (que se adjuntó con el mensaje)
  6. Si los resúmenes obtenidos en los pasos 4 y 5 coinciden, la firma es válida

Firma Digital
Protocolo



Firma digital

- Se ofrecen conjuntamente los servicios de:
  - **Autenticación**, ya que si el documento viene firmado por A, podemos estar seguros de su identidad, dado que sólo él ha podido firmarlo
  - **Integridad** del documento, ya que en caso de ser modificado, resultaría imposible hacerlo de forma tal que se generase el mismo resumen que había sido firmado
  - **No repudio**, ya que nadie excepto A podría haber firmado el documento
  - **No ofrece privacidad**!

Recursos de interés

- Intypedia:
  - Lección 14. Funciones Unidireccionales y algoritmos de hash. http://www.criptored.upm.es/intypedia/video.php?id=introduccion-funciones-hash&lang=es

CERTIFICADOS DIGITALES Y AUTORIDADES DE CERTIFICACIÓN

Certificados digitales

- Una de las funciones principales del cifrado de clave pública es la de tratar el problema de la distribución de claves
- Un usuario puede enviar su clave pública a otro o difundirla a través de Internet
- Problema: cualquiera puede falsificar la clave pública
- Solución: que una tercera parte de confianza firme digitalmente la clave pública (y datos asociados a la misma)

  => certificado de clave pública

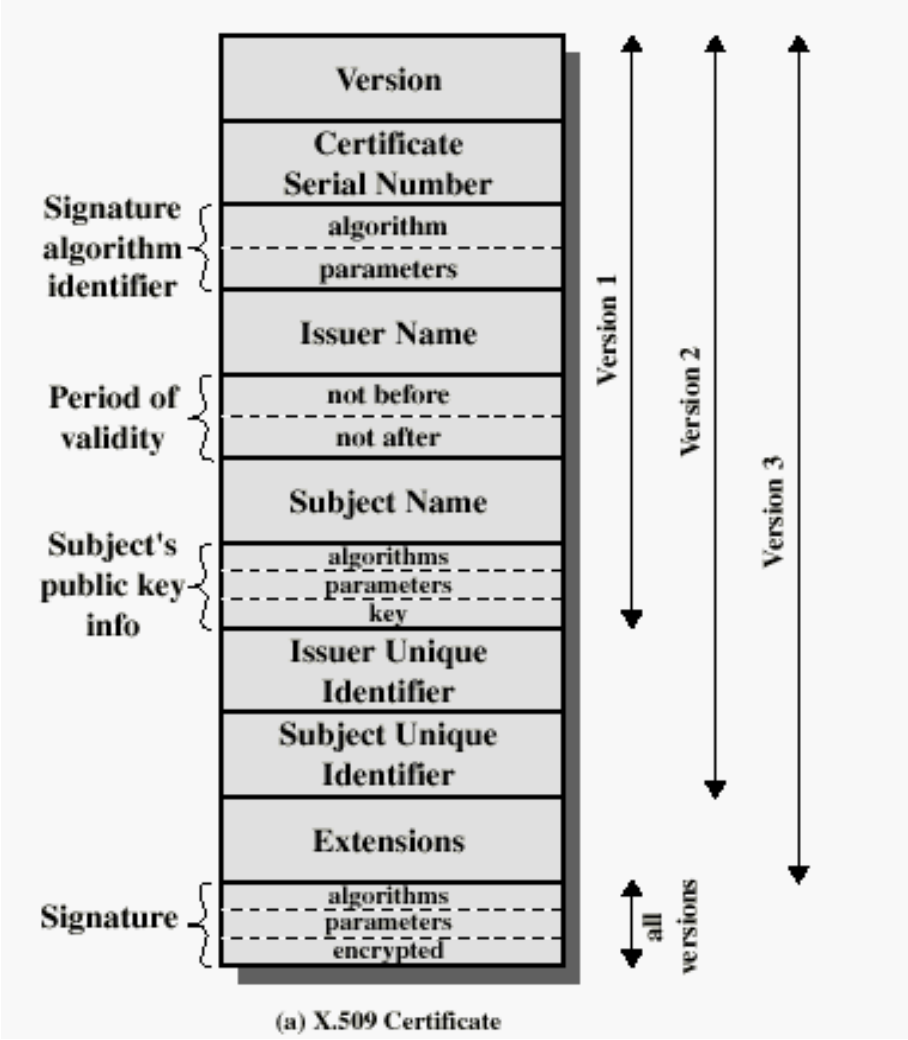
Certificados digitales

- Un certificado digital, básicamente contiene:
  - Clave pública
  - Identificador o nombre de usuario del dueño de la clave
  - Todo esto firmado digitalmente por una tercera parte confiable

- La tercera parte confiable suele ser una Autoridad de Certificación (CA, Certificate Authority) en la que confía la comunidad de usuarios

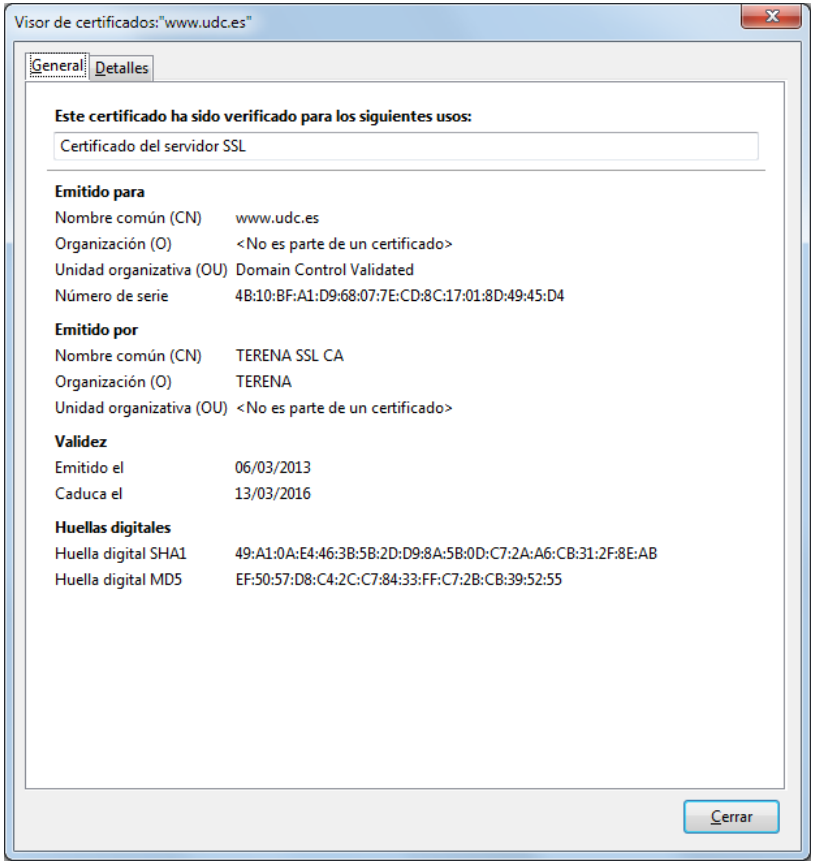
Certificado X.509

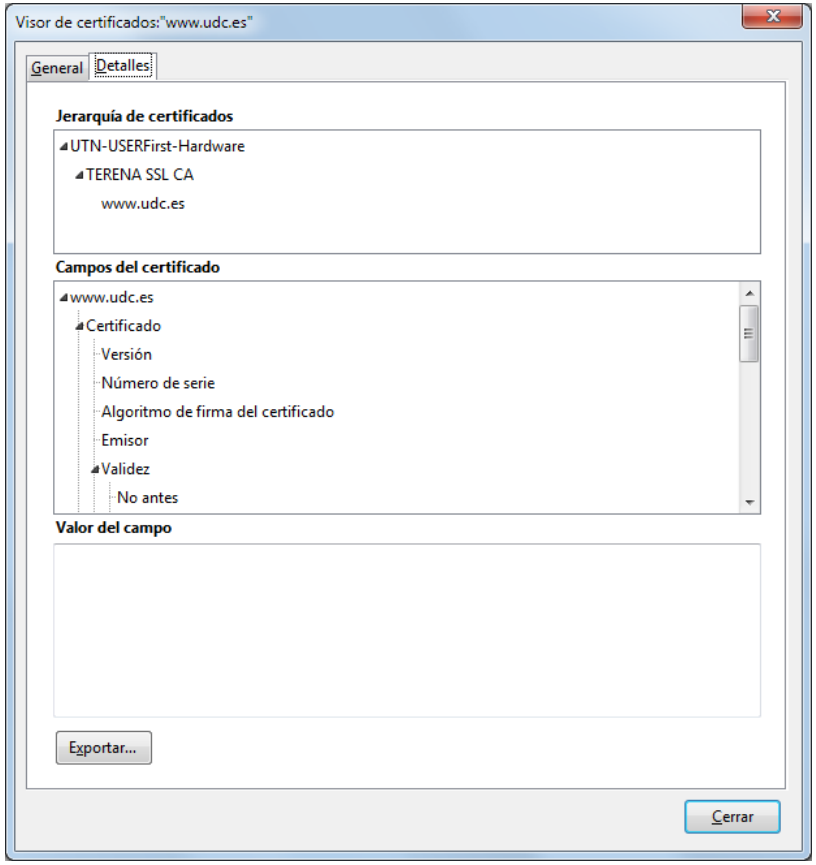
- El formato utilizado actualmente es el X.509v3
- Los certificados X.509v3 se utilizan en multitud de aplicaciones (SSL, SSH, S/MIME, Seguridad IP, ...)



(a) X.509 Certificate

Certificado usado en https://www.udc.es (1)





Certificado usado en https://www.udc.es (2)

```
user@debian-6:~$ openssl x509 -in www.udc.es.pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            4b:10:bf:a1:d9:68:07:7e:cd:8c:17:01:8d:49:45:d4
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=NL, O=TERENA, CN=TERENA SSL CA
        Validity
            Not Before: Mar  6 00:00:00 2013 GMT
            Not After : Mar 12 23:59:59 2016 GMT
        Subject: OU=Domain Control Validated, CN=www.udc.es
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:d7:d4:d0:85:88:47:6d:cd:0c:1e:e2:85:26:27:
                    4c:f5:8d:4f:01:60:2d:55:c3:99:c6:d5:bf:90:32:
                    71:31:4a:9f:fc:77:2a:ff:f6:aa:69:aa:e6:00:3e:
                    68:75:6a:a1:c1:69:81:09:b5:8c:51:1c:ba:d6:20:
                    7d:68:41:c8:7c:f8:b6:2c:38:cb:eb:63:50:61:3d:
                    2d:42:5c:af:59:9b:dc:71:74:dd:fe:3e:22:fa:e4:
                    23:f5:22:1d:06:e1:e2:f6:73:58:4f:f7:7c:f8:7b:
                    0e:0c:c9:42:ec:5e:f0:a5:01:9e:6b:af:0f:90:e1:
                    bb:23:7b:8d:d6:b0:58:81:a6:44:cb:c8:ab:d1:3c:
                    1e:e9:68:c1:d7:37:b6:87:65:23:6b:c1:79:13:95:
                    b5:72:6a:ce:fe:8a:38:69:2f:78:00:86:b1:41:fa:
                    ce:88:76:fa:e9:2d:fd:5f:5d:87:43:67:ca:e0:e8:
                    66:c7:85:c2:3f:99:e9:81:e3:6a:34:69:e1:76:a0:
                    7d:7c:7f:6e:5b:32:32:d3:60:2d:16:81:b7:64:70:
                    58:a4:5b:78:6d:e7:d6:60:d7:48:5a:96:52:31:dd:
                    32:48:7c:51:4e:ac:bb:7c:3f:02:fd:b4:13:1e:7e:
                    6c:10:09:e1:a2:5c:3e:05:80:64:99:09:8b:dc:21:
                    e6:1d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
```

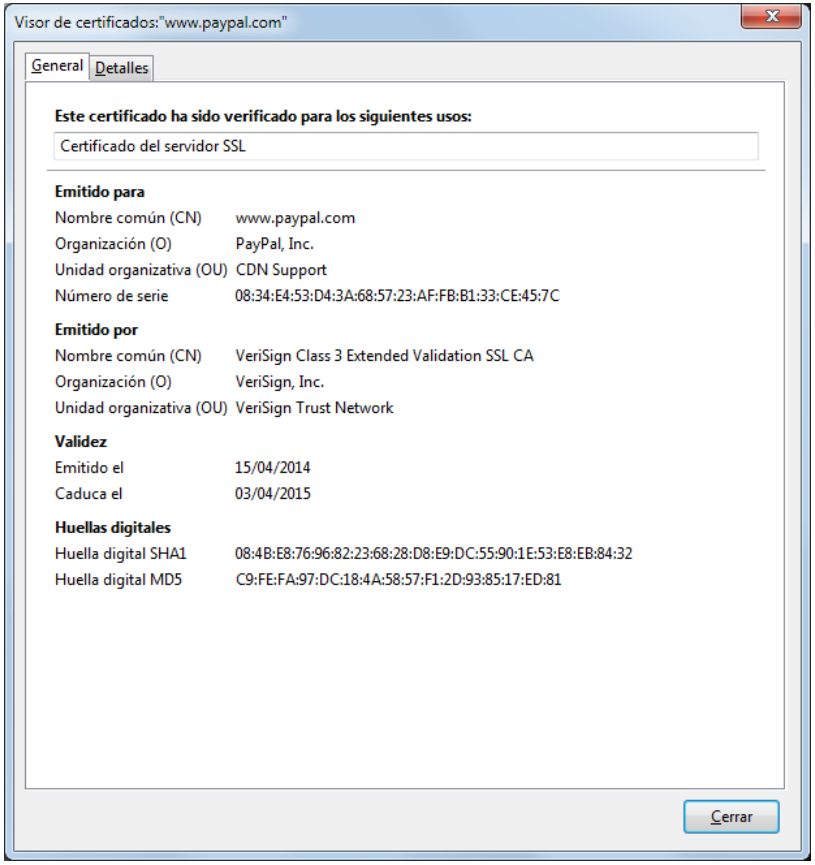
Certificado usado en https://www.udc.es (3)

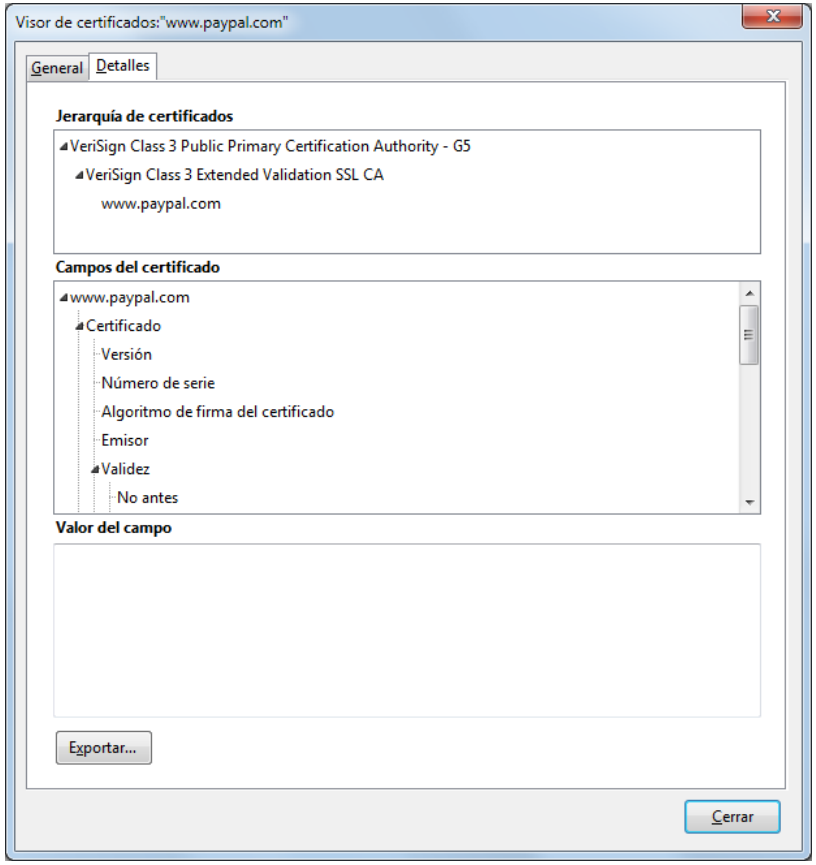
```
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:0C:BD:93:68:0C:F3:DE:AB:A3:49:6B:2B:37:57:47:EA:90:E3:B9:ED

            X509v3 Subject Key Identifier:
                F7:24:5D:C6:01:EF:87:FF:86:08:15:35:97:F5:2E:F0:C9:FE:7D:C7
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.6449.1.2.2.29
                Policy: 2.23.140.1.2.1

            X509v3 CRL Distribution Points:
                URI:http://crl.tcs.terena.org/TERENASSLCA.crl

            Authority Information Access:
                CA Issuers - URI:http://crt.tcs.terena.org/TERENASSLCA.crt
                OCSP - URI:http://ocsp.tcs.terena.org

            X509v3 Subject Alternative Name:
                DNS:www.udc.es
    Signature Algorithm: sha1WithRSAEncryption
```

Certificado usado en https://www.udc.es (y 4)

```
Signature Algorithm: sha1WithRSAEncryption
    8d:8b:38:fb:5d:04:3e:a1:3d:bf:5e:03:89:c2:e9:fd:a9:d5:
    0b:a2:24:3d:85:73:f5:1d:38:12:e4:26:56:26:7d:09:55:c7:
    00:cb:90:d4:b6:fc:05:f5:9d:e6:7e:ac:33:ae:0f:1b:6d:ed:
    cf:02:9b:56:72:68:d4:5d:28:3c:60:3a:9d:c1:e8:25:fd:6d:
    62:d5:b0:65:50:11:d3:4d:fc:dd:32:ae:16:6a:dd:37:7b:4a:
    57:43:f2:53:59:b8:45:12:fa:da:69:bc:66:91:3b:3a:73:91:
    f6:60:97:0e:c0:de:1e:fc:d0:f7:53:aa:d5:93:12:d4:4d:ab:
    09:51:52:5b:59:7e:cb:22:cf:6a:95:0f:77:26:28:82:5f:c8:
    4c:28:89:14:44:f7:00:83:46:f2:2d:81:ea:05:cb:b0:38:7f:
    88:fe:18:f1:62:af:16:b8:e5:86:58:e3:a4:15:40:a8:23:f7:
    77:4c:b2:96:21:9b:71:fe:e3:73:dd:d1:0e:6a:94:f0:d3:d2:
    c1:ef:ca:f5:ec:79:aa:da:ee:e1:58:e7:39:43:70:a0:54:00:
    29:8a:6a:23:4d:70:d1:99:39:4a:b8:ab:d4:6e:4d:49:0d:81:
    9f:4a:17:9a:37:75:92:b4:b4:f8:f2:af:58:63:e4:cf:8c:c4:
    fe:0e:f9:83
```

Certificado usado en https://www.paypal.com (1)





Certificado usado en https://www.paypal.com (2)

```
user@debian-6:~$ openssl x509 -in www.paypal.com.pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            08:34:e4:53:d4:3a:68:57:23:af:fb:b1:33:ce:45:7c
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network, OU=Terms of use
at https://www.verisign.com/rpa (c)06, CN=VeriSign Class 3 Extended Validation SSL
CA
        Validity
            Not Before: Apr 15 00:00:00 2014 GMT
            Not After : Apr  2 23:59:59 2015 GMT
        Subject:
1.3.6.1.4.1.311.60.2.1.3=US/1.3.6.1.4.1.311.60.2.1.2=Delaware/businessCategory=Priva
te Organization/serialNumber=3014267, C=US/postalCode=95131-2021, ST=California,
L=San Jose/street=2211 N 1st St, O=PayPal, Inc., OU=CDN Support, CN=www.paypal.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:be:ae:46:4d:99:6e:6d:6c:35:4b:88:32:38:bb:
                    dc:d0:09:95:d0:9a:e4:36:e7:9f:0a:b0:f2:d7:d2:
                    30:62:03:1f:ad:c6:f4:6d:10:84:f7:79:1b:bc:74:
                    c0:a8:e3:82:fe:d4:0a:93:2e:3d:4b:12:24:ad:ad:
                    5f:5d:ed:1c:c9:1c:6f:13:7b:e2:c1:25:4e:46:5f:
                    4f:3b:2e:5a:cb:c1:5a:b4:82:cf:ad:a3:65:e8:86:
                    33:b5:ed:1d:78:99:a7:c7:d5:fa:10:2e:fb:11:4e:
                    23:58:06:96:87:71:75:51:73:8c:0f:f4:ca:7c:8f:
                    91:25:79:13:dc:b0:f0:de:08:07:01:0b:64:cc:57:
                    6a:12:86:62:17:3e:5d:b9:62:3d:58:7b:2a:6e:f6:
                    a6:30:41:02:fc:ec:64:72:33:d5:d5:3f:6b:6d:97:
                    f3:c1:61:bf:38:3b:ab:41:47:d4:c2:03:d7:3b:59:
```

Certificado usado en https://www.paypal.com (3)

```
                    f3:c1:61:bf:38:3b:ab:41:47:d4:c2:03:d7:3b:59:
                    57:9d:e1:a1:2a:d6:78:e8:83:5d:3d:dd:aa:5d:17:
                    fd:94:d6:e5:7a:ef:02:63:c6:a3:c6:2d:5b:33:08:
                    8b:f5:a5:03:b4:fe:f2:1d:ab:bf:5e:9e:b8:78:39:
                    20:2b:68:61:4f:e4:99:f2:aa:c2:4d:4b:48:cb:68:
                    c2:10:3f:fa:9a:ba:c5:6a:53:8f:22:f3:d7:c9:ed:
                    a4:d5
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:www.paypal.com, DNS:history.paypal.com, DNS:t.paypal.com,
DNS:c.paypal.com, DNS:tms.paypal.com, DNS:tms.ebay.com
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Certificate Policies:
                Policy: 2.16.840.1.113733.1.7.23.6
                  CPS: https://d.symcb.com/cps
                  User Notice:
                    Explicit Text: https://d.symcb.com/rpa

            X509v3 Authority Key Identifier:
                keyid:FC:8A:50:BA:9E:B9:25:5A:7B:55:85:4F:95:00:63:8F:E9:58:6B:43

            X509v3 CRL Distribution Points:
                URI:http://sa.symcb.com/sa.crl

            Authority Information Access:
                OCSP - URI:http://sa.symcd.com
                CA Issuers - URI:http://sa.symcb.com/sa.crt

    Signature Algorithm: sha1WithRSAEncryption
```

Certificado usado en https://www.paypal.com (y 4)

```
    Signature Algorithm: sha1WithRSAEncryption
        76:08:ab:64:f6:f4:0b:e4:81:bd:59:b2:3e:a4:fc:f5:03:75:
        04:59:6a:b5:fe:12:34:2a:04:9c:89:cd:cb:e1:3c:6c:20:39:
        d4:ea:6f:27:34:7f:62:1c:45:72:11:39:c0:45:aa:2a:35:5c:
        b6:06:e3:08:a7:8f:08:af:80:b2:10:ce:a5:28:5b:1c:49:55:
        11:eb:6b:2a:80:c1:09:ed:82:72:48:ca:19:8b:e5:34:94:3c:
        50:26:77:6b:1a:63:ba:6f:63:d1:58:ed:2b:1d:b7:a7:6e:04:
        25:99:c3:94:03:90:ec:0f:4c:93:83:35:86:e3:70:84:0d:3c:
        ce:af:4e:80:4a:d3:91:3f:55:33:2f:1f:67:87:2f:09:a2:41:
        c0:10:4a:2c:c4:88:a0:6f:93:2c:ef:38:d2:61:c7:ec:f3:37:
        7d:c9:32:a5:5c:1e:48:0e:85:6c:47:2a:7f:c6:30:5e:c2:f6:
        2e:dd:e3:4d:ac:ff:ef:48:26:c7:51:74:47:32:46:0b:cd:7a:
        0a:5d:5b:c5:8d:ed:17:bc:de:09:bc:e9:93:a9:7c:85:9c:88:
        a6:83:bc:d6:e5:1f:05:10:df:b2:4f:a2:c5:97:00:8b:57:c7:
        0d:e7:c7:57:57:87:7d:13:9f:5c:5c:f7:f3:cd:00:89:0d:85:
        9a:a2:70:da
```

Curiosidad

- Normalmente, el componente e de la clave publica = $2^{16}+1$ = 65537
  - En binario: 1000000000000001

- Es el **número primo** más grande conocido de la forma $2^{2^n}+1$, donde n=4
  - En teoría de números, los primos de esta forma se conocen como primos de Fermat
    - En concreto, este sería F4

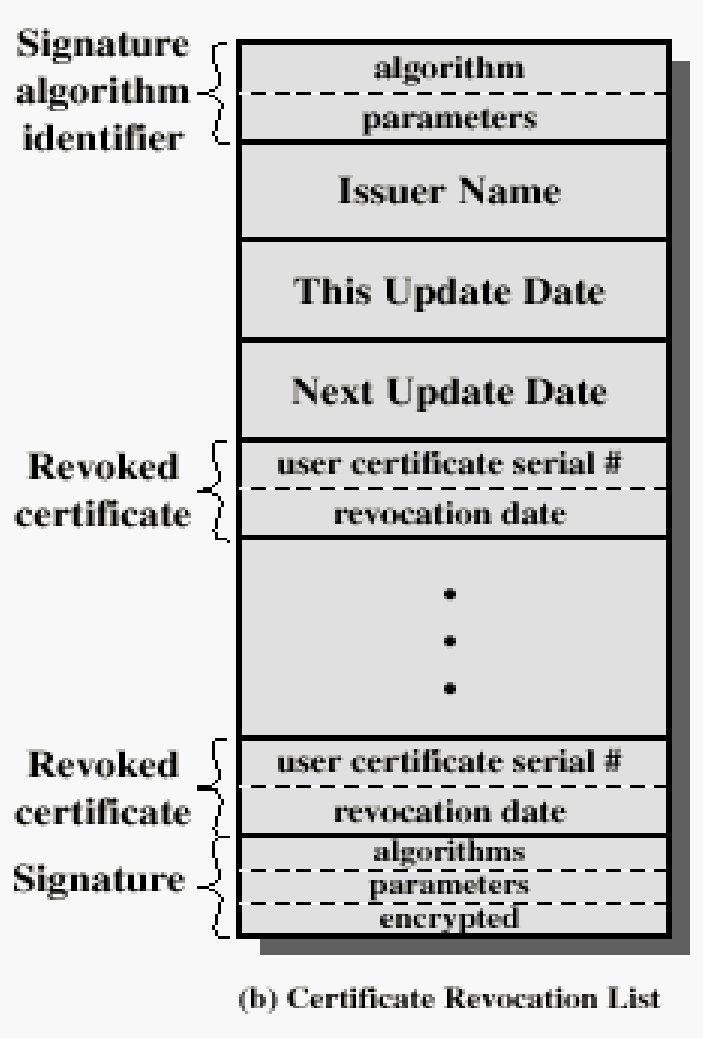
- Tiene un peso Hamming (número de bits = 1) muy bajo
  - Permite realizar operaciones muy rápido

---

1) Número 65537 https://en.wikipedia.org/wiki/65537_%28number%29

Lista de Revocación de Certificados (Certificate Revocation List, CRL)

- Los certificados tienen un período de validez. Si dejan de ser válidos dentro de ese período, es necesario revocarlos.
- Razones para la revocación:
  - Se sospecha que la clave privada del usuario está comprometida.
  - El usuario ya no está certificado por esa AC.
  - Se sospecha que el certificado de la AC está comprometido.



(b) Certificate Revocation List

AUTORIDADES DE CERTIFICACIÓN

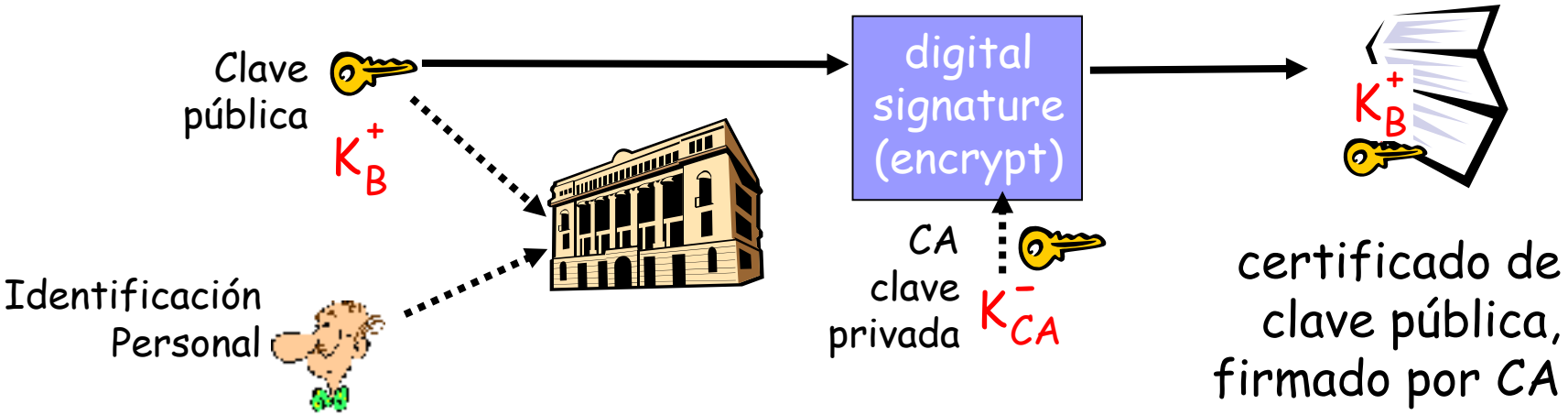
Autoridades de Certificación

- Un usuario puede presentar su **clave pública** ante una CA, para obtener un certificado y luego publicarlo.

- Así, cualquiera que necesite la clave pública de este usuario puede obtener el certificado y verificar que es válida por medio de la firma adjunta.

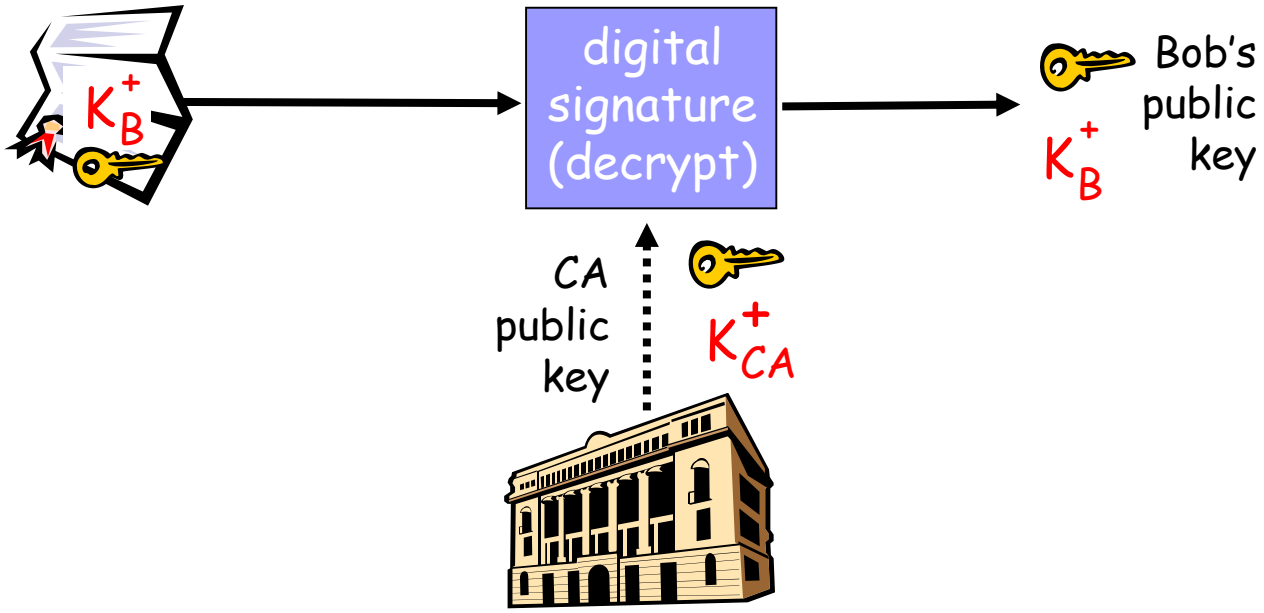
Autoridades de Certificación

- Obtención del certificado.



Autoridades de certificación

- Comprobación del certificado.
  - Obtención del certificado (propio usuario, repositorio, etc.)
  - Verificar la firma del certificado, usando la clave pública de la CA



Autoridades de certificación

- Características:
  - Cualquier usuario con acceso a la clave pública de la AC puede verificar la clave pública del usuario que fue certificada.
  - Sólo la AC puede modificar el certificado sin que esto se detecte.
  - Evita la necesidad de un repositorio de acceso común.

Ejemplos de Autoridades de Certificación

- VeriSign
- Thawte
- GeoTrust
- FNMT
- …

PROTOCOLOS SEGUROS

Contenido

- Secure Socket Layer (SSL)
- Secure Shell (SSH)

SSL

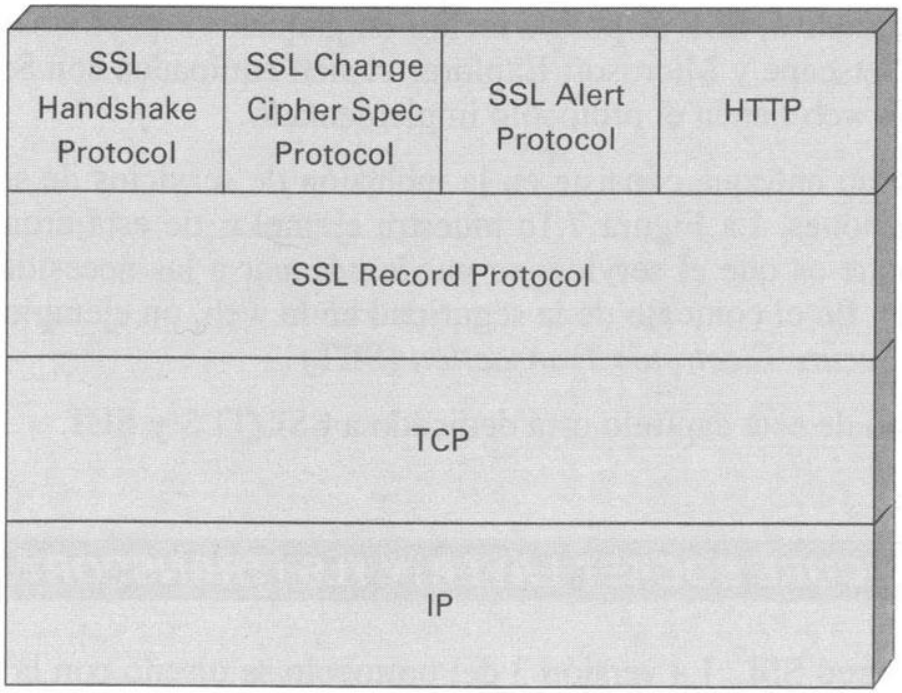
Secure Socket Layer (SSL)

- Es un protocolo de seguridad que permite establecer conexiones seguras a través de redes inseguras, como Internet
- Diseñado por Netscape en 1993
- Está principalmente orientado al Web, aunque está disponible para cualquier aplicación TCP

Secure Socket Layer (SSL)

- Proporciona:
  - **Cifrado**. Cliente y Servidor utilizan técnicas de cifrado asimétrico para intercambiar claves compartidas. Se utilizan dichas claves compartidas para cifrar y descifrar (c. simétrico) la comunicación.
  - **Integridad**. Se utilizan hash para garantizar que los mensajes no son alterados en tránsito
  - **Autenticación del servidor**. El servidor debe disponer de un certificado digital emitido por una AC reconocida (que figure en la lista de ACs del cliente)
  - **Autenticación del cliente (opcional)**. Usualmente, el cliente se autentica ante el servidor mediante un nombre de usuario y una contraseña, aunque también podría hacerlo mediante un certificado digital (algunos bancos y administraciones públicas ofrecen esta posibilidad)

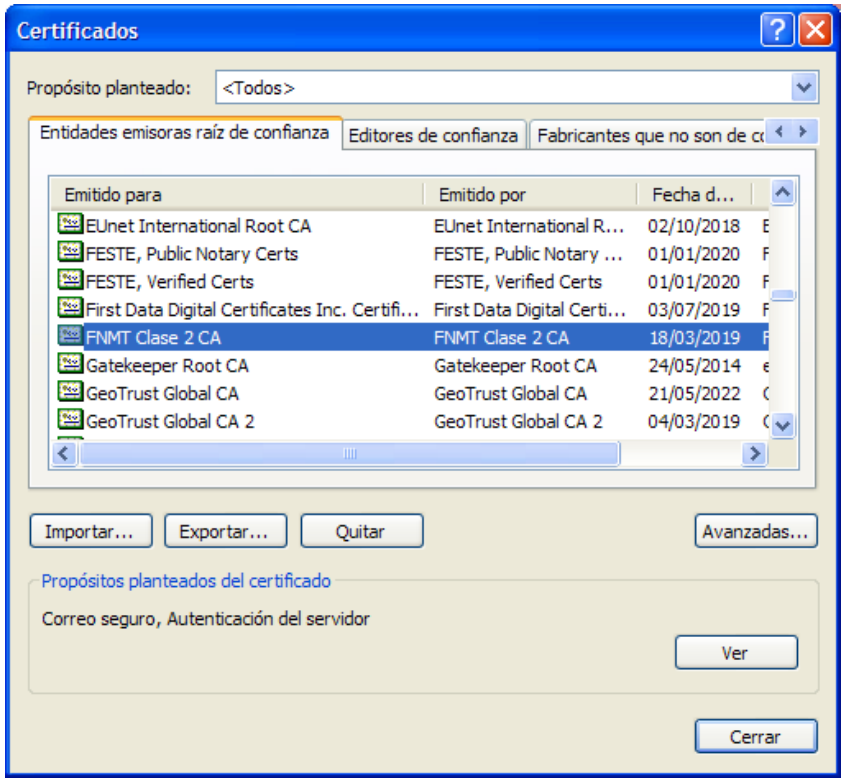
Secure Socket Layer (SSL)



Pila de protocolos SSL (Stallings, 2004)

Secure Sockets Layer
Autenticación del servidor

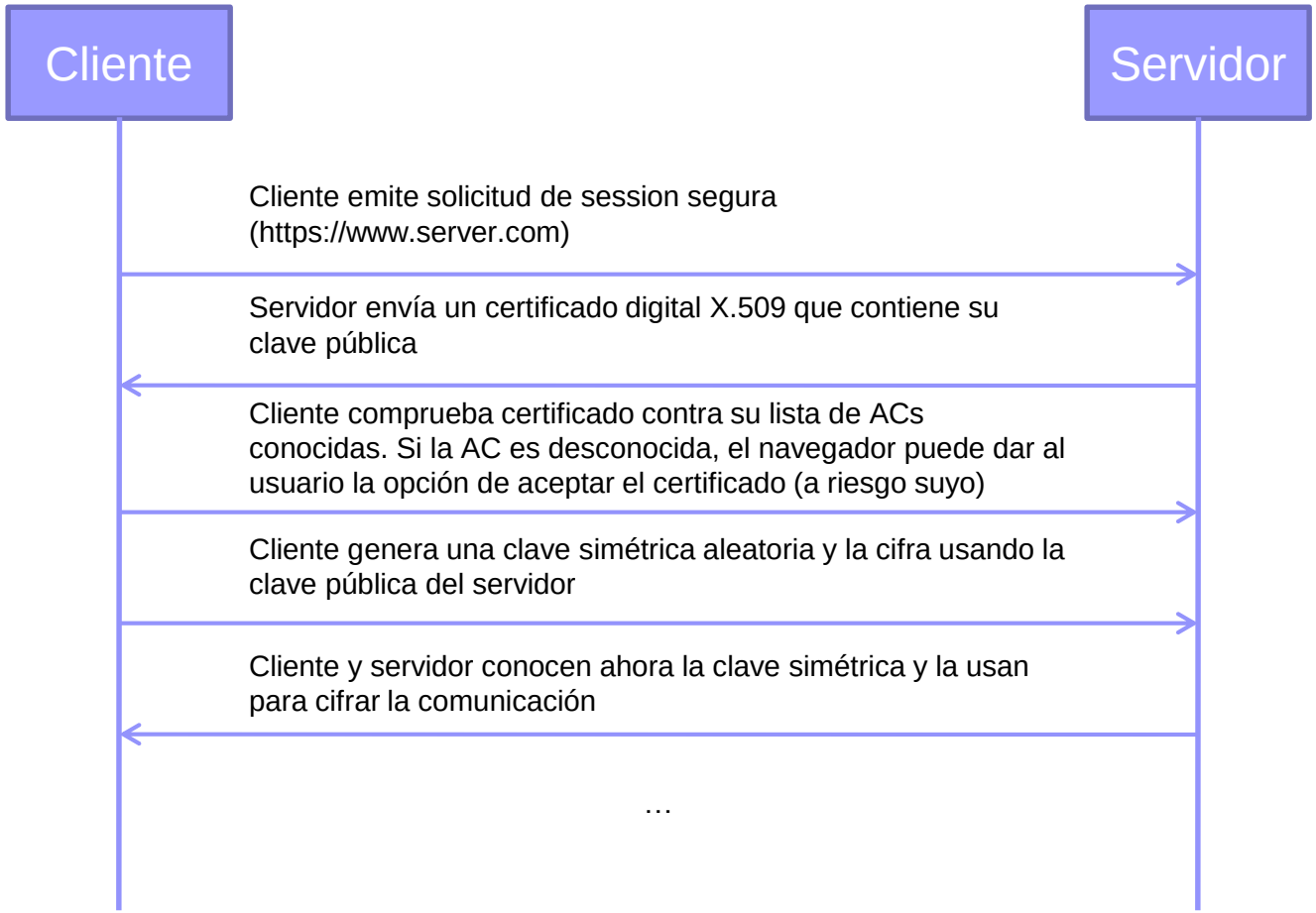
- Los navegadores actuales incluyen soporte para SSL, así como las claves públicas de las **Autoridades de Certificación** de confianza.
- El navegador solicita al servidor que le envíe su certificado. Este certificado debe ser emitido por una AC de confianza.
- El navegador utiliza la clave pública de la AC para verificar que el certificado del servidor es válido.



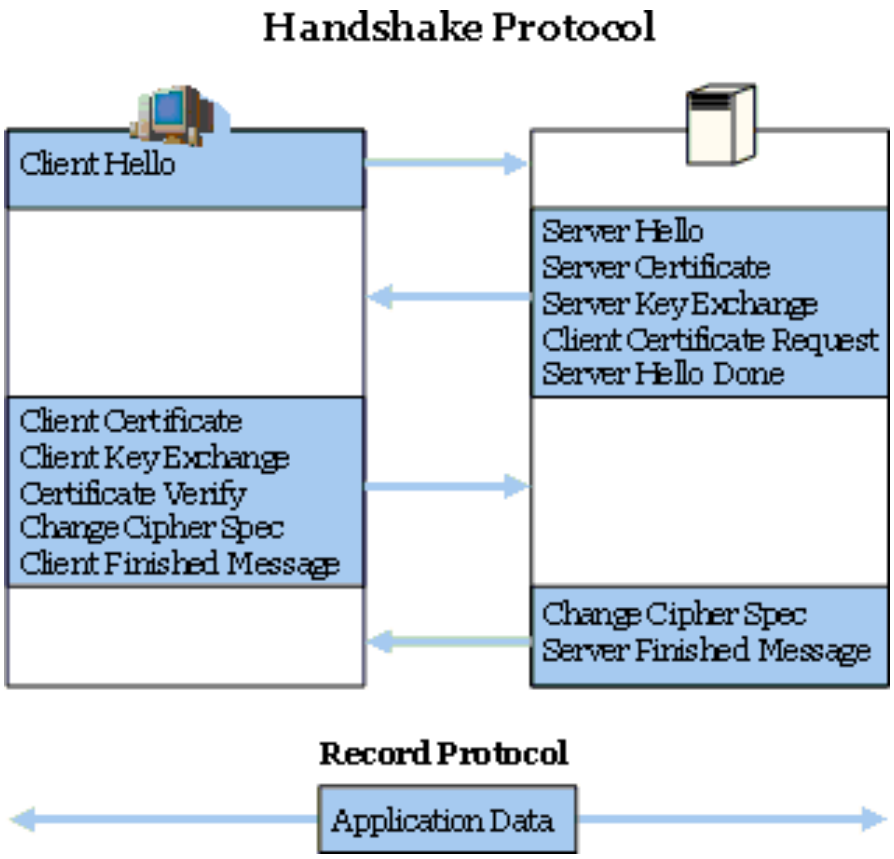
En el navegador se pueden ver las Autoridades de Certificación de confianza

Secure Sockets Layer

SSL HandShake (versión simplificada)



Secure Sockets Layer
SSL HandShake



Extraído de: http://technet.microsoft.com/es-es/library/cc785811(WS.10).aspx

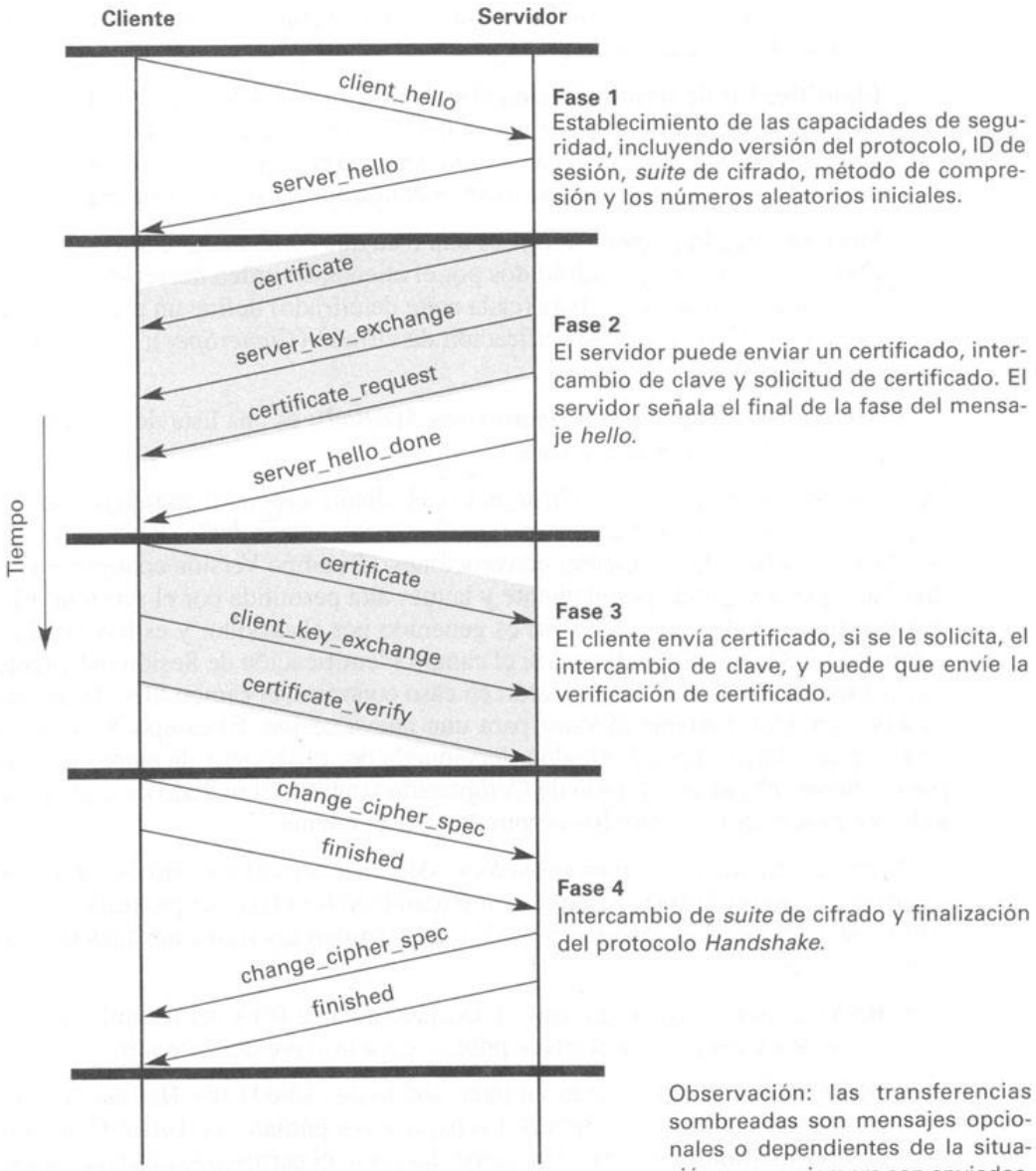
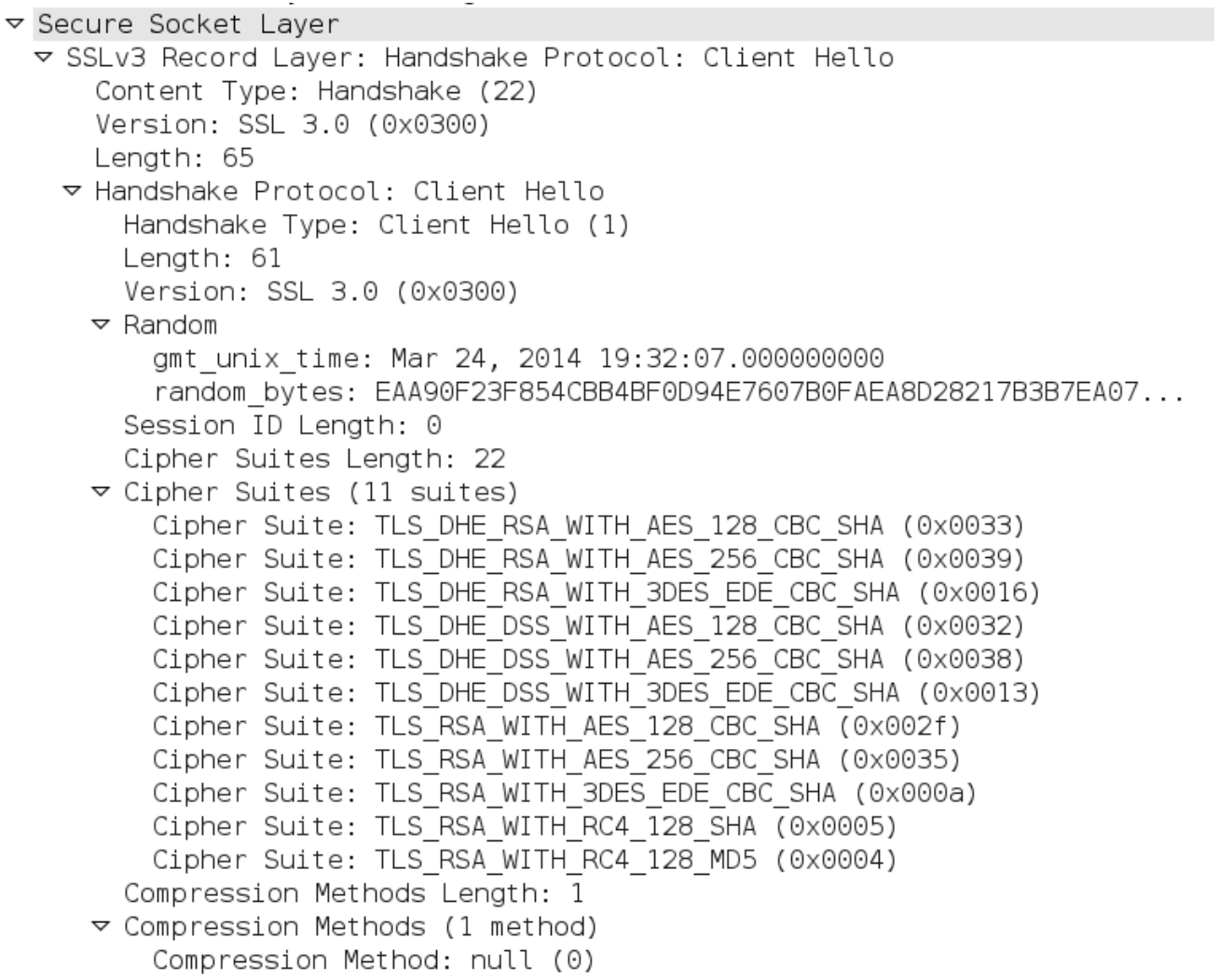
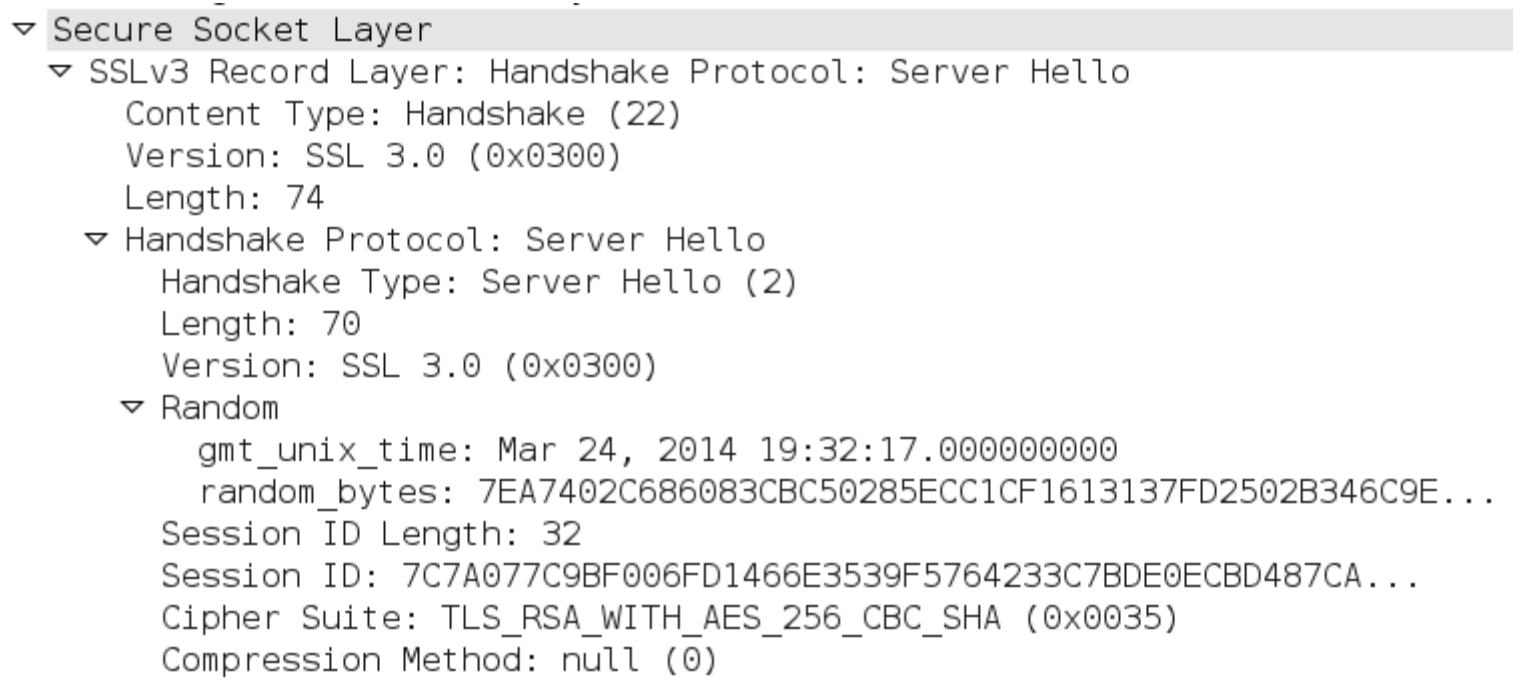


Figura 7.6 Acción del protocolo *Handshake*

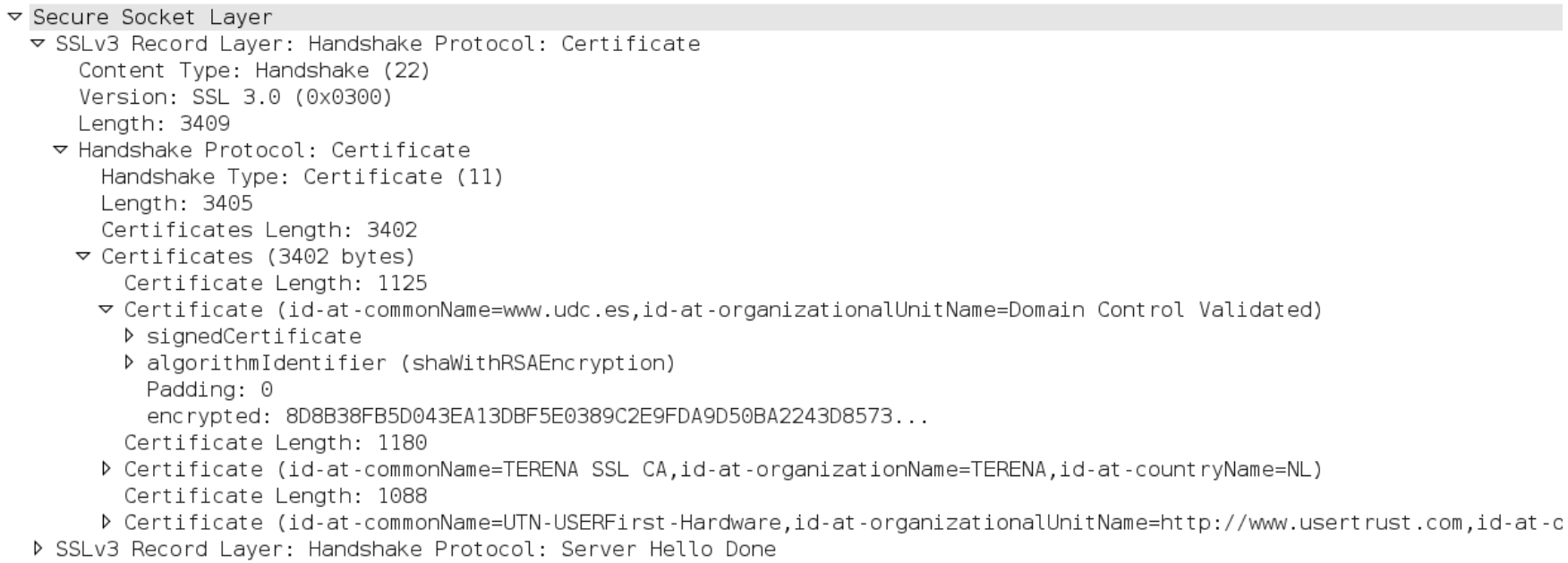
Client Hello



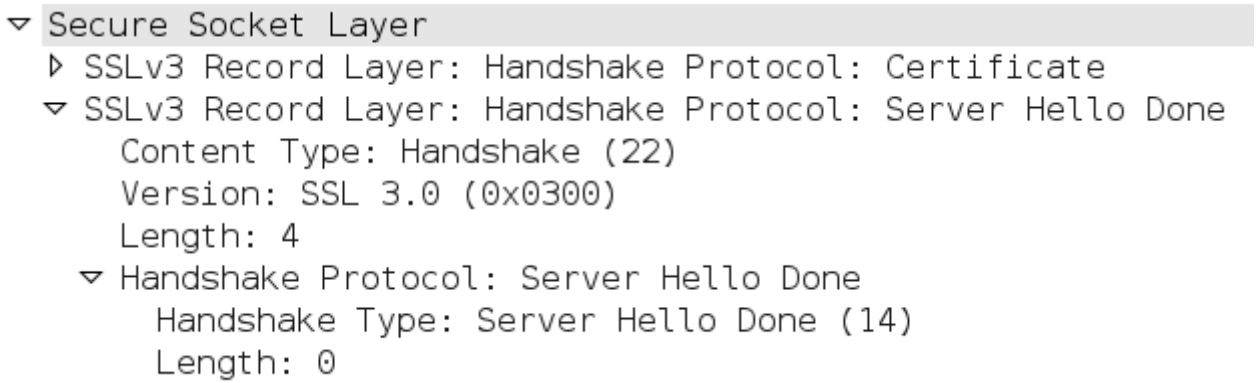
Server Hello



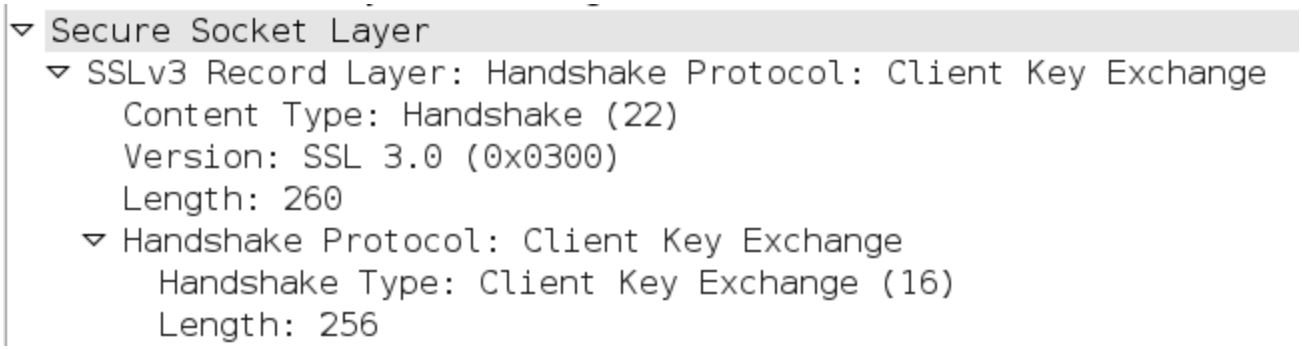
Server Certificate



Server Hello Done



Client Key Exchange



Client Change Cipher Spec & Finished

Secure Sockets Layer

- Algoritmos utilizados:
  - Cifrado simétrico:
    - DES – Data Encryption Standard: block
    - 3DES – Triple strength: block
    - RC2 – Rivest Cipher 2: block
    - RC4 – Rivest Cipher 4: stream
  - Cifrado asimétrico:
    - RSA
  - Intercambio de claves:
    - RSA
    - Diffie-Hellman

Secure Sockets Layer

- Algunos puertos utilizados:
  - https: 443
  - smtps: 465
  - ldaps: 636
  - imaps: 993
  - pop3s: 995
  - ftps: 989&990
  - …

Ataques a SSL

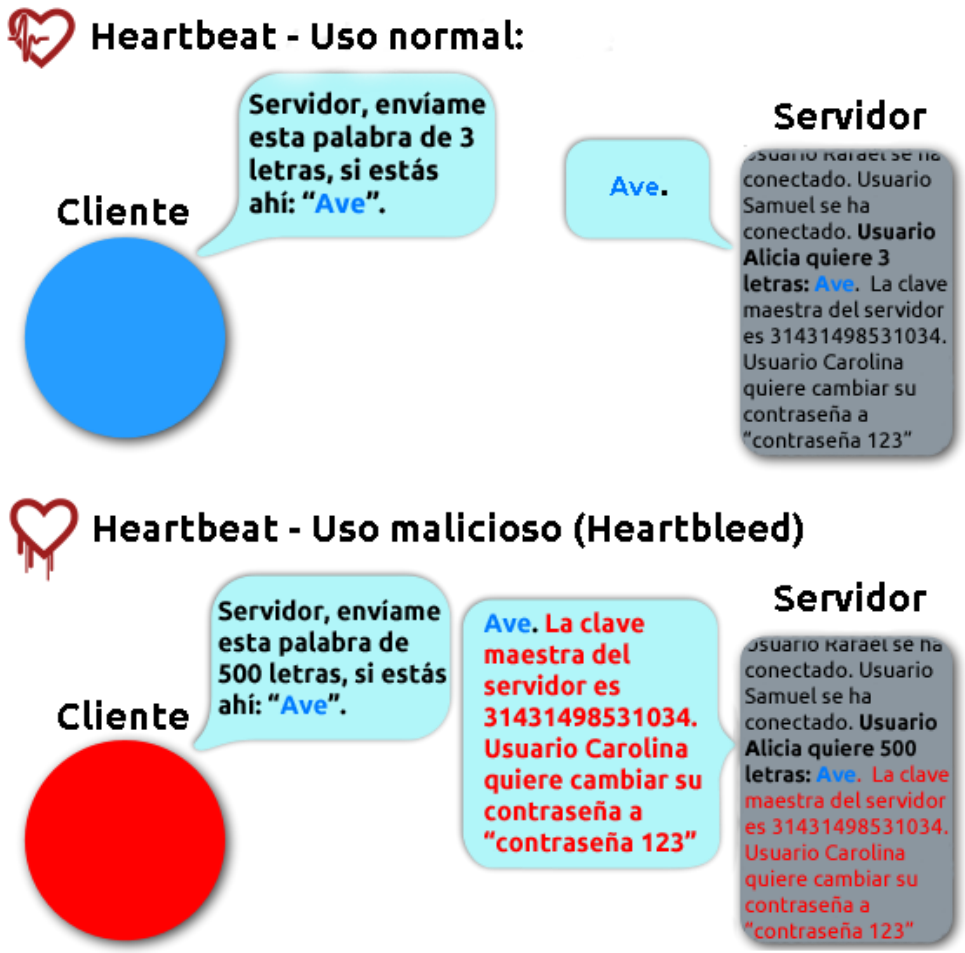
- Generación de certificado falso con ettercap
- Sslstrip: https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf
- Heartbleed (OpenSSL): https://es.wikipedia.org/wiki/Heartbleed
- Poodle: https://es.wikipedia.org/wiki/Ataque_POODLE
- Breach: http://breachattack.com/

Heartbleed [1]

- Bug en OpenSSL 1.0.1f (CVE-2014-0160)
- Permite a un atacante leer la memoria de un servidor o un cliente
- Ataca la extensión Heartbeat
  - Se propuso como un estándar en febrero del 2012
  - Provee una forma de probar y mantener viva un enlace de comunicación segura sin la necesidad de renegociar la conexión cada vez

[1] https://es.wikipedia.org/wiki/Heartbleed

Heartbleed


Heartbeat - Uso normal:
Cliente
Servidor, envíame esta palabra de 3 letras, si estás ahí: "Ave".
Servidor
Ave.
conectado. Usuario Samuel se ha conectado. Usuario Alicia quiere 3 letras: Ave. La clave maestra del servidor es 31431498531034. Usuario Carolina quiere cambiar su contraseña a "contraseña 123"
Heartbeat - Uso malicioso (Heartbleed)
Cliente
Servidor, envíame esta palabra de 500 letras, si estás ahí: "Ave".
Servidor
Ave. La clave maestra del servidor es 31431498531034. Usuario Carolina quiere cambiar su contraseña a "contraseña 123"
conectado. Usuario Samuel se ha conectado. Usuario Alicia quiere 500 letras: Ave. La clave maestra del servidor es 31431498531034. Usuario Carolina quiere cambiar su contraseña a "contraseña 123"

Poodle

- Padding Oracle On Downgraded Legacy Encryption
- Afecta a SSL 3.0 y a algunas versiones de TLS
- Si los atacantes explotan exitosamente esta vulnerabilidad, en promedio, solo necesitan hacer 256 solicitudes SSL 3.0 para revelar un byte de los mensajes cifrados
- CVE-2014-3566

[1] https://es.wikipedia.org/wiki/Ataque_POODLE

Utilidades relacionadas

- Test:
  - https://www.ssllabs.com/ssltest/index.html
- Plug-in:
  - HTTS Everywhere

SSH

SSH

- Desarrollado en 1995 por Tatu Ylönen.
- Es un protocolo de red que permite establecer un canal seguro entre dos dispositivos de red.
- Diseñado para ofrecer una alternativa segura a Telnet y FTP.
- También soporta *tunneling*

SSH

- Provee las siguientes garantías de seguridad:
  - **Autenticación**: El cliente puede verificar que se está conectando al servidor al que indicó.
  - **Confidencialidad**: Los datos intercambiados se transmiten usando cifrado.
  - **Integridad**: Se verifica la integridad de los datos intercambiados mediante hash.

SSH

- SSH se estructura en tres protocolos, que se ejecutan sobre TCP:

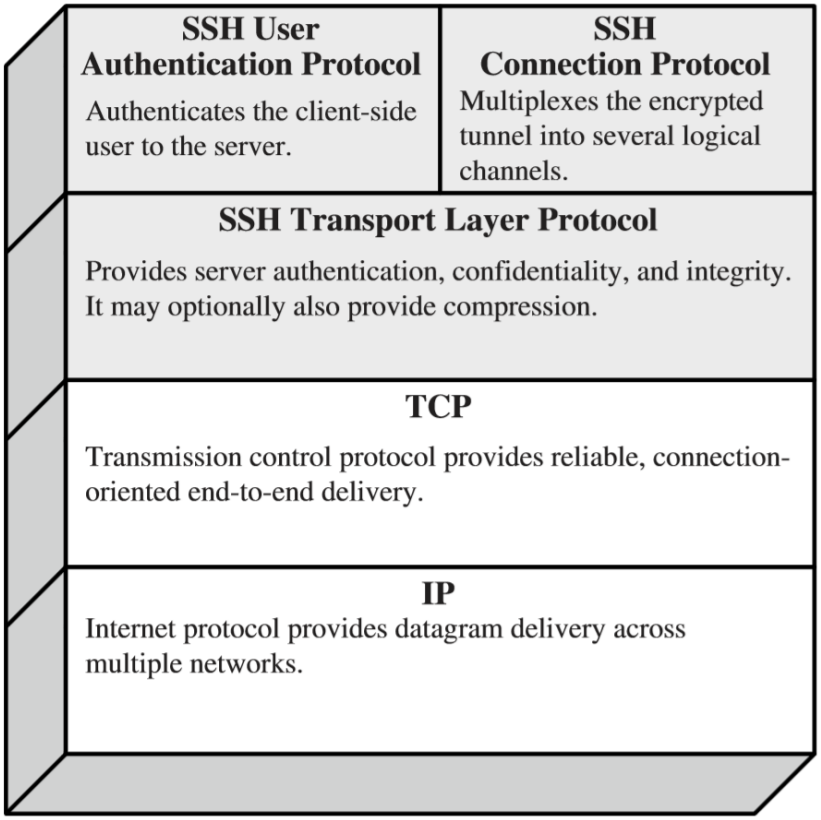


Figure 16.8 SSH Protocol Stack

SSH

- **Transport Layer Protocol**:
  - Proporciona autenticación del servidor, confidencialidad de datos e integridad de datos con *forward secrecy* o secreto-adelante
    - *forward secrecy*: si una clave es comprometida durante una sesión, no se compromete la seguridad de los datos protegidos con claves anteriores (claves válidas en periodos anteriores de tiempo) [1]
  - Podría opcionalmente proporcionar compresión

---

[1] https://es.wikipedia.org/wiki/Perfect_forward_secrecy

SSH

- **User Authentication Protocol**:
  - Autentica el usuario al servidor

- **Connection Protocol**:
  - Multiplexa múltiples canales de comunicación lógicos sobre una único conexión SSH subyacente

Archivos de configuración

- **known_hosts** y **ssh_known_hosts**
  - **known_hosts**
    - Se ubica en ~/.ssh
    - Almacena las claves públicas de los servidores a los que el usuario se conectó previamente
      - La primera vez que el usuario se conecta a un servidor ssh, se pide autorización para añadir la clave pública de dicho servidor al repositorio del usuario (known_hosts)
    - Cada clave va precedida del nombre del servidor (o un hash del mismo)

Archivos de configuración

- **known_hosts** y **ssh_known_hosts**
  - **known_hosts**

```
lsi@debian:~/.ssh$ ssh 10.10.102.44
The authenticity of host '10.10.102.44 (10.10.102.44)' can't be established.
RSA key fingerprint is 10:14:37:eb:65:71:06:c2:9f:7b:0e:41:ab:02:ae:fd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.102.44' (RSA) to the list of known hosts.
lsi@10.10.102.44's password:
```

```
lsi@debian:~/.ssh$ more /home/lsi/.ssh/known_hosts
|1|LHAfcJcIUyDBpyZMqaI5Av/Xc5Y=|3vvByPEPNi59hUJtkQ2Ln5H3NSI= ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC4rv3r8qvm7vtRjCSZuee/MYq5/FORCWblZQXN9QdL9RP8/q+i28G+o
ZtFImuYJQ8JBtxS0MoC862tWPjBpB7fgA/w5GOh1tflEHV3I3lZN2Piznxzhi/aWyw19p4Ojedw+R0im7sgOL
SwfJQrTr0OTsiXYeB8EozM9WzO5eqiuaRz303Qxxcnu17bAKX3VJB48+uGxGBWEKBdoQs43TQbRmTQKNur/Ca
ck+9L0Mkg9FKQQ9xNwuXBE1BAKGp1agyfuyD0rfWvRtNL4hX2JuRbjHxigUb+zP7hYv+7M9iItahkL9/JAWzc
XgOfW59LWk38ubBsHGJRrmvdG8PJzDoD
```

Archivos de configuración

- **known_hosts** y **ssh_known_hosts**
  - **known_hosts**

```
lsi@debian:~/.ssh$ ssh 10.10.102.44
The authenticity of host '10.10.102.44 (10.10.10
RSA key fingerprint is 10:14:37:eb:65:71:06:c2:9
Are you sure you want to continue connecting (ye
Warning: Permanently added '10.10.102.44' (RSA)
lsi@10.10.102.44's password:
```

Hash del nombre del servidor.
Puede ser reemplazado por el nombre del servidor o su dirección IP, para facilitar el mantenimiento.

```
lsi@debian:~/.ssh$ more /home/lsi/.ssh/known_hosts
|1|LHAfcJcIUyDBpyZMqaI5Av/Xc5Y=|3vvByPEPNi59hUJtkQ2Ln5H3NSI= ssh-rsa AAAAB3NzaC1yc2EA
AAADAQABAAABAQC4rv3r8qvm7vtRjCSZuee/MYq5/FORCWblZQXN9QdL9RP8/q+i28G+oZtFImuYJQ8JBtxS0
MoC862tWPjBpB7fgA/w5GOh1tflEHV3I3lZN2Piznxzhi/aWyw19p4Ojedw+R0im7sgOLSwfJQrTr0OTsiXYe
B8EozM9WzO5eqiuaRz303Qxxcnu17bAKX3VJB48+uGxGBWEKBdoQs43TQbRmTQKNur/Cack+9L0Mkg9FKQQ9x
NwuXBE1BAKGp1agyfuyD0rfWvRtNL4hX2JuRbjHxigUb+zP7hYv+7M9iItahkL9/JAWzcXgOfW59LWk38ubBs
HGJRrmvdG8PJzDoD
```

Archivos de configuración

- **known_hosts** y **ssh_known_hosts**
  - **known_hosts**

```
lsi@debian:~/.ssh$ ssh 10.10.102.44
The authenticity of host '10.10.102.44 (10.10.102.44)' can't be established.
RSA key fingerprint is 10:14:37:eb:65:71:06:c2:9f:7b:0e:41:ab:02:ae:fd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.102.44' (RSA) to the list of known hosts.
lsi@10.10.102.44's password:
```

```
lsi@debian:~/.ssh$ more /home/lsi/.ssh/known_hosts
10.10.102.44 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC4rv3r8qvm7vtRjCSZuee/MYq5/FORCWbl
ZQXN9QdL9RP8/q+i28G+oZtFImuYJQ8JBtxS0MoC862tWPjBpB7fgA/w5GOh1tflEHV3I3lZN2Piznxzhi/aW
yw19p4Ojedw+R0im7sgOLSwfJQrTr0OTsiXYeB8EozM9WzO5eqiuaRz303Qxxcnu17bAKX3VJB48+uGxGBWEK
BdoQs43TQbRmTQKNur/Cack+9L0Mkg9FKQQ9xNwuXBE1BAKGp1agyfuyD0rfWvRtNL4hX2JuRbjHxigUb+zP7
hYv+7M9iItahkL9/JAWzcXgOfW59LWk38ubBsHGJRrmvdG8PJzDoD
```

Archivos de configuración

- **known_hosts** y **ssh_known_hosts**
  - **ssh_known_hosts**
    - Si existe, se ubica en /etc/ssh
      - Almacena claves públicas pero a nivel de máquina, en lugar de a nivel de usuario
      - Debe mantenerlo el administrador (a diferencia de known_hosts, que se mantiene automáticamente)
      - Tiene el mismo formato que known_hosts, por lo que podemos crearlo a partir de este

```
root@debian:/$ cp /home/lsi/.ssh/known_hosts /etc/ssh/ssh_known_hosts
```

      - Si una clave está en ssh_known_hosts (global) no es necesario que esté en known_hosts (usuario)

Archivos de configuración

- Pero… ¿cómo se comprueba la autenticidad de la clave?

```
#CLIENTE
lsi@debian:~/.ssh$ ssh 10.10.102.44
The authenticity of host '10.10.102.44 (10.10.102.44)' can't be established.
RSA key fingerprint is 10:14:37:eb:65:71:06:c2:9f:7b:0e:41:ab:02:ae:fd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.102.44' (RSA) to the list of known hosts.
lsi@10.10.102.44's password:
```

```
#SERVIDOR
root@debian:~/.ssh$ ls /etc/ssh/
moduli          sshd_config       ssh_host_dsa_key.pub  ssh_host_rsa_key.pub
ssh_config      ssh_host_dsa_key  ssh_host_rsa_key      ssh_known_hosts

root@debian:~/.ssh$ ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
2048 10:14:37:eb:65:71:06:c2:9f:7b:0e:41:ab:02:ae:fd /etc/ssh/ssh_host_rsa_key.pub
(RSA)
```

Autenticación de usuario con clave pública

- SSH permite autenticación del usuario utilizando clave pública en lugar de login / password
- Para esto se necesita:
  - Generar un par de claves (pública/privada) en el cliente
    - Herramienta ssh-keygen
  - Añadir la clave pública del cliente al archivo authorized_keys del servidor

Autenticación de usuario con clave pública

- Generar un par de claves (pública/privada) en el cliente

```
#CLIENTE
lsi@cliente:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/lsi/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/lsi/.ssh/id_rsa.
Your public key has been saved in /home/lsi/.ssh/id_rsa.pub.
The key fingerprint is:
b7:f4:27:2f:0f:a6:43:71:f3:b7:87:d5:7e:80:67:e2 lsi@d133
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|          . o    |
|        S oo +  .|
|         o.oo = +|
|         ...=+.*.|
|          .oE=..+|
|          .. oo o|
+-----------------+
```

Autenticación de usuario con clave pública

- Añadir la clave pública del cliente al archivo authorized_keys del servidor

```
#CLIENTE
lsi@cliente:~$ scp /home/lsi/.ssh/id_rsa.pub \
lsi@servidor:/home/lsi/.ssh/lsi_cliente_id_rsa.pub
```

```
#SERVIDOR
lsi@servidor:~/.ssh$ cat lsi_cliente_id_rsa.pub >> authorized_keys
```

  - Una vez hecho esto, el cliente debería poder conectarse sin que se le solicite login / password

Control de acceso [1]

- iptables
- tcpwrappers
- ssh:
  - archivo de configuración /etc/ssh/sshd_config [2]

```
#EJEMPLOS:
AllowUsers bob
AllowUsers alice@host2.example.com
AllowUsers *@10.10.102.30
```

[1] http://cromwell-intl.com/linux/ssh-2-access-control.html
[2] http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man5/sshd_config.5?query=sshd_config&arch=i386

Port forwarding

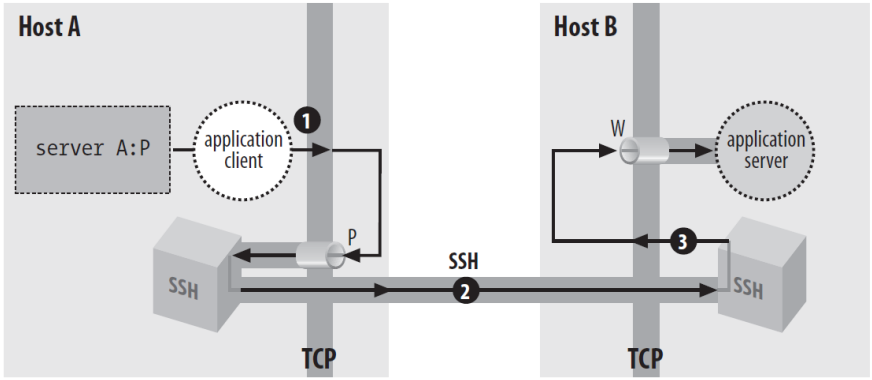
- Permite asegurar accesos por medio de servicios inseguros
  - telnet
  - rsh
  - rlogin
  - ftp, vsftpd
  - mail
  - web
  - ...
- Dos opciones: Local Forwarding y Remote Forwarding

Local Forwarding

- El túnel se abre desde el cliente (la máquina desde la que queremos acceder al servicio inseguro)
- El servidor del servicio inseguro debe tener también servidor de ssh

```
cliente$ ssh –L localPort:serverName:remotePort username@serverName
```

Local Forwarding



(Barrett et. al., 2005)

HostA

Cliente SSH
Cliente telnet (u otra aplicación insegura que queramos tunelizar: mail, web, ftp, ...)

Ejecuta:
Para crear el tunel:
ssh -L 2323:HostB:23 HostB
(crea una sesión ssh)

Para conectarse al telnet, DESDE OTRA TERMINAL:
telnet HostA 2323

HostB

Servidor SSH
Servidor telnet (u otra aplicación insegura que queramos tunelizar: mail, web, ftp, ...)

No ejecuta nada

Remote Forwarding

- El túnel se abre desde el servidor (la máquina que tiene el servidor inseguro al que queremos acceder)
- La máquina cliente (del servicio inseguro) debe tener servidor de ssh

```
servidor$ ssh –R remotePort:clientName:localport username@clientName
```

SSH. Herramientas

- SCP (Secure Copy Protocol)
  - Permite transmitir ficheros entre máquinas sobre una conexión cifrada y segura [1]

```
# SINOPSIS
scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file] [-l limit]
[-o ssh_option] [-P port] [-S program]
[ [user@]host1:]file1 ... [ [user@]host2:]file2
```

```
# Ejemplos:

# Copiar el archivo "foobar.txt" desde un host remoto al host local
$ scp your_username@remotehost.edu:foobar.txt /some/local/directory

# Copiar el archivo "foobar.txt" desde el host local a un host remoto
$ scp foobar.txt your_username@remotehost.edu:/some/remote/directory

# CIFRADO
# scp usa Triple-DES por defecto para cifrar los datos transmitidos, pero es posible
# indicar otros algoritmos:
$ scp -c blowfish some_file your_username@remotehost.edu:~
```

[1] http://linux.die.net/man/1/scp

SSH. Herramientas

- SFTP (SSH File Transfer Protocol)
  - Permite abrir una conexión segura interactiva de ftp [1]

```
# SINOPSIS
sftp [-1Cv] [-B buffer_size] [-b batchfile] [-F ssh_config] [-o ssh_option]
[-P sftp_server_path] [-R num_requests] [-S program] [-s subsystem | sftp_server]
host

sftp [user@]host[:file ...]
```

```
# Ejemplos:

# Abrir una sesión de ftp en example.com
$ sftp user@example.com

# Comandos (modo interactivo)
cd (lcd), ls (lls), put, get, pwd (lpwd), !, bye (exit)
```

[1] http://linux.die.net/man/1/sftp

Bibliografía

- Recomendada
  - Santos del Riego, A (2016). ***Legislación [Protección] y Seguridad de la Información***. Disponible en: http://psi-udc.blogspot.com.
  - Stallings, W. (2011). ***Cryptography and Network Security: Principles and Practice (Fifth ed.)***: Prentice Hall.

- Complementaria
  - Schneier, B. (2007). ***Applied cryptography: protocols, algorithms, and source code in C***: Wiley-India.
    - Información en: https://www.schneier.com/book-applied.html

Bibliografía

- Complementaria
  - Daniel J. Barrett, Richard E. Silverman, Robert G. Byrnes. (2005). **SSH, the secure shell: the definitive guide**. 2nd Edition. O'Reilly.
  - Gluck, Y., Harris, N., & Prado, A. (2013). **Breach: Reviving The Crime Attack**. Disponible en: http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf
  - Schneier, B. (2007). ***Applied cryptography: protocols, algorithms, and source code in C***: Wiley-India.
    - Información en: https://www.schneier.com/book-applied.html

Recursos de interés

- Intypedia:
  - Lección 1. Historia de la criptografía y su desarrollo en Europa. http://www.criptored.upm.es/intypedia/video.php?id=historia-criptografia&lang=es
  - Lección 2. Sistemas de cifra con clave secreta. http://www.criptored.upm.es/intypedia/video.php?id=criptografia-simetrica&lang=es
  - Lección 3. Sistemas de cifra con clave pública. http://www.criptored.upm.es/intypedia/video.php?id=criptografia-asimetrica&lang=es
  - Lección 14. Funciones Unidireccionales y algoritmos de hash. http://www.criptored.upm.es/intypedia/video.php?id=introduccion-funciones-hash&lang=es

Recursos de interés

- Proyecto Thoth
  - http://www.criptored.upm.es/thoth/