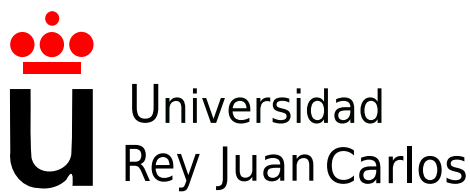
Herramientas criptográficas

Enrique Soriano

LS, GSYC

26 de enero de 2018



Términos

- Confidencialidad.
- Autenticación.
- Integridad.
- No-repudio.

Esteganografía

- esteganografía $\neq$ criptografía
- La esteganografía estudia la forma de ocultar la información dentro de otra información.
- No reemplaza el cifrado, pero lo puede hacer mucho más efectivo.
- Esteganografía + Cifrado = canal confidencial y encubierto.

Esteganografía

Ejemplo: modificación del bit de menos peso en un pixmap (LSB):

- Cuanto mayor es el mensaje oculto, más fácil es detectarlo. Conviene comprimirlo.
- Si se ponen en secuencia es más fácil detectarlos. Si los distribuimos tenemos que saber cómo lo hacemos (secreto compartido, semilla, etc.).
- La cubierta debe ser de alta resolución.
- La cubierta debe tener colores heterogéneos.
- Podemos distribuir la información en más de una cubierta.

Esteganografía

Otros métodos:

- Uso de sinónimos en el lenguaje natural en un mensaje de texto.
- Uso de zonas de fragmentación interna en bloques de disco.
- Uso de bloques no indexados en i-nodos.
- Uso de zonas "raras" en ciertos formatos de ficheros (p. ej. después del EOF en un fichero con estructura interna).
- Reordenado de instrucciones de código máquina.
- Cabeceras de protocolos (checksum, puerto de origen, etc.).
- ...

Criptografía: Términos y definiciones

- Criptografía: arte y ciencia de mantener las comunicaciones seguras.
- Criptografía + Criptoanálisis = Criptología
- Cifrar (E).
- Descifrar (D).
- Mensaje en claro o plaintext (M).
- Mensaje cifrado o ciphertext (C).
- Alice, Bob, Eve (eavesdropper), Mallory.

$$E(M) = C$$
$$D(C) = M$$
$$D(E(M)) = M$$

Cifrado: algoritmos restringidos

El segundo principio de Kerckhoffs:

"La efectividad del sistema no debe depender de que su diseño permanezca en secreto"

Si es restringido:

- No se puede usar en grupos grandes.
- Si alguien revela el secreto, hay que cambiar el algoritmo.
- Cada uno se tiene que inventar su propio algoritmo e implementarlo (software o hardware).
- Son menos seguros: no pueden ser validados por los expertos.

Cifrado: algoritmos no restringidos

- El cifrado y descifrado depende de las claves (K), el algoritmo es público.

$$E_K(M) = C$$
$$D_K(C) = M$$
$$D_K(E_K(M)) = M$$

Seguridad de los algoritmos

El criptoanálisis puede asumir ciertas condiciones para romper el cifrado (conseguir el mensaje sin conocer K). Los más comunes son:

- Ataque de texto cifrado: Eve únicamente conoce el texto cifrado.
- Ataque de texto plano conocido: Eve conoce varios mensajes en claro y cifrados.
- Ataque de texto plano elegido: Eve conoce un texto claro y cifrado de un mensaje que ella ha elegido.
- Ataque de texto plano elegido adaptativo: como el anterior, pero Eve es capaz de modificar el mensaje y ver cómo cambia el cifrado.
- Ataque de *side-channel*: Eve puede obsevar efectos laterales del algoritmo y deducir información.

Seguridad de los algoritmos

El resultado de un criptoanálisis puede resultar en (clasificación de Lars Knudsen):

- **Rotura completa:** ha encontrado K, tal que $D_K(C) = M$.
- **Deducción global:** ha encontrado un algoritmo alternativo A equivalente a $D_K(C) = M$, pero no sabe K.
- **Deducción local:** el atacante ha encontrado el texto claro de un texto cifrado concreto.
- **Deducción de información:** el atacante ha encontrado parte de la clave o del texto plano de un mensaje, pero no toda.
- **Distinción del algoritmo:** el atacante es capaz de distinguir datos cifrados de datos aleatorios.

Seguridad de los algoritmos

- **Algoritmo incondicionalmente seguro:** El texto cifrado no proporciona nada de información sobre el texto plano.
- **Algoritmo computacionalmente seguro:** teóricamente, el atacante puede romper el algoritmo, pero en la práctica no es computacionalmente viable. Son vulnerables a ataques de *fuerza bruta*.

Historia: algoritmos antiguos

- Sustitución: p.ej. el cifrado de César.

$$C_i = val((pos(M_i + K) \text{ mód } nelem(alfabeto))$$

  - Monoalfabética: sólo un mapeo $1 \rightarrow 1$.
  - Polialfabética: usa varios mapeos distintos $1 \rightarrow 1$
  - Homofónica: el mapeo es $1 \rightarrow N$.
  - Poligráfica: sustituye $N \rightarrow M$.
- Rotores: concatenación de sustituciones.
- Transposicionales: filas - columna.

Un ordenador actual rompe estos algoritmos en segundos.

Pseudo Random Function: PRF

Abstracción: Pseudo Random Function:

- Función determinista y eficiente que mapea dos conjuntos de datos X e Y en base a una clave de un conjunto K:

  $F : X \times K \mapsto Y$

- Es segura si la salida de F podría ser la salida **de cualquier función del conjunto de todas las funciones** de ese domino a ese rango $\rightarrow$ los datos de salida son computacionalmente indistinguibles de datos aleatorios.
- No requiere ser eficientemente invertible.

Pseudo Random Permutation: PRP

Abstracción: Pseudo Random Permutation

- Función determinista y eficiente que mapea de un conjunto de datos al mismo conjunto de datos, X, en base a una clave de un conjunto K:

  $F : X \times K \mapsto X$

- Cualquier PRP segura es una PRF segura
- Tiene que ser invertible.

One time pad (Vernam cipher)

- Cifrado perfecto, incondicionalmente seguro.
- K tiene que ser aleatoria.
- K no se puede reutilizar.
- La longitud de K es la longitud de M.

$$C_i = M_i \oplus K_i$$
$$M_i = C_i \oplus K_i$$

Algoritmos de clave simétrica

- Se usa la misma clave para cifrar y descifrar.

$$E_K(M) = C$$
$$D_K(C) = M$$
$$D_K(E_K(M)) = M$$

- Dos tipos: stream y bloques.

Clave simétrica: Stream cipher

- Inspirados en one-time-pad.
- Se usa un *keystream* para cifrar el mensaje.
- El *keystream* tiene que tener un periodo muy largo.
- Normalmente se usa como clave un secreto más un *nonce*.
- Son muy *maleables*. Conviene combinarlos con otro mecanismo de autenticación de mensajes (p. ej. MAC).
- Ejemplo: RC4. Presente en software actual (TLS/SSL, WEP...) pero se considera **inseguro** si el atacante tiene una gran capacidad de cómputo. La RFC 7465 prohibe el uso de RC4.

Clave simétrica: Block cipher

- Se cifra en bloques de un tamaño fijo. El mensaje claro tiene que tener un tamaño múltiplo del bloque.
- El tamaño de la clave y el tamaño de bloque determinan el cifrado.
- En general, se basan en una serie de rondas en las que se realizan permutaciones (P-box) y sustituciones (S-box) en base a la clave.
- La modificación de cualquier bit del bloque cifrado modifica todos los bits del bloque descrifrado.

Clave simétrica: Block cipher

Padding:

- Si el tamaño del mensaje no es múltiplo del tamaño del bloque, necesitamos completar el último bloque con datos de relleno.
- Hay que tratar con cuidado los errores de padding (*padding oracle attack*, etc.): se puede deducir el tamaño de los mensajes, ...
- Ejemplo, ANSI X.923: rellenar los bytes del final con ceros, menos el último, que indica el número de bytes del *padding*.
- Ejemplo, PKCS#7: rellenar con todos los bytes puestos al número de bytes del padding:

```
01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
etc.
```

En ambos ejemplos, si el tamaño del mensaje es múltiplo del tamaño de bloque, se añade un bloque nuevo con todos sus datos de relleno.

Clave simétrica: Block cipher

AES (Rijndael):

- Estándar actual.
- Bloque de 128 bits.
- La clave puede ser de 128, 192 o 256 bits. Se recomienda usarlas de 256 bits.
- 10, 12, y 14 rondas respectivamente.
- Es muy rápido y conveniente para máquinas poco potentes.
- Si dudas, usa este.

Clave simétrica: Block cipher

Triple-DES:

- Es una evolución de DES, antiguo estándar, que hoy se considera inseguro.
- Consiste en cifrar tres veces cada bloque con DES.
- Bloque de 64 bits.
- Las claves son de 56 bits (Op3, K, K, K) , 112 bits (Op2, K_1, K_2, K_1), o 168 bits (Op1, K_1, K_2, K_3).
- En total aplica 48 rondas de DES (16 rondas * 3).
- La opción 1 se considera segura.

Clave simétrica: Block cipher

Otros:

- Blowfish: Bloque de 64 bits. Clave es de longitud variable, de 1 a 448 bits, y tiene 16 rondas.
- Twofish: finalista AES, es una evolución de blowfish. Bloque de 128 bits, claves de 128, 192 o 256 bits.
- CAST5 (o CAST-128): Usado en GPG. Bloque de 64 bits, claves de 40 a 128 bits. 12 o 16 rondas.

Clave simétrica: modos de operación

- Hay muchos distintos: ECB, CBC, CTR, CFB, OFB ...
- Lo importante es **estar seguro de que estamos usando bien el modo de operación**.

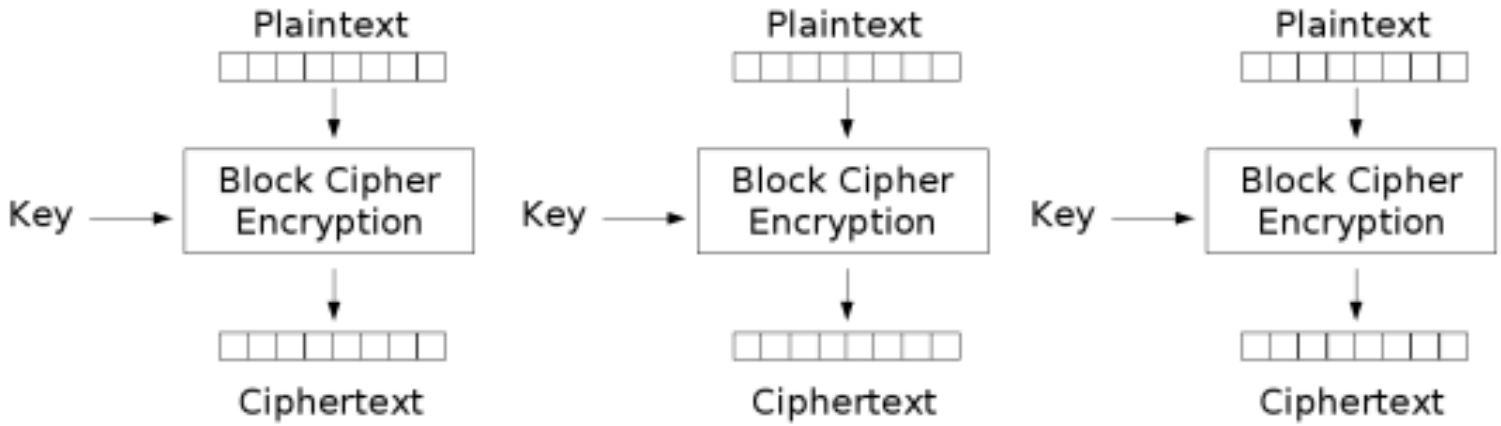
Clave simétrica: ECB

Modo Electronic Code Book (ECB):

- Cada bloque se cifra con la clave K.
- Dos bloques claros iguales tienen el mismo bloque cifrado.
- La modificación/eliminación de un bloque cifrado no afecta a otros.
- Permite cifrado/descifrado en paralelo.

Clave simétrica: ECB

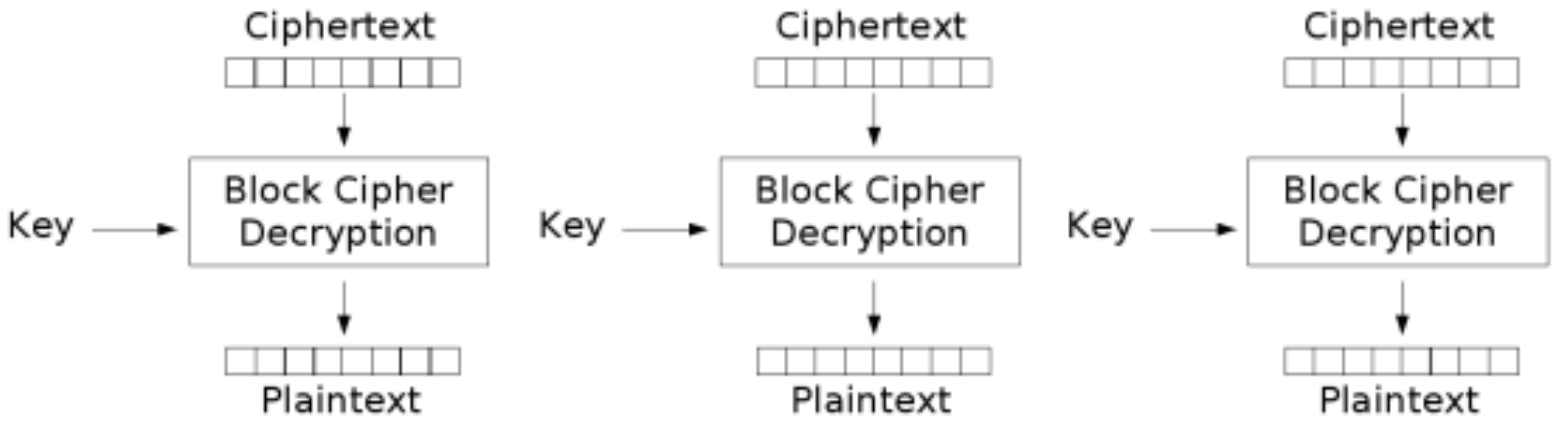
Cifrado:



Electronic Codebook (ECB) mode encryption

Clave simétrica: ECB

Descifrado:



Electronic Codebook (ECB) mode decryption

Clave simétrica: ECB

No es buena idea usar este modo:

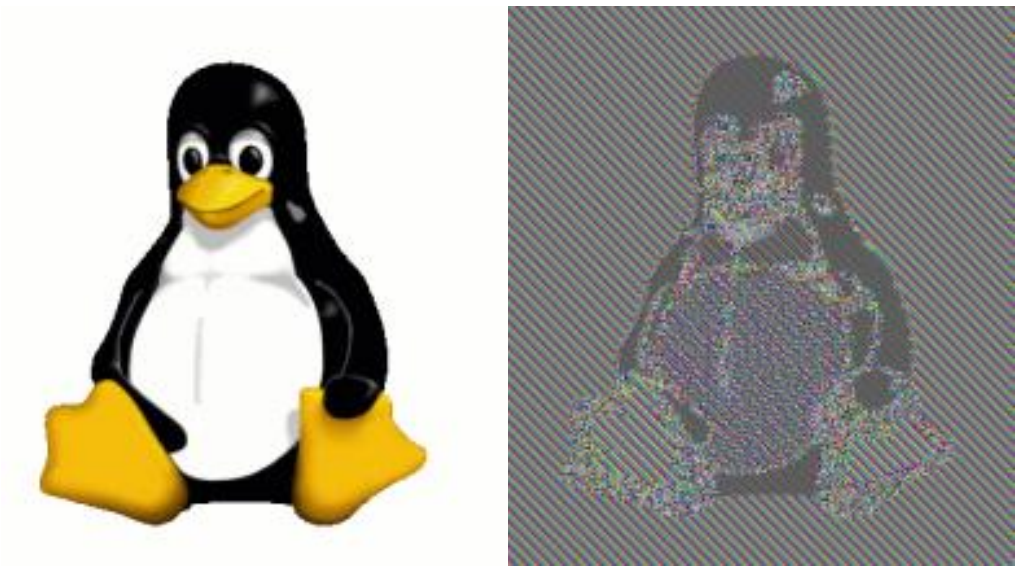


Figura: Cifrado ECB de un pixmap[1].

[1] Imagen ©Larry Ewing

Clave simétrica: CBC

Modo Cipher Block Chaining (CBC):

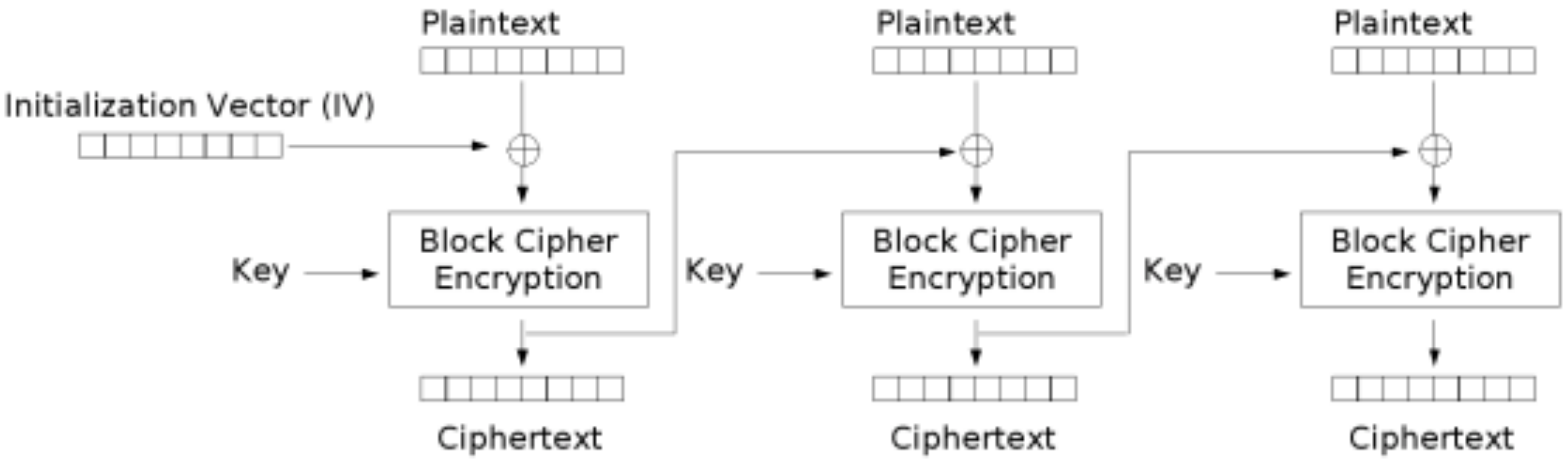
- Los bloques se encadenan, dependiendo el cifrado de un bloque del bloque anterior.

$$C_i = E_K(M_i \oplus C_{i-1})$$

- Para el primer bloque se usa el vector de inicialización (IV).

Clave simétrica: CBC

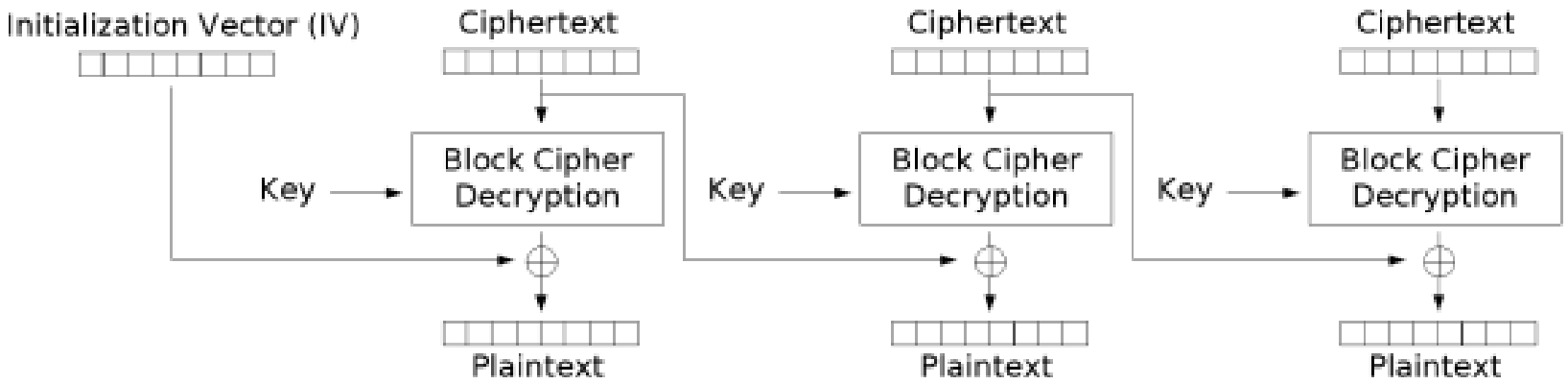
Cifrado:



Cipher Block Chaining (CBC) mode encryption

Clave simétrica: CBC

Descifrado:



Cipher Block Chaining (CBC) mode decryption

Clave simétrica: CBC

Modo Cipher Block Chaining (CBC):

- El IV se puede enviar en claro.
- El IV no se debe repetir.
- El IV debe ser lo más aleatorio posible (no predecible).

Clave simétrica: CBC

Modo Cipher Block Chaining (CBC):

- No permite paralelizar el cifrado (sí el descifrado).
- Un error muy común es pensar que proporciona integridad debido a la propagación del error.
- Permite modificaciones peligrosas de los bloques.
- Junto con la necesidad de padding, es peligroso (*padding oracle attack*).

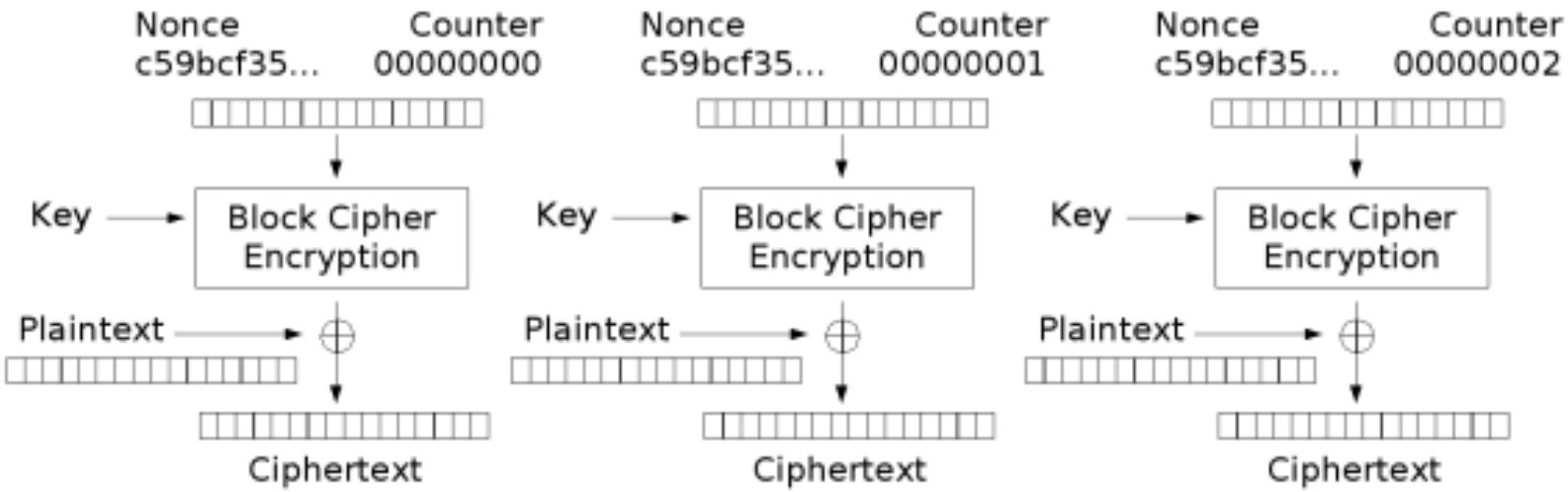
Clave simétrica: CTR

Modo Counter (CTR):

- Todavía es menos popular que CBC, pero es un posible candidato para reemplazarlo y actualmente se recomienda su uso.
- Convierte el cifrado de bloques en cifrado de stream. Otros modos como OFB o CFB también hacen esto.
- Se cifra el IV (o *nonce*) para conseguir un bloque de keystream.
- De nuevo: **no** se puede reutilizar el IV. Que el IV sea predecible no es un problema (evidentemente).

Clave simétrica: CTR

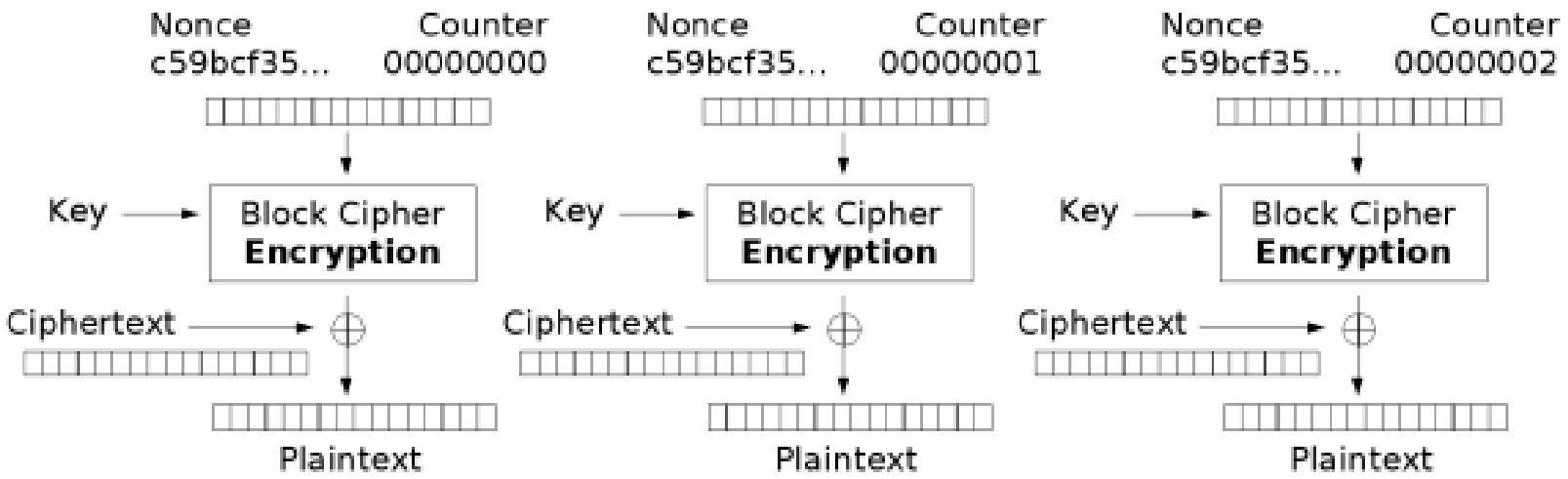
Cifrado:



Counter (CTR) mode encryption

Clave simétrica: CTR

Descifrado:



Counter (CTR) mode decryption

Clave simétrica: CTR

Modo Counter (CTR):

- No requiere *padding*: dejamos de usar el *keystream* cuando llegamos al final del mensaje.
- Permite cifrar/descifrar en paralelo.
- Igual de rápido que CBC.
- ¡Cuidado! como CBC, es *maleable*: modificando un bit del texto cifrado se modifica dicho bit en el texto claro.

Clave simétrica: Authenticated Encryption

- Modos en los que se proporciona confidencialidad e integridad: CCM, EAX, GCM, OCB.
- Authenticated Encryption With Associated Data (AEAD): se proporciona confidencialidad e integridad del texto plano e integridad de unos datos adicionales que van en claro. P. ej. GCM.
- Recomendados. Poco a poco están reemplazando a los modos tradicionales como CBC.

Clave simétrica: GCM

GCM (Galois/Counter Mode). Entrada de la función de cifrado:

- El texto plano.
- Datos adicionales (AAD). El algoritmo protege su integridad pero **no** protege su confidencialidad.
- El vector de incialización (IV). Debe ser unico, nunca se debe reusar. Se puede enviar en claro.

Salida de la función de cifrado:

- El texto cifrado.
- La etiqueta de autenticación o *tag*. La longitud es configurable (128, 120, 112, 104 o 96 bits).

Clave simétrica: GCM

Entrada de la función de descifrado:

- El vector de inicialización (IV).
- Los datos adicionales (AAD).
- El texto cifrado.
- La etiqueta de autenticación.

Salida de la función de descifrado:

- El texto claro.
- Error en caso de violación de la integridad del texto claro o la etiqueta de autenticación.

¿Comprimir antes de cifrar?

- Intuitivamente, parece una buena idea:
  - Elimina redundancia en la entrada del cifrado.
  - Hace que el texto plano no sea directamente legible.
- ... **pero** permite ataques de side-channel, por ejemplo:
  - Diferentes lenguajes naturales se comprimen con diferente ratio.
  - Diferentes mensajes del mismo tamaño generan salidas de distinto tamaño.
  - Ataque de texto plano parcialmente elegido: permite comprobar si el resto del mensaje contiene una cadena dada.

Intercambio de claves

- Clave simétrica: K tiene que ser un secreto compartido.
- ¿Si no hemos establecido K por un canal seguro?

Diffie-Hellman

- Permite negociar una clave entre dos partes anónimas (*Key Agreement Protocol*).
- **No proporciona autenticación**.

Diffie-Hellman

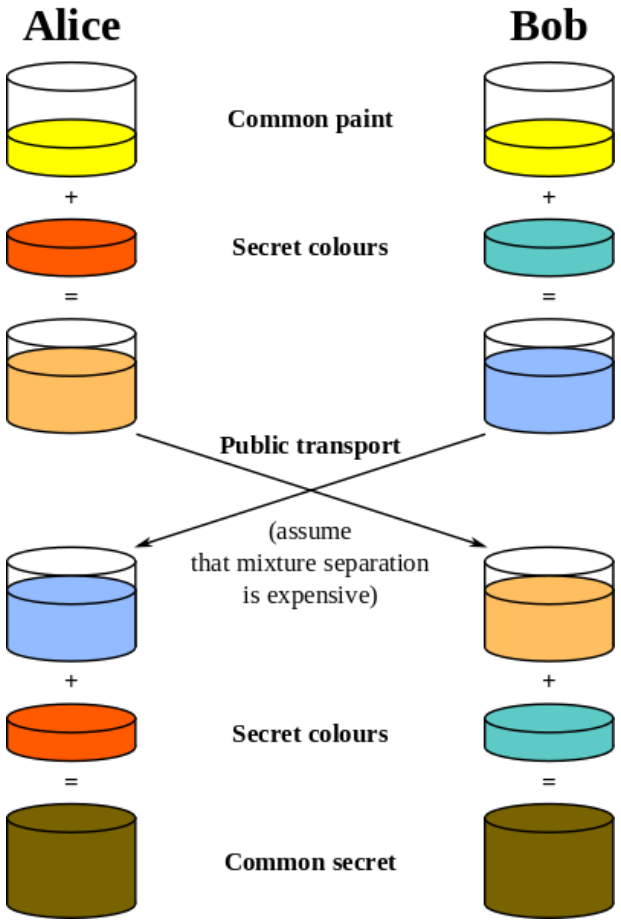


Figura: Idea de DH.[2]

[2] Imagen Wikimedia Commons, public domain.

Diffie-Hellman

1. Alice genera un número secreto a, y dos números p y g.
2. Alice $p, g \rightarrow$ Bob
3. Alice calcula $A = g^a$ mód p.
4. Alice $A \rightarrow$ Bob
5. Bob genera un número secreto b.
6. Bob calcula $B = g^b$ mód p.
7. Alice $\leftarrow B$ Bob
8. Alice calcula $s = B^a$ mód p
9. Bob calcula $s = A^b$ mód p

Diffie-Hellman

Los números se tienen que elegir cuidadosamente. Entre otras cosas, tienen que cumplir:

- p debe ser un número primo grande.
- a y b deben ser enteros grandes.
- El número g es una raíz primitiva módulo p: para cualquier número j primo entre sí con p [3], hay un entero k tal que

$$g^k \equiv j \pmod{p}$$
esto es,
$$g^k \mod p = j \mod p$$

- Para Eve, es computacionalmente imposible calcular:

$$s = B^a \mod p = A^b \mod p$$

[3] j y p son primos entre sí (coprimos) si no tienen ningún divisor común excepto 1 y -1

Diffie-Hellman

- DH sigue siendo seguro, pero...
- 2015: Logjam Attack. Aprovecha una vulnerabilidad en la implementación de DH de TLS para hacer un *downgrade attack* mediante un *Man in the middle* para usar claves débiles.
- Las implementaciones populares usan los mismos números primos para DH. Un adversario con la suficiente capacidad de cálculo puede precomputar la mayor parte del ataque para esos números primos populares.
- Solución: configurar DH con números primos de 2048 bits o más.

Algoritmos de clave asimétrica

- Se usan distintas claves para cifrar y descifrar.

$$E_{K1}(M) = C$$
$$D_{K2}(C) = M$$
$$D_{K2}(E_{K1}(M)) = M$$

Algoritmos de clave pública

- Es un esquema de clave asimétrica, con una clave pública y otra privada.
- *Par de claves*:
  - Pública: la conoce todo el mundo.
  - Privada: es un secreto que sólo conoce el propietario.
- Lo que cifra la clave pública lo descifra la privada, y viceversa.
- Algoritmos más usados: RSA, Elgamal, ECIES (Elliptic Curve Integrated Encryption Scheme).

Clave pública: RSA

Su seguridad reside en el tiempo que requiere factorizar un número muy largo. Para generar el par de claves hay que tener en cuenta (entre otras cosas):

- Se escogen dos primos largos **distintos** p y q (> 200 cifras).
- $n = p * q$
- El n que se usa actualmente (i.e. la *longitud* de la clave RSA) va de 1024 a 4096 bits, no se recomienda menos de 2048 bits. Cada vez que se dobla la longitud de la clave, descifrar es 6-7 veces más lento.
- $\phi = (p - 1) * (q - 1)$
- Se escoge e, tal que e y ϕ son primos entre sí (coprimos). No debe ser muy grande. Se suele usar 65537 (número 4 de Fermat).
- Se escoge d tal que

  $e * d \equiv 1 \pmod{\phi}$
- Clave pública: (e, n)
- Clave privada: (d, n)

Clave pública: RSA

- Se divide el mensaje en bloques menores que n.
- Cifrado: $c = m^e$ mód n
- Descifrado: $m = c^d$ mód n
- No se suele usar para cifrar más de un bloque (una hash, una clave de sesión, etc.).
- En RSA, el padding es muy importante para alargar el mensaje si es lo suficientemente pequeño y aleatorizar la entrada (que no se cifre siempre igual el mismo mensaje).
- Se recomienda usar OAEP: no es *padding*, es *armoring*: aplica permutaciones en el texto plano y añade ruido pseudoaleatorio. Es una transformación **reversible** para *agitar* los datos.

Clave pública: RSA

Curiosidades:

- Viejo pero seguro (con clave suficientemente larga).
- Hay números que no se cifrarán. **La cantidad es despreciable dentro del módulo**.
- Hay más de una clave privada para una clave pública, pero **no es factible encotrarlas**.

¿Simétrica o asimétrica?

- Ninguna nos proporciona la integridad de los datos transmitidos (excepto modos autenticados como GCM).
- Son distintas, cada una tiene un propósito.
- Simétrica: rápida. RSA es $\approx$100 veces más lento que DES.
- Pública: facilidad para la distribución de claves.
- Recomendación: aplicar un esquema híbrido, usar el algoritmo de clave pública para enviar una clave de sesión para un algoritmo simétrico.

Secret Sharing

Secret Sharing: Consiste en compartir un secreto M entre N participantes, de tal forma que sea necesario que al menos T (threshold) participantes estén de acuerdo para recuperar dicho secreto. P. ej. requerir el acuerdo de T personas para conocer una contraseña.

- Ejemplo: SSSS (Shamir's Secret Sharing Scheme)
- Es **incondicionalmente seguro**: sin T *shadows*, no puedes recuperar el mensaje M. No es posible hacer un ataque de fuerza bruta.
- Para proteger secretos largos, se cifra una clave de un algoritmo de cifrado simétrico (p. ej. AES) y se cifra el secreto largo con ella.

Cifrado homomórfico

Cifrado homomórfico:

- Permite realizar operaciones sobre el texto cifrado. Son intrínsicamente maleables.
- La operación da el mismo resultado que si se hace sobre el texto claro.
- Ventaja: se puede realizar una operación sobre los datos manteniendo su confidencialidad $\rightarrow$ muy interesante para Cloud computing, sistemas de salud, etc.
- Algunos algoritmos convencionales tienen operaciones con esta propiedad (homomorfismo parcial). P. ej. en RSA:

  $$E_K(M1) * E_K(M2) = E_K(M1 * M2)$$

- FHE (Fully Homomorphic Encryption): cuando el criptosistema soporta computaciones arbitrarias sobre el texto cifrado. Todavía en desarrollo.

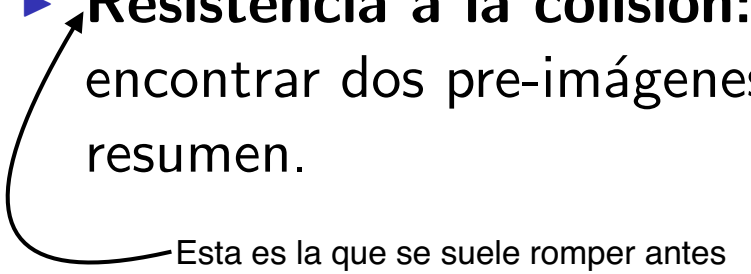
Resúmenes (HASH)

- A partir de cualquier cantidad de datos, generan un resumen de un tamaño fijo.
- Cualquier modificación en la entrada modifica todo el resumen.
- Funciones de un único sentido.
- Del dominio a la imagen es muy rápida.
- A la inversa llevaría millones de años.
- Se usan para proporcionar integridad a los datos.

Resúmenes (HASH)

Propiedades de una hash segura:

- **Primera resistencia pre-imagen:** Para cualquier resumen dado, no es computacionalmente posible encontrar la pre-imagen que los genera.
- **Segunda resistencia pre-imagen:** No es computacionalmente posible encontrar una pre-imagen que genere el mismo resumen que otra pre-imagen dada.
- **Resistencia a la colisión:** No es computacionalmente posible encontrar dos pre-imágenes cualquiera que generen el mismo resumen.



Resúmenes (HASH)

Resistencia ante ataques de fuerza bruta:

- Si una función hash tiene un número N de posibles imagenes, para encontrar una pre-imagen que genere un hash dado con una probabilidad de 0,5 hace falta realizar $\approx 2^{N-1}$ hashes.
- Un AntMiner S9 realiza 5800 Mhash/s. Cuesta \$2400.

Resúmenes (HASH)

Resistencia a la colisión:

- Ataque de cumpleaños: "Si la imagen tiene N posibles valores, se espera encontrar dos valores M1 y M2 cualesquiera tal que

  $H(M1) = H(M2)$

  evaluando la función H sobre

  $1{,}2\sqrt{N}$

  valores distintos."

Resúmenes (HASH)

Ataque de colisión con prefijo elegido (*prefix collision attack*):

- Dados dos prefijos P1 y P2, encontrar dos mensajes arbitrarios M1 y M2 tal que:

  $$H(P1||M1) = H(P2||M2)$$

- Mucho más peligroso que un ataque de colisión. P. ej. dos ficheros binarios con distintas instrucciones y basura al final, pero con la misma hash.

Resúmenes (HASH)

Las más usadas:

- MD5: resumen de 128 bits. No se considera segura. Ataque de colisión de 2^{24} operaciones (unos cuantos segundos). Ataque de colisión con prefijo elegido de 2^{50} operaciones. En algunos sítios se sigue usando (!!!).
- SHA-1: resumen de 160 bits. Empieza a tener problemas: Ataque de colisión de 2^{51} operaciones. Familia diseñada por la NSA y sujeta a patentes.
- SHA-2: resúmenes de 224, 256, 384 o 512 bits. Segura.
- SHA-3: resúmenes de 224, 256, 384 o 512 bits. Segura.
- RIPEMD: resúmenes de 128, 256 y 360 bits. Segura y abierta.

Códigos de autenticación (MAC)

- Es un resumen dependiente de una contraseña.
- Aseguran la integridad y autenticidad de un mensaje.
- No proporciona no-repudio, ya que K es un secreto compartido.
- Una forma (bastante lenta) de generarlo:

$$H(E_K(M))$$

- Por ejemplo, HMAC-SHA1 es[4]:

$$MAC_K(M) = H(\ \ (K \oplus const1)\ \ ||\ \ H((K \oplus const2)||M)\ \ \ \ )$$

[4] const1 y const2 son dos constantes.

Códigos de autenticación (MAC)

- Concatenar la clave con los datos es **inseguro**:

  $MAC_K(M) = H(K||M)$
  $MAC_K(M) = H(M||K)$

- SHA1, SHA2, MD5, etc. son vulnerables ataques de *Key Length Extension* que permiten generar MACs válidas para mensajes derivados de un mensaje conocido, **sin saber la clave**.
- Moraleja: usa una MAC estándar, como HMAC-SHA1.

Códigos de autenticación (MAC)

Se deben usar dos claves distintas: K_e para cifrar y K_m para generar la MAC. Se suele hacer de tres formas distintas:

- *Encrypt-and-MAC* (p. ej. SSH), no recomendada (insegura con algunos algoritmos). Sin saber K_e no se puede comprobar la integridad del texto cifrado:

$$E_{K_e}(M) \,||\, MAC_{K_m}(M)$$

- *MAC-then-Encrypt* (p. ej. SSL), no recomendada (insegura con algunos algoritmos). Sin saber K_e no se puede comprobar la integridad del texto cifrado:

$$E_{K_e}(M||MAC_{K_m}(M))$$

- *Encrypt-then-MAC* (p. ej. IPSEC), **es segura y se considera un estándar actualmente**. Se puede comprobar la integridad del texto cifrado sin conocer K_e:

$$E_{K_e}(M) \,||\, MAC_{K_m}(E_{K_e}(M))$$

Firma digital

- Basadas en algoritmos de clave pública. Básicamente:

  $E_{K_{priv}}(H(M))$

- Hay distintos algoritmos: RSASSA-PKCS1 , RSASSA-PSS, DSA, EC-DSA...

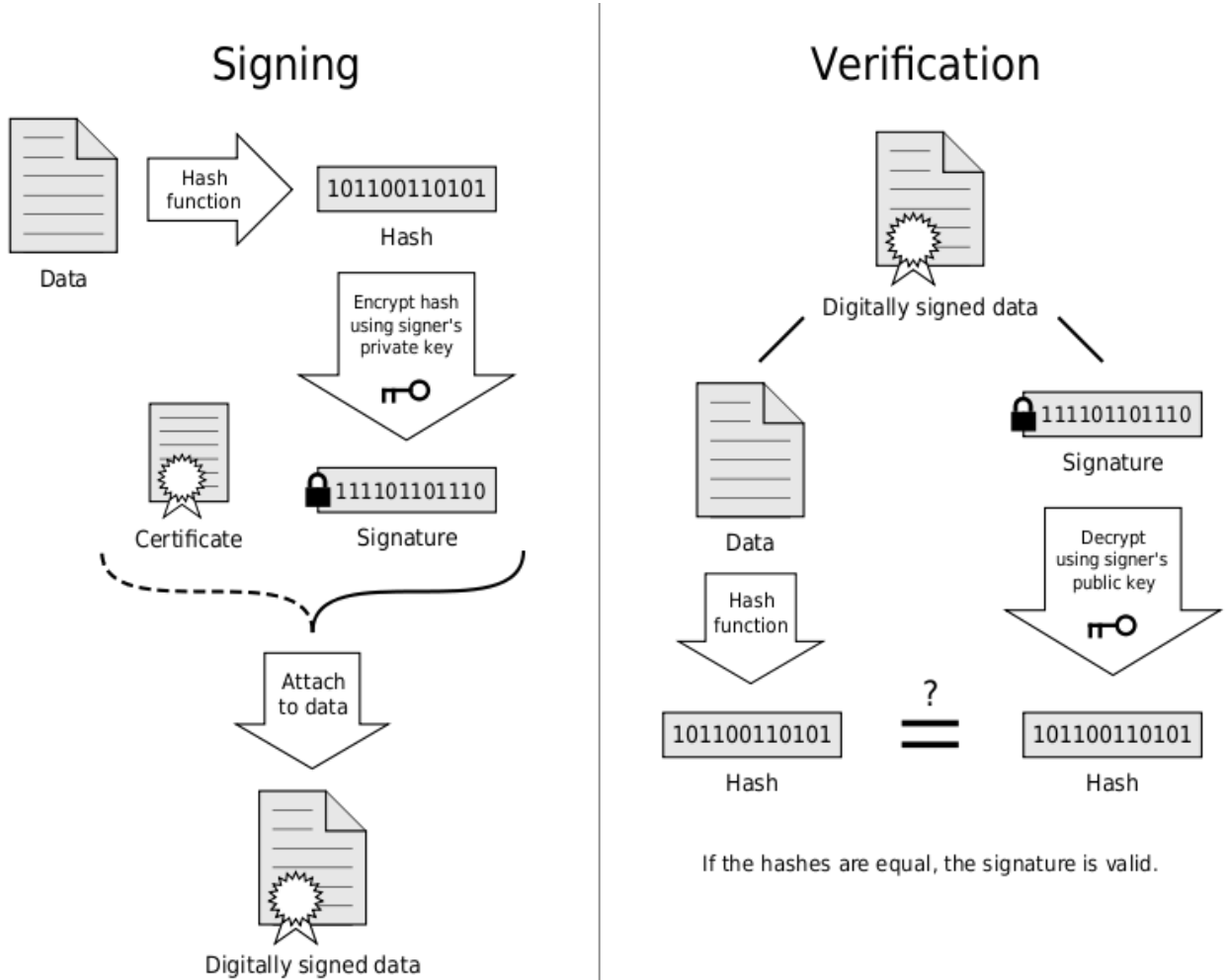


Figura: Firma digital[5].

[5] Imagen (cc) acdx, Wikimedia Commons.

Ejemplo de firma digital: RSASSA-PKCS1 V1.5.

Con una clave RSA de longitud Len:

- $T = IDhash||H(M)$ siendo $IDhash$ el tipo de hash usada (19 bytes para $IDhash$).
- PS es una cadena de bytes, todos con valor 0xFF, de longitud $Len - len(T) - 3$.
- Se genera:

  $EM = 0x00||0x01||PS||0x00||T$

- La firma es:

  $E_{K_{priv}}(EM)$

Firma digital

- Las hay deterministas y no deterministas (para un mismo mensaje se generan firmas distintas cada vez, p. ej. RSASSA-PSS).
- La firma proporciona autenticación e integridad:
  - No es falsificable.
  - No es reusable.
  - No es alterable.
  - No es repudiable.
- Una firma puede estar a la vez firmada por otros: **contrafirma**.

PKI X.509

- Problema: distribuir una clave pública garantizando que pertenece al sujeto deseado y no a otro sujeto.
- **Certificado digital**: documento que incluye la clave de un sujeto y que está firmado por un sujeto del que nos fiamos (CA).
- **Autoridad Certificadora (CA)**: firma el certificado, dando fe de que la clave pública que incluye pertenece al sujeto.
- Estándar: PKCS7.

PKI X.509

- Los certificados de las claves públicas de las CA se llaman **certificados raíz**.
- Los certificados raíz vienen preinstalados en el software o son fácilmente cotejables. Pueden estar autofirmados.
- A partir del certificado raíz se establece una cadena de confianza.
- PKI sigue un esquema jerárquico.

PKI X.509

Un certificado X.509 contiene:

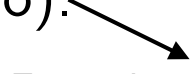
- Versión
- Número de serie
- Algoritmo de cifrado
- Emisor, que es un nombre x.500 (rfc1779) que tiene atributos tales como el país (C), estado (ST), nombre común (CN), organización (O), e-mail, localidad (L), etc.
- Validez (desde-hasta)
- Sujeto
- Clave pública del sujeto (algoritmo y clave)
- Id del Emisor (opcional)
- Id del sujeto (opcional)
- ...(extensiones)...
- Algoritmo de la firma
- Firma

PKI: certificados X.509

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   ...
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
       d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
       0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
       ...
```

PKI: certificados X.509

- Para revocar un certificado se usaban listas negras (CLR).
- Ahora se utiliza el protocolo OCSP (rfc 2560): se pregunta el status de un certificado a un servidor OSCP (*responder*, normalmente un servidor de la autoridad emisora del certificado) a través de HTTP. La respuesta está firmada por el servidor y puede ser:
  - good: está ok.
  - revoked: está revocado.
  - unknown: el servidor no sabe nada sobre ese certificado.
- La fecha de caducidad no tiene en cuenta el tiempo en el que la longitud de clave usada puede ser segura (ojo).



Esto quiere decir que podría caducar antes (debido a vulnerabilidades)

Web of trust: OpenPGP

- Esquema no jerárquico para distribuir claves públicas.
- Los usuarios firman las claves del resto de usuarios en los que confían, en reuniones, etc.
- Cada usuario guarda las claves públicas firmadas del resto de los usuarios en su **anillo de claves**.
- El usuario asigna cierto nivel de confianza las claves públicas que adquiere.
- La clave es válida/inválida dependiendo del número de firmas que tenga y de la confianza que asignamos a dichas firmas.

GPG

Nivel de confianza:

- Unknown: no se conoce nada sobre el sujeto emisor.
- None: no se confía en el sujeto.
- Marginal: se confía en el sujeto.
- Full: se confía totalmente en el sujeto.
- Ultimately: tus propias claves.

GPG

Ejemplo: la clave se considera válida si se cumple alguna de estas condiciones[6]:

- Está firmada por nosotros mismos.
- Está firmada por por X claves con confianza completa (firmadas por nosotros).
- Está firmada por por Y claves con confianza marginal (no firmadas por nosotros).

[6] Depende de la configuración y de la versión.

Números aleatorios: PRNG

- Es muy difícil generar números realmente aleatorios (distribución uniforme y no predecibles).
- Un generador de números pseudo-aleatorios (PRNG) genera números que **no** son aleatorios, pero lo parecen: son pseudo-aleatorios.
- Los números pseudo-aleatorios pasan cualquier test estadístico de aleatoriedad.
- Un PRNG se inicializa con un valor, la **semilla**. La secuencia de números generados es siempre la misma para una **semilla** dada.

Es un error inicializar con la misma semilla

Números aleatorios: PRNG

- Debemos reiniciar el generador constantemente con nuevas semillas (con la mayor entropía que podamos conseguir) $\rightarrow$ hay que evitar que se pueda adivinar o alterar el estado actual del generador.
- Hay que tener cuidado con los generadores por omisión (p. ej. Python, Java, .NET). Muchas veces no son criptográficamente seguros.

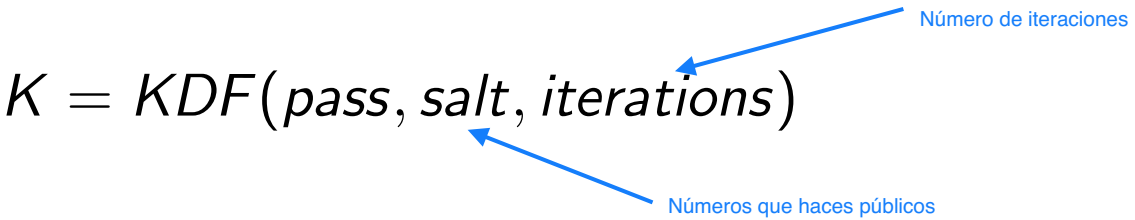
Cuidado con los generadores de números pseudo aleatorios

Números aleatorios: CSPRNG

- Un generador de números pseudo-aleatorios criptográficamente seguro (CSPRNG) crea una secuencia no predecible ni reproducible de forma fiable.
- Hardware random number generator (HWRNG): hardware especial para generar entropía (p.ej. los usados en máquinas tragaperras, etc.).
- `/dev/random` y `/dev/urandom` en Linux son seguros y ambos usan un CSPRNG implementado en el kernel.

Key Derivation Function (KDF)

- Lentas a propósito (*iterations*) para evitar ataques.



- La sal tiene dos objetivos:
  - Hacer que dos contraseñas iguales no generen la misma clave.
  - Evitar ataques con valores precalculados.
- A veces la sal se borra para obligar a todos a romperla con fuerza bruta, para que tarde más (*key strengthening*).
- Ejemplos: Bcrypt, PBKDF2, scrypt, S2K (GPG).

Cifrado de discos

- Puede ser:
  - Disco completo (FDE, Full Disk Encryption): cifrado a nivel del dispositivo de bloques. P. ej.: BitLocker (Windows), FileVault2 (OSX), LUKS/dm-crypt (Linux), etc.
  - Parcial: implementado en el sistema de fichero. P. ej. EFS (Windows), eCryptfs (Linux), etc.
- Puede estar asistida por hardware específico (TPM).

Hardware: Trusted Platform Module (TPM)

- Hardware dedicado que implementa las herramientas criptográficas y almacena secretos.
- Incluye un procesador para generar claves, cifrar/descifrar y un CSPRNG.
- El procesador tiene una par de claves RSA "quemado" de fábrica.
- Sirve de *root of trust* para el arranque.
- Objetivos: comprobar la integridad del sistema, FDE, proteger passwords, protección de derechos de autor y licencias, ...

Hardware: Hardware Security Module (HSM)

- Cajas dedicadas a tareas criptográficas, almacenamiento de secretos y gestionar el ciclo de vida de las claves.
- Su hardware y su software están totalmente dedicados a eso.
- Algunos te dejan desplegar tu propio software dentro, otros no.
- Los hay especializados en pagos, en red, USB, etc.