Capítulo 8
Seguridad de red

Redes de computadores
Bloque 4

Redes de computadoras: Un enfoque descendente,
5ª edición.
Jim Kurose, Keith Ross
Pearson Educación,
2010.

A note on the use of these ppt slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:
- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Capítulo 8: Seguridad de red

Objetivos:

- comprender los principios de la seguridad de red:
  - criptografía: más allá de la "confidencialidad"
  - autenticación
  - integridad de mensajes
- seguridad en la práctica:
  - cortafuegos, sistemas de detección de intrusiones
  - seguridad en las capas de aplicación, transporte, red, enlace

Capítulo 8: hoja de ruta

8.1 ¿Qué la seguridad de red?
8.2 Principios de criptografía
8.3 Integridad de mensajes
8.4 Conexiones TCP seguras: SSL
8.5 Seguridad en la capa de red: IPsec
8.6 Seguridad en redes inalámbricas

¿Qué es la seguridad de red?

Confidencialidad: el mensaje solo deben "entenderlo" el emisor y el receptor
- el emisor cifra el mensaje...
- ...y el receptor lo descifra

Autenticación: confirmación mutua de identidad emisor-receptor

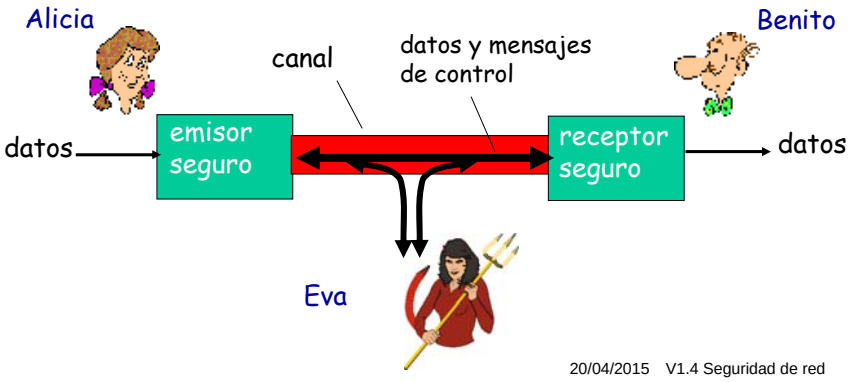
Integridad: emisor y receptor quieren asegurarse de que el mensaje no ha sido alterado sin que ellos se den cuenta

No-repudio: Un firmante no puede después negar que ha firmado (= repudiar su firma)

Seguridad operacional: A nivel corporativo, una organización quiere mantener su red asegurada

Amigos y enemigos: Alicia, Benito y Eva

- Alicia y Benito quieren comunicarse "confidencialmente"
- Eva (intruso) puede interceptar, añadir o borrar mensajes



Hay malotes por ahí...

P: ¿Qué pueden hacer "los malos"?

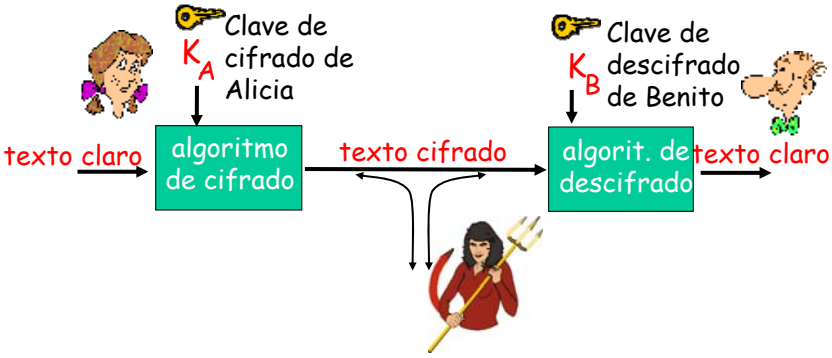
R: Bastante, por ejemplo:

- *escuchar:* interceptar mensajes
- *insertar:* activamente mensajes en el canal
- *suplantar:* falsificar una dirección de emisión (o cualquier otro campo) en un paquete
- *intermediar:* eliminar al emisor o al receptor de una conexión activa y tomar su puesto, suplantándolo
- *denegación de servicio:* impedir la prestación de un servicio (p. ej., sobrecargando el servidor)

Capítulo 8: hoja de ruta

8.1 ¿Qué la seguridad de red?
8.2 Principios de criptografía
8.3 Integridad de mensajes
8.4 Conexiones TCP seguras: SSL
8.5 Seguridad en la capa de red: IPsec
8.6 Seguridad en redes inalámbricas

La jerga criptográfica…



m mensaje en claro
$K_A(m)$ texto cifrado con la clave K_A
$m = K_B(K_A(m))$

Esquema simple de cifrado

Cifrado por sustitución: sustituir una cosa por otra.

- cifrado monoalfabético: sustituir una letra por otra

```
texto claro:  abcdefghijklmnopqrstuvwxyz
                |                       |
texto cifrado: mnbvcxzasdfghjklpoiuytrewq
```

Ejemplo:

```
texto claro: te quiero. alicia
texto cifrado: uc pyscok. mgsbsm
```

Clave: la correspondencia desde el conjunto superior, de 26 letras, con el inferior, también de 26 letras

Cifrado polialfabético

- *n* cifradores monoalfabéticos, $M_1, M_2, \ldots, M_n$
- Se toman repetidamente, según cierta secuencia:
  - p. ej. $n = 4$, M_1, M_3, M_4, M_3, M_2; M_1, M_3, M_4, M_3, M_2;
- Para cada símbolo en el texto claro (=letra), usar el siguiente cifrador monoalfabético:
  - *sol*: *s* de M_1, *o* de M_3, *l* de M_4
- Clave: los cifradores monoalfabéticos y la secuencia

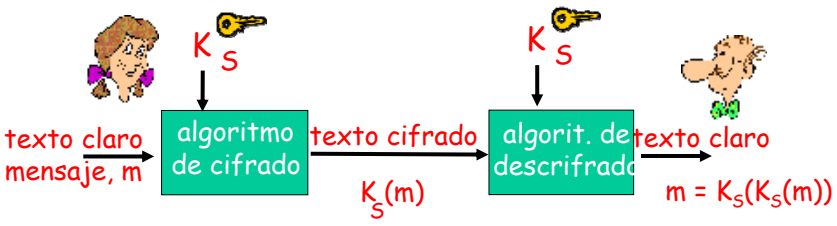
Ataques a los esquemas de cifrado

- Ataque al texto cifrado conocido:
  - Eva tiene el texto cifrado y puede analizarlo.
- Dos métodos:
  - Probar todas las claves: (pero tiene que saber cuándo obtiene un texto plano válido…)
  - Análisis estadístico
- Ataque al texto plano conocido:
  - Eva sabe algún texto plano y el correspondiente cifrado.
  - ej., en el monoalfabético, Eva determina parejas tipo (a,m),(l,g),(i,s),…
- Ataque al texto plano elegido:
  - Eva puede obtener textos cifrados a partir de textos planos elegidos por ella.

Tipos de Criptografía

- El secreto está en la clave…:
  - Algoritmos públicamente conocidos
  - Solo las claves son secretas
- Criptografía de clave pública
  - A base de dos claves
- Criptografía simétrica
  - Se usa una única clave
- Funciones resumen (*hash*)
  - No se usan claves
  - No hay nada secreto… ¡y es útil a pesar de todo!

Criptografía de clave simétrica



clave simétrica: Benito y Alicia comparten la misma clave (simétrica) K_S

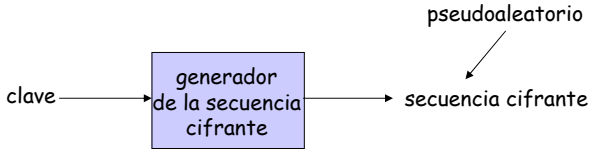
- ej., la clave es saber cuál es el esquema de sustitución en el cifrado monoalfabético

P: ¿Cómo se ponen de acuerdo en ello Benito y Alicia?

Dos tipos de cifrado simétrico

- Cifrado en flujo
  - se cifra de bit en bit
- Cifrado en bloque
  - Se trocea el texto claro en bloques de igual tamaño
  - Se cifra cada bloque como una unidad

Cifrado en flujo



- Combinar cada bit de la secuencia cifrante con un bit del texto claro para obtener un bit del texto cifrado
- m(i) = i-ésimo bit del mensaje
- ks(i) = i-ésimo bit de la secuencia cifrante
- c(i) = i-ésimo bit del texto cifrado
- Cifrar: $c(i) = ks(i) \oplus m(i)$ ($\oplus$ = "o" exclusivo, suma módulo 2)
- Descifrar: $m(i) = ks(i) \oplus c(i)$

Cifrado en bloque

- El mensaje en claro se divide en bloques de *k* bits (ej. de 64 bits).
- Se hace una biyección entre cada bloque de *k* bits del texto claro y un bloque de *k* bits del texto cifrado (es decir, una permutación)

Ejemplo con *k* = 3:

entrada	salida
000	110
001	111
010	101
011	100

entrada	salida
100	011
101	010
110	000
111	001

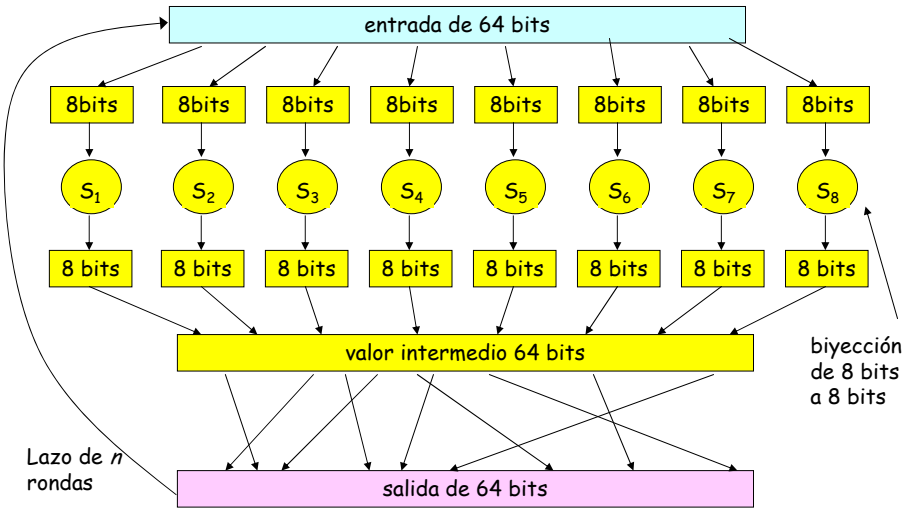
¿Cuál es criptograma correspondiente a 010110001111?

Cifrado en bloque

- ¿Cuántas biyecciones hay para $k = 3$?
  - ¿Cuántas entradas de 3 bits? $2^3 = 8$
  - ¿Cuántas permutaciones de las entradas?
  - Respuesta: 8! = 40.320 ; ¡no son tantas!
- En general, $2^k!$ posibles biyecciones; para $k = 64$ son muchísimas
- Problema:
  - Una tabla con 2^{64} entradas y cada entrada con 64 bits es inviable.
- Solución: usar una función que *simule* una tabla permutada aleatoriamente

Prototipo de función

Tomado de Kaufman et al



¿Por qué se hacen las rondas?

- Si solo hubiera una ronda, entonces cada bit de la entrada afectaría como máximo a 8 bits de la salida.
- En la 2ª ronda, los 8 bits afectados se esparcen por las "cajas-S".
- ¿Cuántas rondas?
  - Es lo mismo que para barajar bien las cartas
  - Menos eficiencia para *n* más alto.

Cifrar un mensaje largo

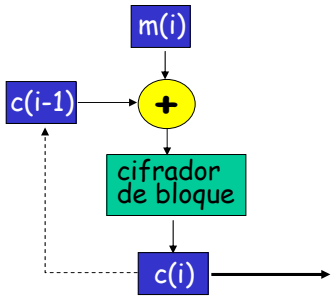
- ¿Por qué no dividir el mensaje en trozos y cifrar cada trozo independientemente?
  - Porque si el mismo bloque apareciera dos veces, recibiría el mismo criptograma: ¡inseguro!
- Qué tal si…
  - generamos un r(i) de 64 bits para cada bloque de texto claro m(i),
  - calculamos $c(i) = K_S(m(i) \oplus r(i))$,
  - transmitimos c(i), r(i), i=1,2,…
  - y, en recepción, $m(i) = K_S(c(i)) \oplus r(i)$.
  - Problema: ineficiente, hay que enviar c(i) y r(i)

Encadenado de bloques cifrados (CBC)

- CBC genera sus propios números aleatorios
  - el cifrado del bloque actual se hace depender del bloque cifrado anterior:
  - $c(i) = K_S(m(i) \oplus c(i-1))$
  - $m(i) = K_S^{-1}(c(i)) \oplus c(i-1)$ (¡ojo a los paréntesis!)
- ¿Y el primer bloque?
  - Usamos un vector de inicio (VI), aleatorio, = c(0)
  - VI no ha de ser secreto
- Cambiamos VI en cada mensaje o sesión
  - Así aseguramos que un mismo mensaje, enviado dos veces, produce un criptograma distinto.

Encadenado de bloques cifrados

- *CBC (cipher block chaining):*
  - se realiza el XOR del i-ésimo bloque de entrada, m(i) con el (i-1)-ésimo bloque de salida c(i-1), que es el cifrado correspondiente al bloque anterior
  - c(0) se transmite tal cual.



DES: cifrado simétrico (obsoleto)

DES: Data Encryption Standard

- Estándar en EEUU [NIST 1993]
- Clave simétrica de 56 bits, con bloques de 64 bits como entrada
- Cifrado en bloque con CBC
- ¿Es seguro DES?
  - Pues no: se ataca por fuerza bruta en menos de un día, aunque no hay ataques analíticos
- Un DES más seguro:
  - 3DES: en su forma más segura es cifrar tres veces con tres claves distintas:

  (concretamente: cifrar, descifrar, cifrar)

AES: Advanced Encryption Standard

- nuevo estándar de clave simétrica aprobado por el NIST, para reemplazar al DES
- procesa datos en bloques de 128 bits
- claves de 128, 192 o 256 bits
- si el ataque por fuerza bruta (probar todas las claves) tardara 1 segundo en DES, tardaría 149 billones de años en AES

Criptografía de clave pública

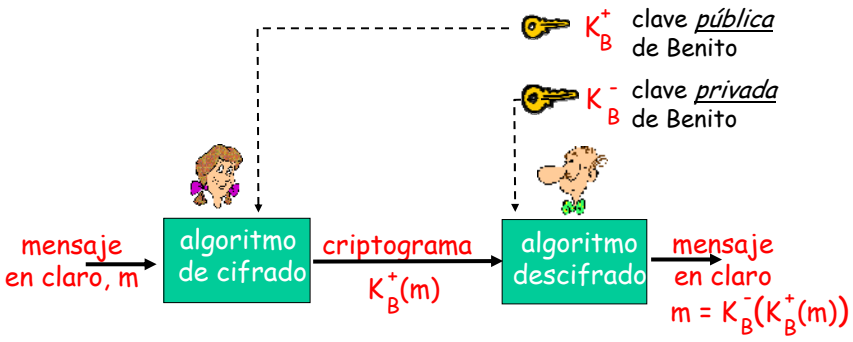
clave *simétrica*

- exige a emisor y receptor compartir la clave secreta
- P: ¿cómo ponerse de acuerdo en la clave para la primera vez? (¡y sin reunirse!)

criptografía de *clave pública*

- inventada por [Diffie-Hellman76, RSA78]
- emisor y receptor *no* comparten claves
- *clave pública*: conocida por todos
- *clave privada*: solo conocida por el receptor

Criptografía de clave pública



Algoritmos de cifrado de clave pública

Requisitos:

1. se necesita $K_B^+(\cdot)$ y $K(\cdot)_B^-$ tales que

$$K_B^-(K_B^+(m)) = m$$

2. dada una clave pública K_B^+, debe ser imposible calcular la clave privada K_B^-

RSA: algoritmo de Rivest, Shamir, Adleman

Pre-requisito: aritmética modular

- x mod n = resto de x al dividirlo por n
- Propiedades:

  [(a mod n) + (b mod n)] mod n = (a+b) mod n
  [(a mod n) - (b mod n)] mod n = (a-b) mod n
  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- Así pues:

  $(a \bmod n)^d \bmod n = a^d \bmod n$
- Ejemplo: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: Creación del par de claves pública/privada

1. Se eligen dos primos grandes, p, q. (p.ej. de 1024 bits cada uno)
2. Se calcula $n = pq$, $z = \varphi(n) = (p-1)(q-1)$
3. Se elige e (con $e < n$) coprimo con z.
4. Se elige d tal que: $ed \bmod z = 1$.
5. clave pública: (n,e), clave privada: (p,q,d).

K_B^+ K_B^-

RSA: Cifado, descifrado

0. Sean (n,e) y (n,d) calculados como se ha dicho
1. Para cifrar m ($< n$), calcular
   $c = m^e \bmod n$
2. Para descifrar c, calcular
   $m = c^d \bmod n$

funciona porque... $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

Ejemplo de RSA

Benito elige $p = 5$, $q = 7$. Así: $n = 35$, $z = 24$.
$e = 5$ (e, z son coprimos).
$d = 5$ ($ed - 1$ divisible por z).

Vamos a cifrar un mensaje de 8 bits:

	bits	m	m^e	$c = m^e \bmod n$
cifrado:	00001100	12	248832	17

	c	c^d	$m = c^d \bmod n$
descifrado:	17	1419857	12

¿Por qué funciona RSA?

- Hay que demostrar que $c^d \bmod n = m$ donde $c = m^e \bmod n$
- Teorema (de Euler):
  - para todo x e y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - con $n = pq$ and $z = (p-1)(q-1)$
- En nuestro caso,

$$c^d \bmod n = (m^e \bmod n)^d \bmod n$$
$$= m^{ed} \bmod n$$
$$= m^{(ed \bmod z)} \bmod n$$
$$= m^1 \bmod n$$
$$= m$$

RSA: otra propiedad importante

Lo siguiente será *muy útil* en el futuro:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{usar primero la clave pública, luego la privada}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{usar primero la clave privada, luego la pública}}$$

¡Sale lo mismo!

RSA: demostramos la propiedad

¿Por qué $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$?

Es, simplemente, la aritmética modular:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n\end{aligned}$$

¿Por qué es RSA seguro?

- A partir de la clave pública de Benito, (n,e), ¿qué tan difícil es determinar d?
- se necesitaría factorizar n (sin saber p o q)
- resulta que factorizar un número grande es difícil.

Generación de claves RSA

- encontrar dos primos grandes, p y q
- típicamente: por medio de ensayo y error

Claves de sesión

- La exponenciación es computacionalmente intensiva
- DES es típicamente 100 más rápido que RSA

Clave de sesión, K_S

- Benito y Alicia se intercambian una clave simétrica, K_S, mediante RSA.
- A partir del momento en que ambos comparten K_S, usan criptografía de clave simétrica estándar

Capítulo 8: hoja de ruta

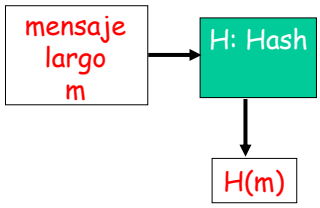
8.1 ¿Qué la seguridad de red?
8.2 Principios de criptografía
8.3 Integridad de mensajes
8.4 Conexiones TCP seguras: SSL
8.5 Seguridad en la capa de red: IPsec
8.6 Seguridad en redes inalámbricas

Integridad de mensajes

- permite a las dos partes de una comunicación verificar la integridad de sus mensajes.
  - es decir, que el contenido del mensaje no ha sufrido alteración
  - el mensaje no se ha repetido (tal vez maliciosamente)
  - se mantiene la sucesión de mensajes
- hablemos de la *función resumen (message digest)*

Funciones "resumen"

- una función resumen H(·) toma como entrada un mensaje de longitud arbitraria y emite un "resumen" de longitud fija: la "firma del mensaje"
- obviamente, H(·) es una función muchos-a-1
- H(·) en inglés se llama *hash* o *message digest*



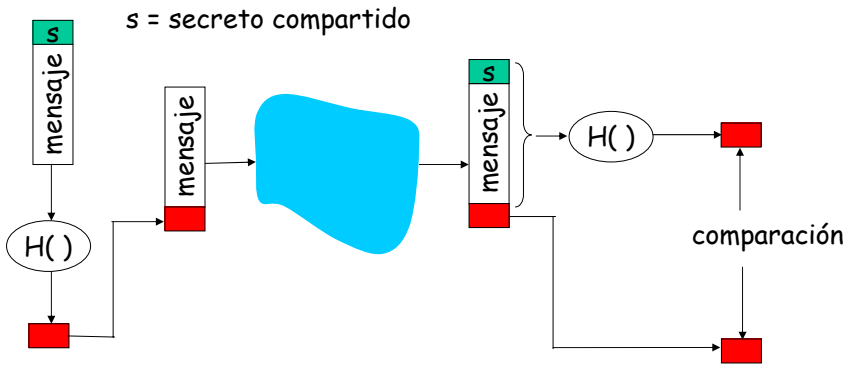
propiedades deseables :

- *fácil de calcular*
- *unidireccional*: no se puede obtener m a partir de H(m)
- *resistente a colisiones*: es muy difícil conseguir m y m' tales que H(m) = H(m')
- que parezca un valor *aleatorio*

Algoritmos para funciones resumen

- MD5: función resumen ya obsoleta, por ser vulnerable a los ataques por colisión (definida en RFC 1321, desaconsejada en RFC 6151)
  - computa un resumen de 128 bits en 4 rondas.
- SHA-x son los actualmente recomendados.
  - estándar en EEUU [NIST, FIPS PUB 180-3]
  - resúmenes de distintos tamaños: 160, 224, 256, 384, 512 bits.

Autenticación (e Integridad): MAC



- ***Autentica al emisor***
- ***Verifica la integridad del mensaje***
- ¡OJO! No hay cifrado (no hay "confidencialidad")
- Notación: $MD_m = H(s||m)$; enviar $m||MD_m$
  - || es el operador "concatenación".

HMAC

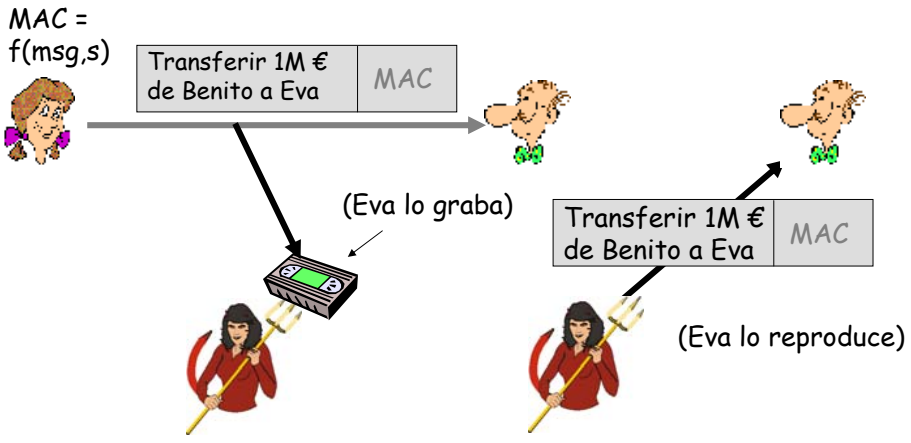
- forma típica de MAC (= "hashed MAC")
- resuelve problemas de seguridad que tiene el esquema anterior cuando se usan *funciones resumen iterativas*
- forma de evitar problemas: usar

$$H(s \,||\, H(s \,||\, m))$$

Autenticación OSPF

- dentro de un AS, los routers se mandan mensajes entre sí.
- OSPF ofrece varias posibles métodos de autenticación:
  - sin autenticación
  - contraseña compartida: insertada en claro en el campo "authentication" (de 64 bits) en el paquete OSPF
  - resumen criptográfico
- resumen criptográfico con MD5
  - el campo "authentication" (de 64 bits) incluye un número de secuencia de 32 bits
  - MD5 "resume" una concatenación del paquete OSPF y la contraseña compartida
  - el resumen MD5 se añade al final del paquete OSPF y se encapsula en un datagrama IP

Ataque por repetición



¡Así, Eva consigue 2M €!

Para defenderse: valores aleatorios únicos (*nonce*)

Firma digital

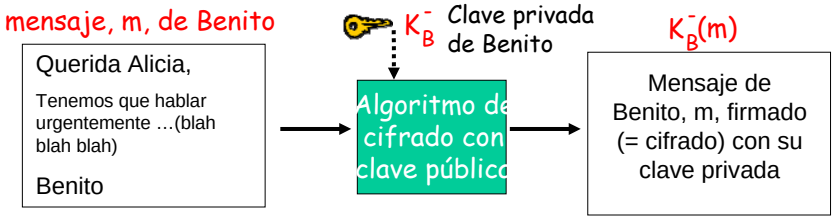
Técnica criptográfica análoga a la firma manuscrita

- El emisor (Benito) firma digitalmente un documento, del que afirma que es su creador y/o dueño.
- Parecido a MAC, pero usando criptografía de clave pública
- *verificable, infalsificable*: el receptor (Alicia) puede probar ante cualquiera (el juez, p. ej.) que
  - el documento está íntegro, (integridad)
  - Benito, y solo Benito, ha podido firmar el documento, (no-repudio)

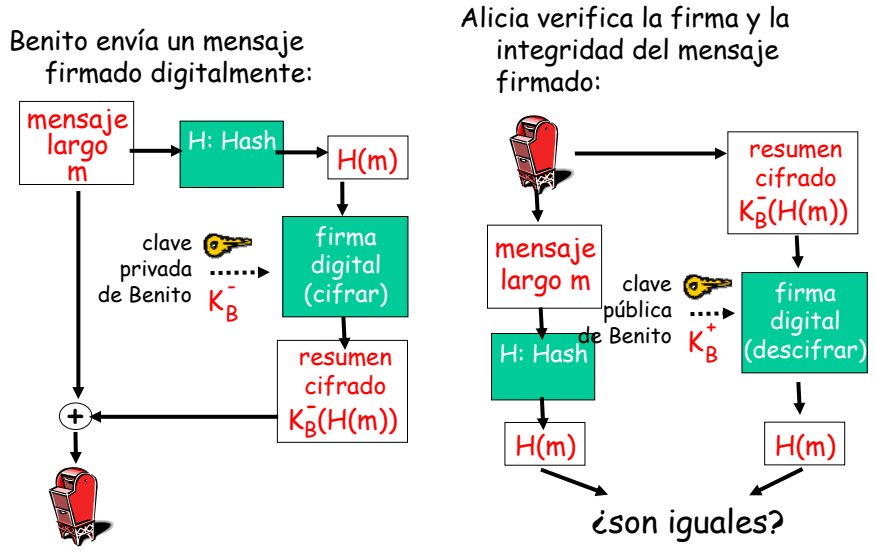
Firma digital

firma digital simple para un mensaje m:

- Benito firma m cifrándolo con su K_B^-, creando un mensaje firmado, $K_B^-(m)$



Firma digital = firma del "resumen"



Firma digital

- Alicia recibe un mensaje m y la firma digital $K_B^-(m)$
- Alicia verifica que Benito firmó m aplicando su clave pública K_B^+ a $K_B^-(m)$ y comprobando que $K_B^+(K_B^-(m)) = m$.
- si $K_B^+(K_B^-(m)) = m$, el que haya firmado m ha usado necesariamente la clave privada de Benito.

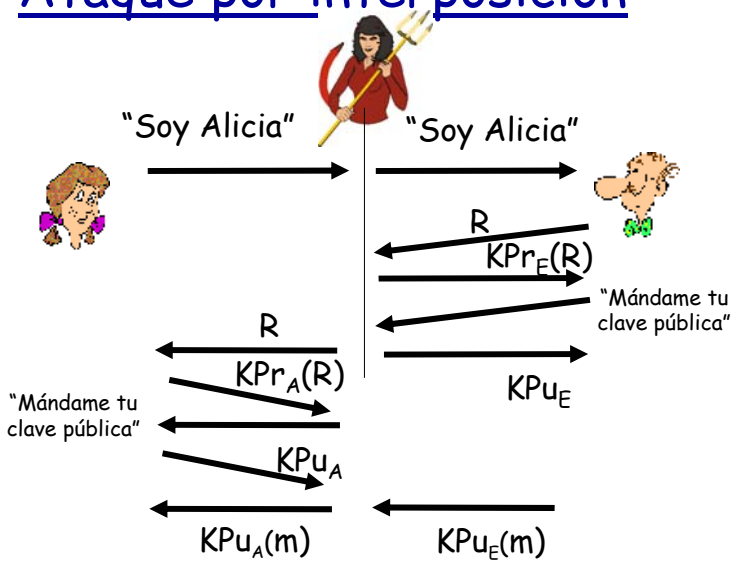
  Por tanto, Alicia está segura de que:
  - ✓ Benito firmó m.
  - ✓ ningún otro pudo hacerlo.
  - ✓ Benito firmó m y no m'.

  No-repudio:
  - ✓ Alicia puede llevar m, y la firma $K_B^-(m)$ ante un tribunal y probar que Benito firmó m.

Certificación de claves públicas

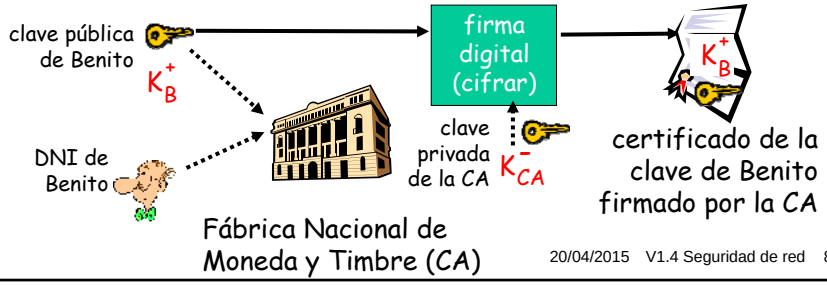
- Motivación: Eva le gasta a Benito la bromita de las pizzas
  - Eva manda un correo con el pedido a la pizzería: *Mándeme cuatro pizzas mozzarella. Gracias, Benito.*
  - Eva firma el pedido con su clave privada (de ella) y lo manda a la pizzería.
  - Eva envía a la pizzería su clave pública (de ella) diciendo que es la de Benito.
  - La pizzería verifica (correctamente) que la firma es de "Benito" y le manda las pizzas.
  - Benito tiene que pagar la cuenta.

Ataque por interposición



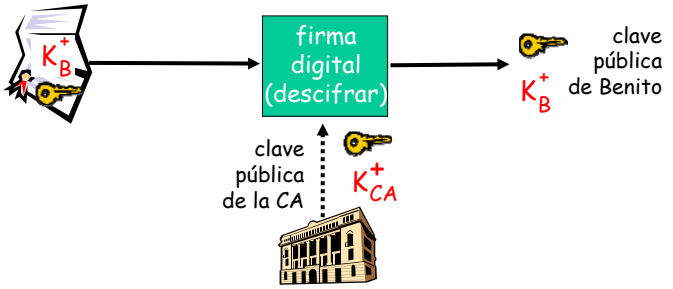
Autoridades de certificación

- Autoridad de certificación (CA): mantiene el registro de claves públicas asignadas a una entidad particular E (una persona, un router).
- E registra su clave pública en la CA.
  - E muestra a la CA una prueba de identidad (p. ej. DNI).
  - CA crea un registro (un "certificado") que relaciona E con su clave pública.
  - el certificado con la clave pública de E va firmado por la CA



Autoridad de certificación

- si Alicia quiere saber la clave pública de Benito:
  - se consigue (de donde sea) el certificado de Benito.
  - aplica la clave pública de la CA al certificado y así recupera la clave pública de Benito.



Certificados: resumen

- estándar fundamental X.509 (RFC 2459)
- el certificado contiene:
  - nombre del emisor
  - nombre de la entidad, dirección, nombre de dominio, etc.
  - la clave pública de la entidad
  - firma digital (firmada con la clave privada del emisor, es decir, de la CA)
- *Public-Key Infrastructure* (PKI)
  - certificados, autoridades de certificación
  - cadena de certificación
  - un poco de "peso pesado".

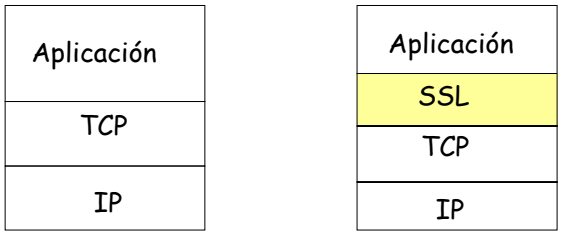
Capítulo 8: hoja de ruta

8.1 ¿Qué la seguridad de red?
8.2 Principios de criptografía
8.3 Integridad de mensajes
8.4 Conexiones TCP seguras: SSL
8.5 Seguridad en la capa de red: IPsec
8.6 Seguridad en redes inalámbricas

SSL: *Secure Sockets Layer*

- protocolo de seguridad de amplio uso
  - soportado por todos los navegadores y servidores web
  - https
  - miles de millones de euros pasan por SSL
- originalmente:
  - Netscape, 1993
- variación -TLS: *transport layer security*, RFC 2246
- ofrece
  - *confidencialidad*
  - *integridad*
  - *autenticación*
- objetivos:
  - comercio electrónico
  - cifrado (especialmente para las tarjetas de crédito)
  - autenticación de servidores web
  - autenticación de clientes
  - minimizar problemas al hacer negocios con nuevos clientes/proveedores
- disponible para todas las aplicaciones TCP
  - interfaz "socket seguro"

Capas en SSL y TCP/IP

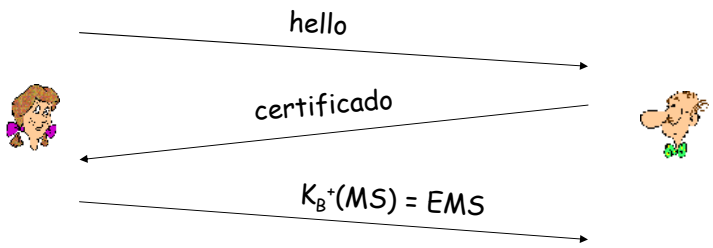


- SSL ofrece una API a las aplicaciones
- Librerías/clases para Java y C/C++ disponibles

SSL de juguete: un canal seguro sencillo

- *negociación:* Alicia y Benito usan sus certificados y claves privadas para autenticarse e intercambiar una clave base
- *derivación de la clave:* A partir de la clave base, Alicia y Benito derivan un conjunto de claves
- *transferencia de datos:* los datos se trocean y transmiten en registros
- *cierre de la conexión:* mensajes especiales para cierre seguro de la conexión

Ejemplo de negociación



- MS = clave base
- EMS = clave base cifrada

Ejemplo de derivación de clave

- Se considera mala práctica usar la misma clave para operaciones criptográficas distintas:
  - usar una para MAC y otra distinta para cifrar
- cuatro claves:
  - K_c = clave de cifrado para datos desde cliente a servidor
  - M_c = clave MAC para datos desde cliente a servidor
  - K_s = clave de cifrado para datos de servidor a cliente
  - M_s = clave MAC para datos de servidor a cliente
- claves derivadas mediante la *función de derivación de claves* (KDF)
  - recoge la clave base y (posiblemente) valores aleatorios adicionales y crea las claves

Ejemplo de registros de datos

- Podríamos cifrar los datos en flujo, según los pasamos a TCP, pero…
  - ¿dónde ponemos el MAC? Si lo ponemos al final del todo, no se puede comprobar la integridad mientras no lleguen todos los datos.
  - P. ej., en mensajería instantánea, no podríamos ir mostrando el mensaje según va llegando, habría que esperar al final.
- en vez de eso, dividimos el mensaje en registros,
  - cada registro lleva su MAC
  - y el receptor puede procesar cada registro independientemente
- ojo: en el registro hay que distinguir los datos del MAC.
  - usaremos registros de longitud variable

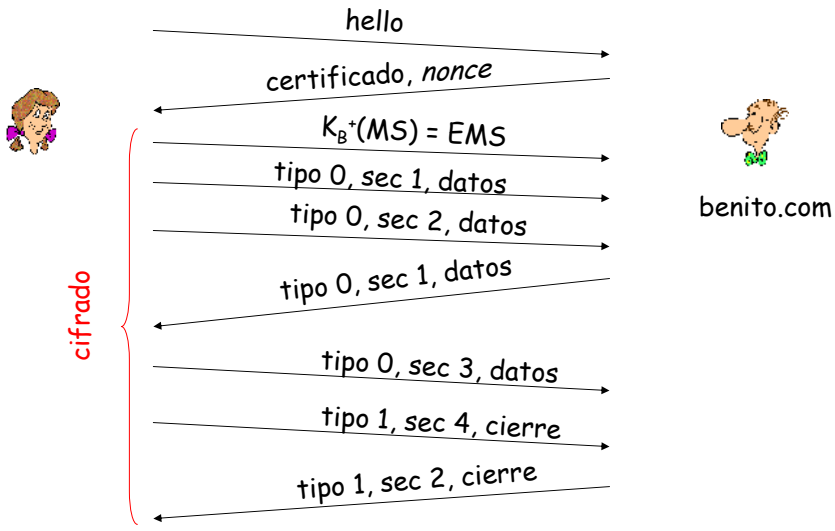
Ejemplo: números de secuencia

- para evitar el ataque por repetición, o reordenación…
- colocamos números de secuencia en el MAC:
  - MAC = MAC(M_x, secuencia || datos)
  - Nota: ¡no existe un campo para el número de secuencia!
- el atacante podría repetir *todos* los registros…
  - usar valores aleatorios únicos (*nonces*) en la generación de claves

Ejemplo: información de control

- ataque por truncado:
  - el atacante falsifica un segmento de cierre de conexión TCP…
  - así que un lado (o los dos) se creen que hay menos datos de los que realmente hay..
- una solución: incluir un *tipo de registro* donde el cierre reciba un valor determinado, p. ej.,
  - tipo 0 para datos; tipo 1 para cierre
- MAC = MAC(M_x, secuencia || tipo || datos)

Resumen del ejemplo SSL



El ejemplo no está completo…

- ¿tamaño de los campos?
- ¿protocolos de cifrado?
- ¿cómo se negocia?
  - se debiera permitir que cliente y servidor soporten diversos algoritmos de cifrado…
  - …y que, antes de comenzar la transferencia de datos, negocien cuál quieren usar

Elencos de cifrados SSL

- cada elenco
  - algoritmo de clave pública
  - algoritmo de clave simétrica
  - algoritmo MAC
- SSL soporta varios elencos de cifrados
- negociación: cliente y servidor acuerdan qué elenco se usa
  - cliente hace su oferta
  - el servidor elige uno

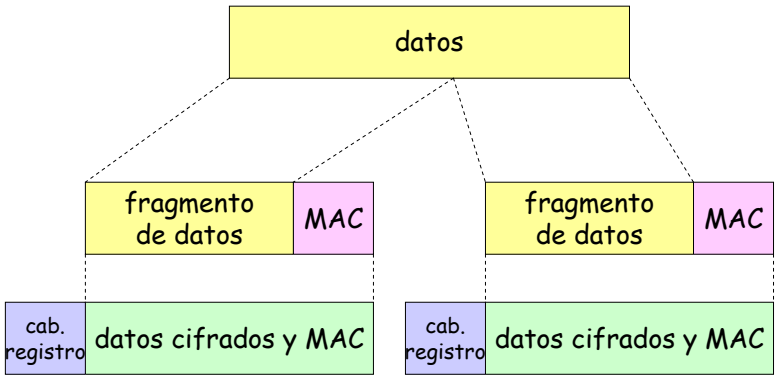
Típicos cifrados simétricos en SSL:

- DES – Data Encryption Standard: en bloques
- 3DES – Triple DES: en bloques
- RC2 – Rivest Cipher 2: en bloques
- RC4 – Rivest Cipher 4: en flujo

Clave pública SSL:

- RSA

Registros del protocolo SSL

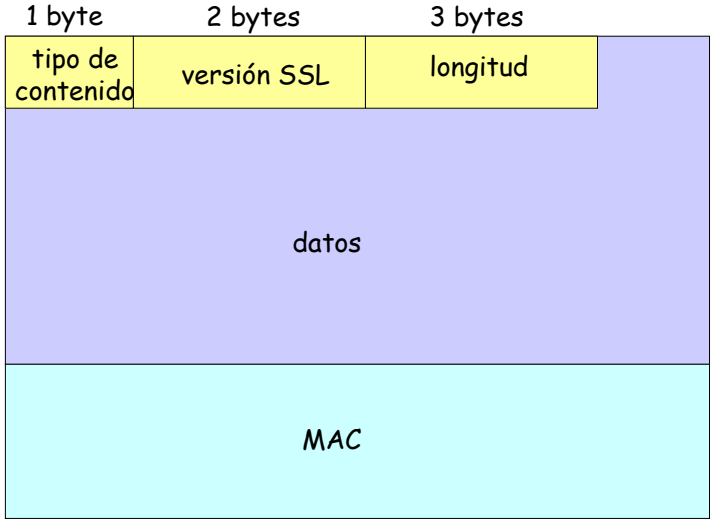


cabecera de registro: tipo de contenido; versión; longitud

MAC: incluye número de secuencia, clave MAC M_x

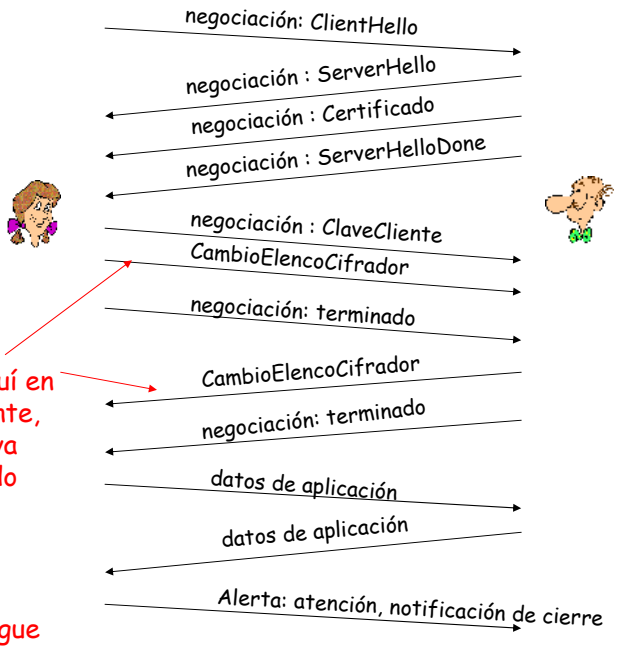
fragmento: cada fragmento SSL 2^{14} bytes (~16 Kbytes)

Formato del registro SSL



datos y MAC cifrados (con el algoritmo simétrico negociado)

Conexión Real



Derivación de claves

- *nonce* de cliente, *nonce* del server, y pre-clave base se meten al generador pseudo-aleatorio…
  - produce clave base
- clave base y nuevos *nonces* se meten a otro generador aleatorio: genera "bloque de claves"
- bloque de claves troceado y distribuido al azar:
  - clave MAC cliente
  - clave MAC servidor
  - clave de cifrado del cliente
  - clave de cifrado del servidor
  - vector de initialización del cliente (VI)
  - vector de initialización del servidor (VI)

Capítulo 8: hoja de ruta

8.1 ¿Qué la seguridad de red?
8.2 Principios de criptografía
8.3 Integridad de mensajes
8.4 Conexiones TCP seguras: SSL
8.5 Seguridad en la capa de red: IPsec
8.6 Seguridad en redes inalámbricas

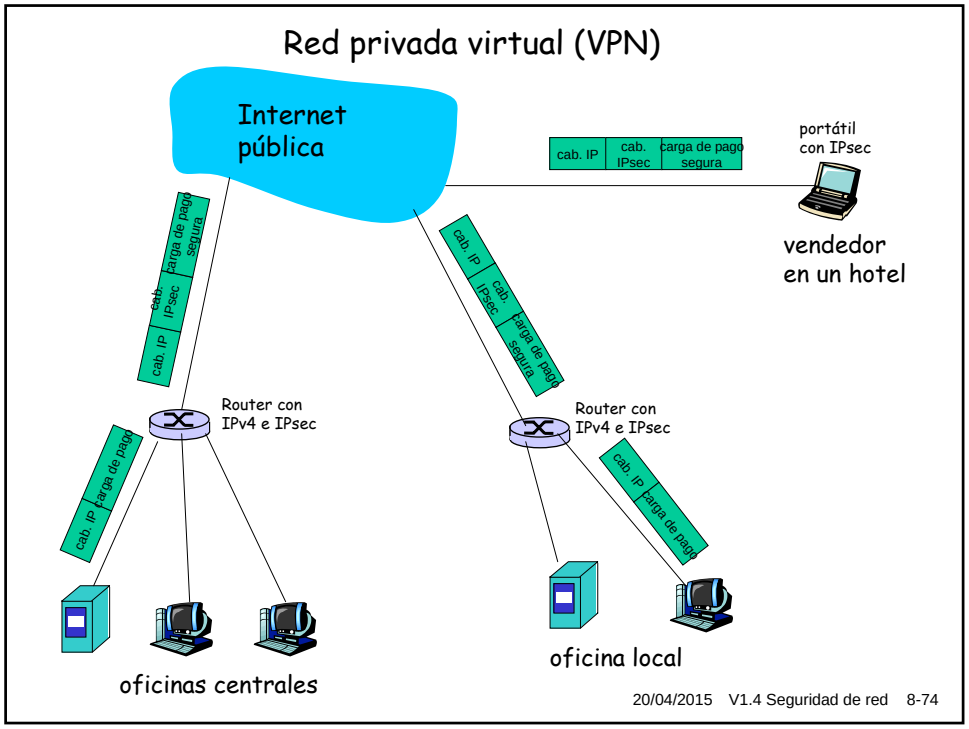
¿Qué es confidencialidad en la capa de red?

entre dos entidades de red:

- el emisor cifra la parte de datos del datagrama (*carga de pago*), que puede ser:
  - un segmento TCP o UDP, un mensaje ICMP, un mensaje OSPF…
- así los datos enviados de una entidad a otra estarían ocultos:
  - páginas web, correo electrónico, transferencia de ficheros P2P, paquetes TCP SYN…

Redes privadas virtuales (*VPN*)

- las instituciones suelen preferir redes privadas, por cuestión de seguridad.
  - es costoso: routers y enlaces separados, infraestructura para DNS, etc.
- VPN: en vez de lo anterior, el tráfico entre instituciones se envía por la Internet pública.
  - se cifra antes de entrar en Internet…
  - …y así está separado (desde un punto de vista lógico, no físico) del resto de tráfico

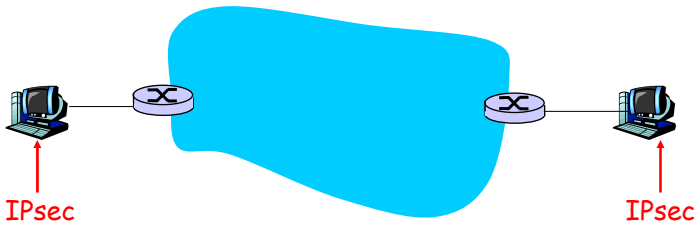


Servicios IPsec

- integridad de datos
- autenticación del origen
- prevención del "ataque por repetición"
- confidencialidad

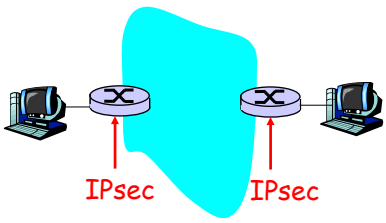
- dos protocolos ofrecen diversos modelos de servicio:
  - AH
  - ESP

IPsec modo transporte

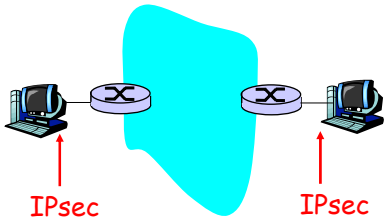


- datagrama IPsec emitido y recibido por el sistema terminal
- protege los protocolos de nivel superior

IPsec – modo "tunelado"



- routers de frontera con soporte IPsec



- sistemas terminales con soporte IPsec

Dos protocolos

- *Authentication Header* (AH)
  - ofrece autenticación del emisor e integridad de datos, pero *no confidencialidad*
- *Encapsulation Security Protocol* (ESP)
  - ofrece autenticación del emisor e integridad de datos *y confidencialidad*
  - más usado que AH

Cuatro posibles combinaciones…

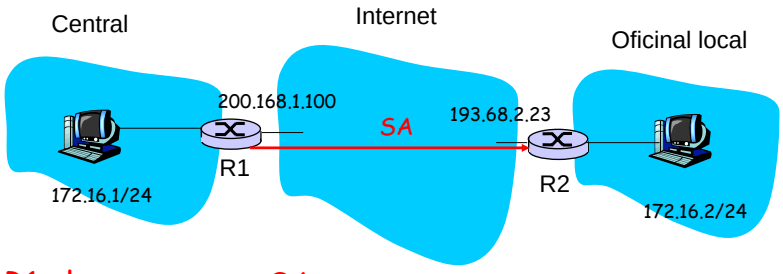
Modo terminal con AH	Modo terminal con ESP
Modo túnel con AH	Modo túnel con ESP

más común e importante

Security associations (SAs)

- antes de enviar datos se establece una "asociación de seguridad (SA)" desde la entidad emisora a la receptora
  - Las SAs son *unidireccionales*.
- Emisor y receptor mantienen *información de estado* acerca de su SA
  - Ojo: los nodos terminales TCP también mantienen *información de estado*
  - mientras que IP es sin conexión, IPsec es *orientado a conexión*
- ¿cuántas SAs en una VPN con oficinas centrales, locales y *n* viajantes?

Ejemplo de SA de R1 a R2



R1 almacena para SA

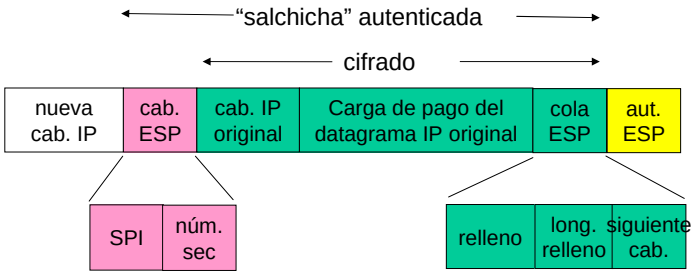
- Identificador SA de 32 bits: *Security Parameter Index (SPI)*
- Interfaz SA origen (200.168.1.100)
- Interfaz SA destino (193.68.2.23)
- tipo de cifrado a usar (ej., 3DES con CBC)
- clave de cifrado
- tipo de comprobación de integridad (ej., HMAC con MD5)
- clave de autenticación

Base de datos de SAs (SAD)

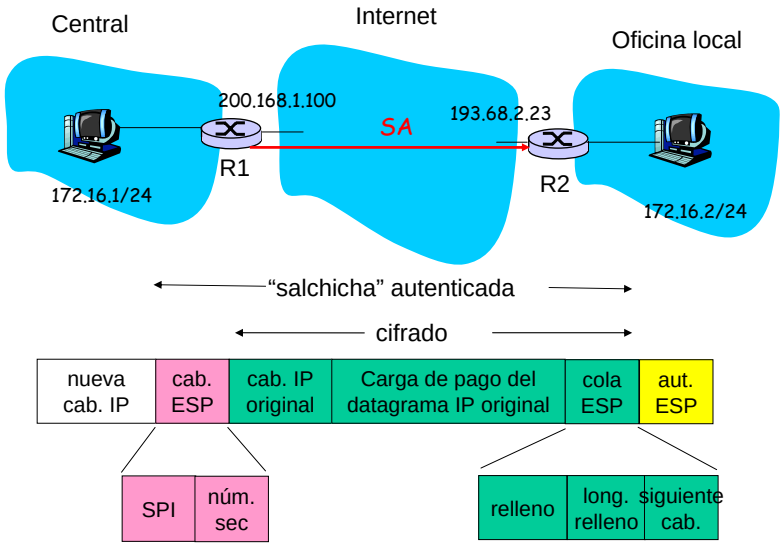
- los puntos terminales mantienen el estado de las SAs en la SAD.
- con *n* viajantes, 2 + 2*n* SAs en SAD de R1
- al enviar un datagrama IPsec, R1 accede a SAD para determinar cómo procesar ese datagrama.
- cuando el datagrama IPsec llega a R2, R2 examina SPI en el datagrama IPsec, indexa SAD con SPI, y procesa el datagrama como corresponda.

datagrama IPsec

nos centramos ahora en modo túnel con ESP



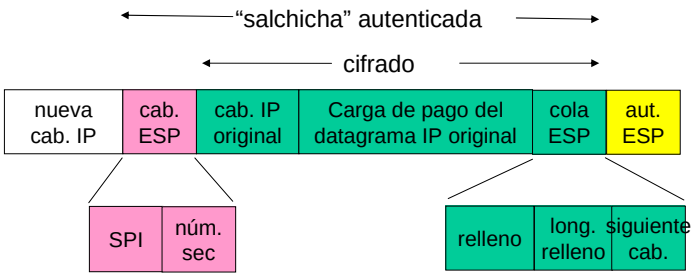
¿Qué ocurre?



R1 convierte el datagrama IP en un datagrama IPsec

- añade al final del datagrama original (que incluye los campos de la cabecera original) un campo "cola ESP".
- cifra el resultado usando el algoritmo y la clave especificados por la SA.
- añade al comienzo del cifrado una "cabecera ESP", creando así la "salchicha".
- crea un MAC autenticado sobre toda la salchicha, usando el algoritmo y la clave especificados por la SA;
- añade el MAC al final de la salchicha, completando así la "carga de pago";
- crea una nueva cabecera IP con los clásicos campos IPv4, y la añade por delante de la carga de pago.

Dentro de la "salchicha":



- cola ESP: relleno para cifrador en bloque
- cabecera ESP:
  - está el SPI para que el receptor sepa qué ha de hacer
  - número de secuencia, para evitar ataques por repetición
- el MAC en el campo "aut. ESP" está creado con la clave secreta compartida

número de secuencia IPsec

- para cada nueva SA, el emisor inicializa la secuencia a "0"
- por cada datagrama enviado sobre una SA:
  - el emisor incrementa el número de secuencia…
  - …y lo coloca en el campo "número de secuencia".
- objetivo:
  - evitar que un atacante repita un paquete espiado
  - los paquetes duplicados y autenticados pueden interrumpir el servicio
- método:
  - se comprueban en destino los duplicados
  - no se mantiene registro de TODOS los paquetes recibidos: solo de una ventana

Base de datos de política de seguridad (SPD)

- política: para cada datagrama, el emisor debe saber si ha de usar IPsec
- si es que sí, con qué SA
  - puede usar: direcciones IP fuente y destino; número de protocolo
- la info en SPD indica "qué" hacer con los datagramas entrantes
- la info en SAD indica también "cómo" hacerlo

Resumen: servicios IPsec

- Si Eva está colocada entre R1 y R2 y no sabe las claves…
  - ¿Podrá ver el contenido original del datagrama?
  - ¿Podrá ver las direcciones IP fuente o destino, el protocolo de transporte, el puerto?
  - ¿Podrá alterar bits sin que se note?
  - ¿Podrá suplantar a R1 usando la IP de R1?
  - ¿Podrá repetir un datagrama?

Internet Key Exchange

- En los ejemplos anteriores, se establecía manualmente la SA de cada nodo terminal:

  Ejemplo de SA

  SPI: 12345
  IP fuente: 200.168.1.100
  IP destino: 193.68.2.23
  Protocolo: ESP
  Algoritmo de cifrado: 3DES-cbc
  Algoritmo HMAC: MD5
  Clave de cifrado: 0x7aeaca…
  Clave HMAC: 0xc0291f…
- Pero hacer esto manualmente para una VPN con cientos de nodos es imposible …
- solución: *IPsec IKE (Internet Key Exchange)*

IKE: PSK y PKI

- cada entidad Ipsec tiene un certificado que incluye su clave pública
- el protocolo exige que
  - cada entidad intercambie certificados,
  - negocie los algoritmos de autenticación y cifrado
  - intercambien el material criptográfico necesario para crear las claves de sesión
- es parecido a la negociación de SSL

Fases IKE

- IKE tiene dos fases:
  - fase 1: mediante un intercambio de Diffie-Hellman, se crea un canal seguro con una clave compartida.
    - No se intercambia visiblemente ningún certificado
  - fase 2: utilizando el canal creado, se intercambian certificados y se revelan las identidades y se establecen las claves de sesión para las dos SA

Resumen de IPsec

- protocolo IKE para intercambio de algoritmos, claves secretas, números SPI
- protocolos AH o ESP (o ambos)
  - AH ofrece integridad y autenticación del origen
  - ESP (con AH) añade cifrado a lo anterior
- los pares IPsec pueden ser: dos nodos terminales, dos routers/cortafuegos, un router/cortafuegos y un nodo terminal, etc

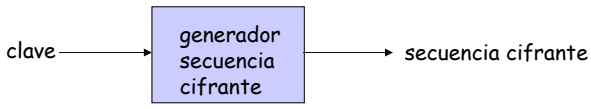
Capítulo 8: hoja de ruta

8.1 ¿Qué la seguridad de red?
8.2 Principios de criptografía
8.3 Integridad de mensajes
8.4 Conexiones TCP seguras: SSL
8.5 Seguridad en la capa de red: IPsec
8.6 Seguridad en redes inalámbricas

Objetivos del diseño WEP

- criptografía de clave simétrica
  - confidencialidad
  - autorización del nodo terminal
  - integridad de datos
- auto-sincronizado: cada paquete se cifra por su lado
  - dado un paquete y la clave, puede descifrar;
  - no depende del paquete anterior (que podría estar perdido) al revés que CBC en los cifradores de bloque
- eficiente
  - se puede implementar en hardware o software

Repaso: Cifrado simétrico en flujo



- combinar cada byte de la secuencia cifrante con cada byte del mensaje en claro
  - m(i) = i-ésima unidad del mensaje
  - ks(i) = i-ésima unidad de la secuencia cifrante
  - c(i) = i-ésima unidad del criptograma
  - $c(i) = ks(i) \oplus m(i)$ ($\oplus$ = o exclusivo)
  - $m(i) = ks(i) \oplus c(i)$
- WEP usa RC4

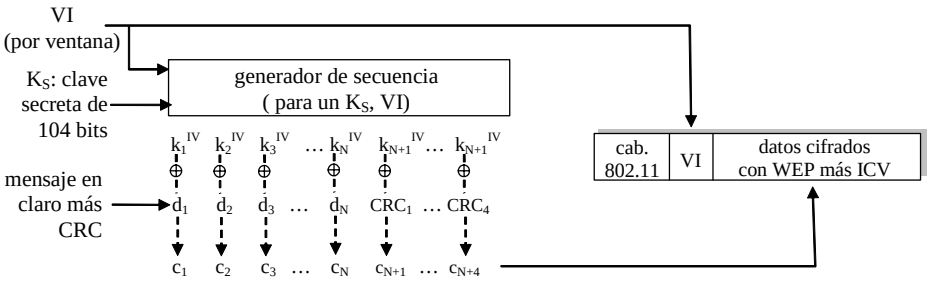
Cifrado en flujo e indepencia de cada paquete

- recordemos: cada paquete cifrado por separado
- si para la ventana n+1, usamos la secuencia cifrante desde donde la dejamos en la ventana n, no tenemos cifrados independientes
- idea en WEP: inicializar la secuencia cifrante con la clave + nuevo VI (vector de inicialización) para cada paquete:

cifrado WEP (1)

- el emisor calcula un valor de comprobación de integridad sobre los datos (ICV)
  - hash/CRC de cuatro bytes, para la integridad de los datos
- cada parte tiene una clave compartida de 104 bits
- el emisor crea un VI de 24 bits y se lo añade a la clave, que pasa a tener 128 bits
- añade también un ID de clave (en un campo de 8 bits)
- los 128 bits se pasan a un generador pseudo-aleatorio, para obtener la secuencia cifrante
- los datos de la ventana + ICV se cifran con RC4:
  - los bytes de la secuencia cifrante se suman con los bytes de los datos e ICV
  - VI e IDclave se añaden al criptograma para crear la carga de pago
  - la carga de pago se inserta en un paquete 802.11 frame

cifrado WEP (2)



Un nuevo VI para cada ventana

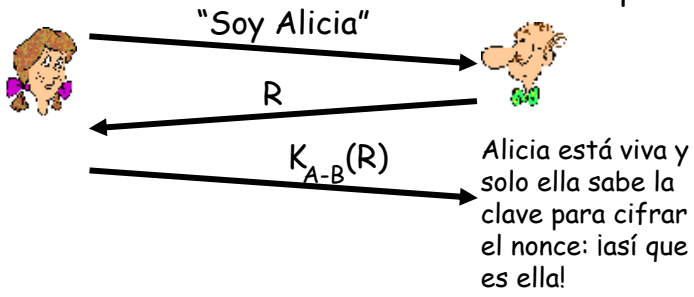
descifrado WEP

- el receptor extrae VI
- introduce VI, la clave compartida en generador pseudo-aleatorio, obtiene secuencia cifrante
- suma la secuencia con los datos cifrados y obtiene los datos en claro e ICV
- verifica la integridad de datos mediante ICV
  - obsérvese que la integridad del mensaje se garantiza aquí de una forma distinta al MAC o a las firmas digitales.

Autenticación extremo-a-extremo con *nonce*

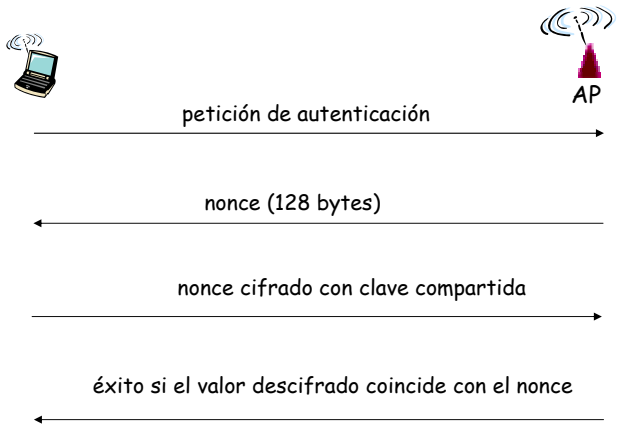
Nonce: número (R) usado *solo una vez en la vida*

Cómo: para comprobar que Alicia "vive", Benito le envía un *nonce*, R. Alicia ha de devolverlo, pero cifrado con la clave secreta compartida



Autenticación WEP

No todos los APs lo hacen, aun usando WEP. AP indica en la trama-baliza si es necesaria la autenticación. Se realiza antes de la asociación.



Vulnerar el cifrado 802.11 WEP

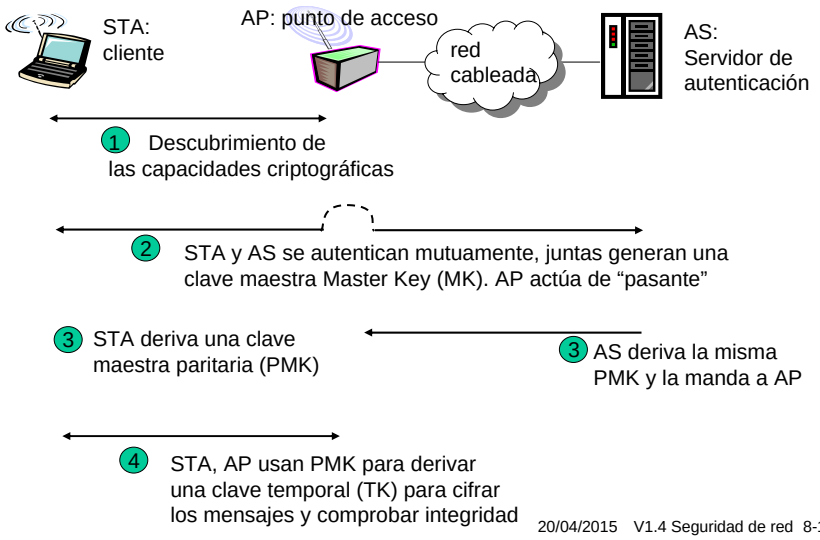
hay un *agujero* de seguridad:

- si usamos un VI de 24 bits, con un VI por trama, al final los VIs se van a repetir
- VI transmitido en claro -> VI es fácil detectar el reuso
- ataque:
  - Eva hace que Alicia cifre textos claros conocidos, d_1 d_2 d_3 d_4 …
  - Eva ve: $c_i = d_i$ XOR k_i^{VI}
  - Eva conoce c_i d_i, puede calcular k_i^{VI}
  - Eva sabe la secuencia de claves k_1^{VI} k_2^{VI} k_3^{VI} …
  - Cuando VI se reuse, ¡Eva puede descifrar!

802.11i: seguridad mejorada

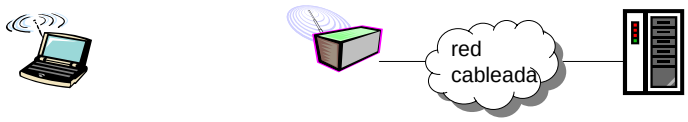
- posibles numerosas (y más sólidas) formas de cifrar
- ofrece distribución de claves
- usa un servidor de autenticación separado del punto de acceso (AP)

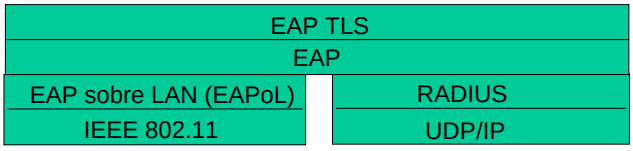
802.11i: cuatro fases de operación



EAP: *extensible authentication protocol*

- EAP: protocolo desde el cliente (móvil) al servidor de autenticación
- EAP se envía sobre "enlaces separados":
  - móvil-a-AP (EAP sobre LAN)
  - AP al servidor de autenticación (RADIUS sobre UDP)





Capítulo 8: temas para leer

8.7 Correo electrónico seguro

8.8 Seguridad en redes: cortafuegos
bastiones y detección de intrusiones