Capítulo 8
Seguridad en Redes
8.6 Conexiones TCP Seguras: SSL

Basado en:
Computer Networking: A Top Down Approach.
Jim Kurose, Keith Ross.

Capítulo 8 contenidos

8.1 ¿Qué es la seguridad en la red?

8.2 Principios de criptografía

8.3 Integridad de mensajes

8.4 Autenticación extremo a extremo

8.5 Dando seguridad a e-mail

8.6 Conexiones TCP seguras: SSL

8.7 Seguridad en capa de Red: IPsec

8.8 Seguridad en redes locales inalámbricas

8.9 Cortafuegos y Sistemas de detección de intrusión (IDS)

SSL: Secure Sockets Layer
Sockets seguros (Capa 4)

- Protocolo de seguridad ampliamente difundido
  - Usado en la mayoría de los navegadores y servidores web
  - https
  - Usado en transferencias de comercio electrónico. Billones $/año van sobre SSL
- Originalmente implementado por Netscape en 1994
- Existen variantes:
  - TLS: transport layer security, RFC 5246 v1.2 del 2008
- Provee
  - Confidencialidad
  - Integridad
  - Autenticación
- Objetivos originales:
  - Permitir el comercio electrónico en la Web
  - Encriptación (especialmente de números de tarjetas de créditos)
  - Autenticación de servidores Web
  - Opcionalmente autenticación de clientes
  - Minimizar riesgos al hacer negocios con nuevos clientes
- Disponible para toda conexión TCP
  - Interfaz de socket segura

SSL y TCP/IP

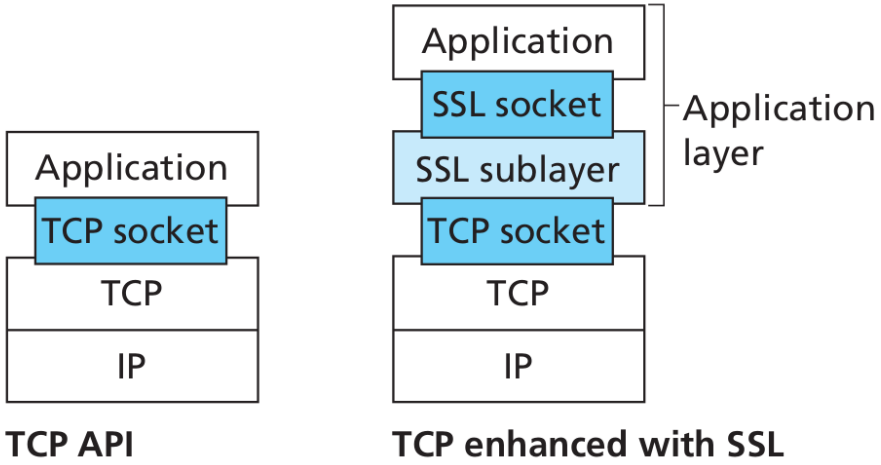
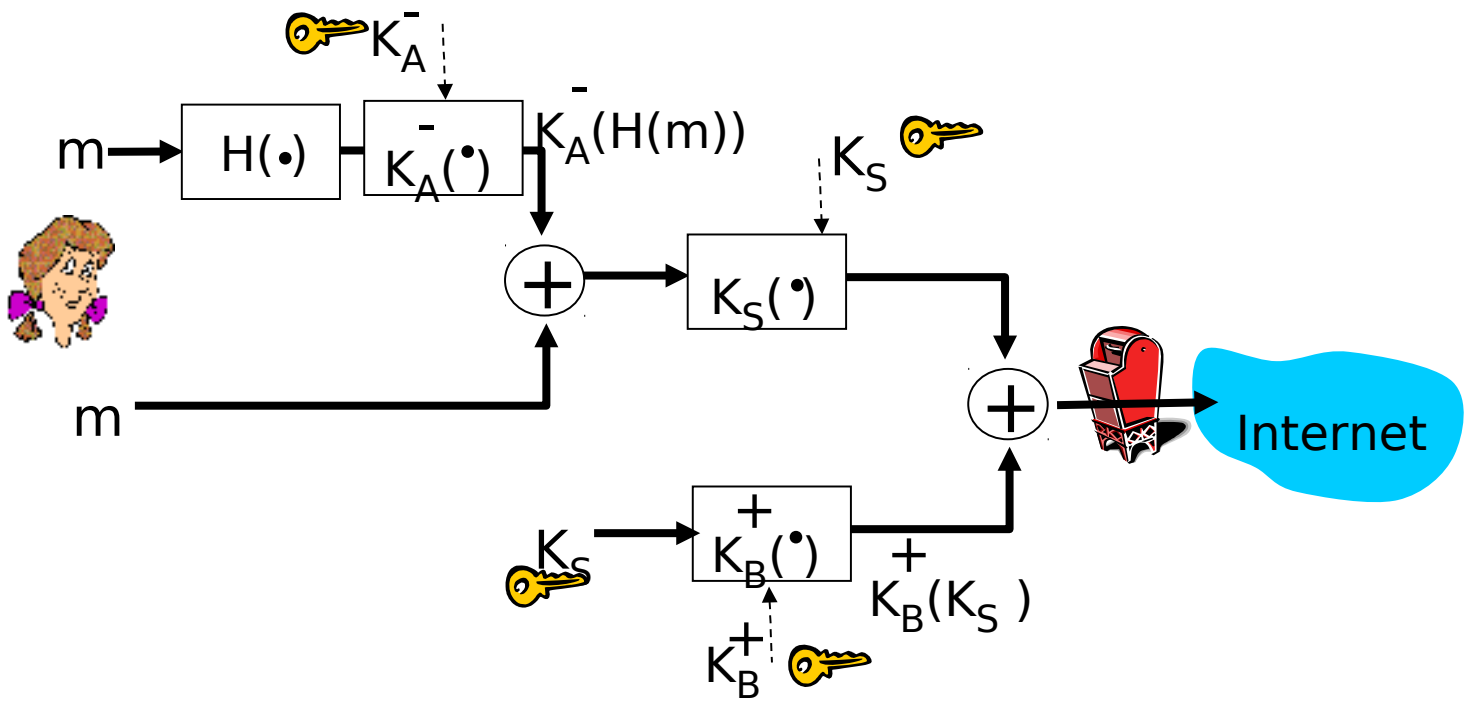


Figure 8.24 ♦ Although SSL technically resides in the application layer, from the developer's perspective it is a transport-layer protocol

- SSL provee una interfaz de programación de aplicaciones (API) para desarrollar aplicaciones
- Existen Bibliotecas SSL en C y clases SSL para Java y C++

Se podría hacer algo similar a PGP (Pretty Good Privacy):



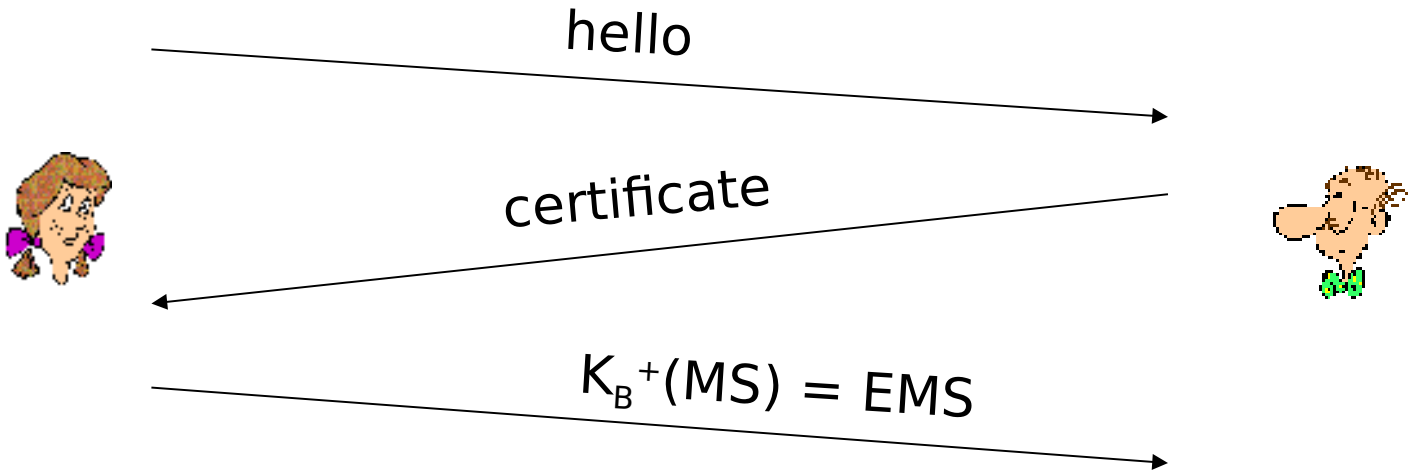
- Pero queremos enviar flujos de byte y datos interactivos
- Queremos un conjunto de claves para toda la conexión.
- Queremos intercambio de certificados como aparte del protocolo en fase de establecimiento de conexión (handshake)

Idea simple de canal seguro: 4 pasos

- Handshake: Alice y Bob usan sus certificados y claves privadas para autenticarse mutuamente e intercambiar un secreto compartido.
- Derivación de Claves: ambos usan el secreto compartido para derivar un conjunto de claves
- Transferencia de datos: Los datos a ser transferidos son divididos en una serie de registros.
- Cierre de conexión: Mensaje especial para cerrar conexión en forma segura.

Idea: Un handshake simple

- Luego de establecer una conexión TCP.



- MS = master secret
- EMS = encrypted master secret

Idea para derivación de clave

- Se podría usar la clave maestra, pero es considerado malo usar la misma clave para más de una operación de encriptación.
  - Se opta por usar claves diferentes para código de autenticación de mensaje (MAC) y encriptación.
- Se usa la clave maestra para generar 4 claves:
  - K_c = clave de sesión para encriptación de datos de cliente a servidor.
  - M_c = clave MAC de datos de cliente a servidor.
  - K_s = clave de sesión para encriptación de datos de servidor a cliente.
  - M_s = clave MAC de datos de servidor a cliente.
- Estas claves son derivadas de un secreto maestro enviado por Alice al inicio.

Registro (bloque) de datos simple

- ¿Por qué enviar bloques y no flujo TCP?
  - ¿Dónde pondríamos el MAC? Al final, no se tendría integridad hasta el final!
  - Por ejemplo, en mensajería instantánea, debemos chequear integridad antes de desplegar el mensaje
- Se divide el flujo en una serie de registros
  - Cada registro lleva un MAC
  - Receptor puede verificar cada registro a su llegada.
- Problema: El receptor debe distinguir datos del código de autenticación (MAC)
  - Deseamos usar un registro de largo variable. Debemos agregar campo largo.

Números de secuencia

- Problema: Atacante podría capturar y regenerar un registro o cambiar su orden.
- Solución: poner número de secuencia en MAC:
  - MAC = MAC(M_x, #secuencia+data)
  - Debemos agregar campo # secuencia
- Problema: Atacante aún podría reproducir todos los registros
- Solución: Usar números únicos (random nonce)

Información de control

- Problema: Ataque de truncado:
  - Atacante falsifica un segmento de cierre de conexión.
  - Uno o ambos lados piensan que hay menos datos que los reales.
- Solución: Usar tipo de registro
  - tipo 0 para datos; tipo 1 para cierre
- MAC = MAC(M_x, #secuencia+type+data)

No es necesario enviarlo

SSL simple: resumen

bob.com

hello

certificate, nonce

$K_B^+(MS) = EMS$

type 0, seq 1, data

type 0, seq 2, data

type 0, seq 1, data

type 0, seq 3, data

type 1, seq 4, close

type 1, seq 2, close

encriptado

SSL simple no está completo

- ¿Qué largo tienen los campos?
- ¿Qué protocolo de encriptación usar?
- No tenemos negociación
  - Deberíamos permitir al cliente y servidor soportar diferentes algoritmos de encriptación.
  - Deberíamos permitir al cliente y servidor elegir juntos algoritmos específicos antes de la transferencia.

Mecanismos de cifrado SSL

- Aspectos a acordar:
  - Algoritmo de clave pública
  - Algoritmo de encriptación simétrica
  - Algoritmo MAC
- SSL soporta varios mecanismos de cifrado.
- Negociación: Cliente y servidor deben acordar algoritmos.
  - Cliente ofrece opciones
  - Servidor selecciona una

Cifrado SSL simétrico comunes

- DES – Data Encryption Standard: bloques
- 3DES – Triple strength: bloques
- RC2 – Rivest Cipher 2: bloques
- RC4 – Rivest Cipher 4: flujo (stream)

Cifrado de clave pública

- RSA

SSL Real: Handshake (1)

Propósito

1. Autenticar al servidor
2. Negociación: acordar algoritmos de cifrado.
3. Establecer claves
4. Autenticación del cliente (opcional)

SSL real: Handshake (2)

1. El cliente envía una lista de algoritmos que soporta, junto con un número de unicidad del cliente (para evitar replicación de mensajes).
2. Servidor elije algoritmo desde lista; envía: su elección + certificado + número de unicidad del servidor
3. Cliente verifica certificado, extrae clave pública del servidor, genera “pre_master_secret”, lo encripta con clave pública de servidor, lo envía al servidor
4. Cliente y servidor calculan independientemente la clave de encriptación y la calve MAC a partir de pre_master_secret y números de unicidad
5. Cliente envía un MAC de todos los mensajes de handshake
6. Servidor envía un a MAC de todos los mensajes de handshake

SSL real: Handshaking (3)

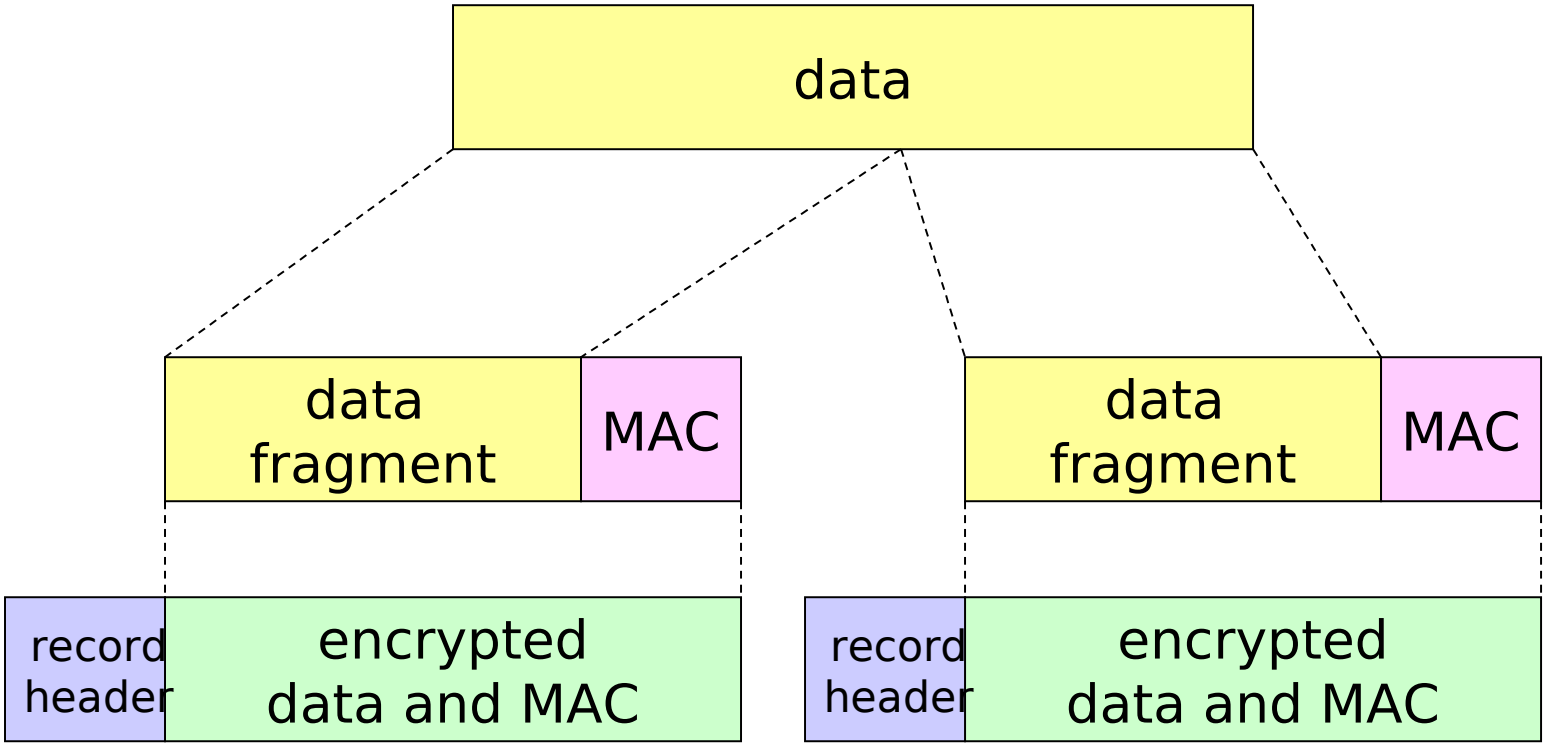
Los últimos 2 pasos protegen el handshake de ser manipulado (ej. Man-in-the-middle)

- Cliente típicamente ofrece un rango de algoritmos de cifrado, algunos robustos y otros débiles.
- “Man-in-the middle” podría borrar los robustos de la lista
- Los últimos 2 pasos lo evitan
  - Los últimos dos mensajes son encriptados.

SSL: Handshaking (4)

- ¿Por qué usar dos números de unicidad aleatorios?
- Supongamos el intruso observa todos los mensajes entre Alicia y Bob.
- Más tarde, intruso establece una conexión TCP con Bob y envía exactamente la misma secuencia.
  - Bob (Amazon) piensa que Alicia hace dos compras separadas de lo mismo.
  - Solución: Bob envía diferentes números aleatorios cada vez en cada conexión. Así las claves de cifrado serán distintas ambas veces.
  - Mensajes del intruso fallarán los chequeos de integridad de Bob.

SSL: Registro del Protocolo

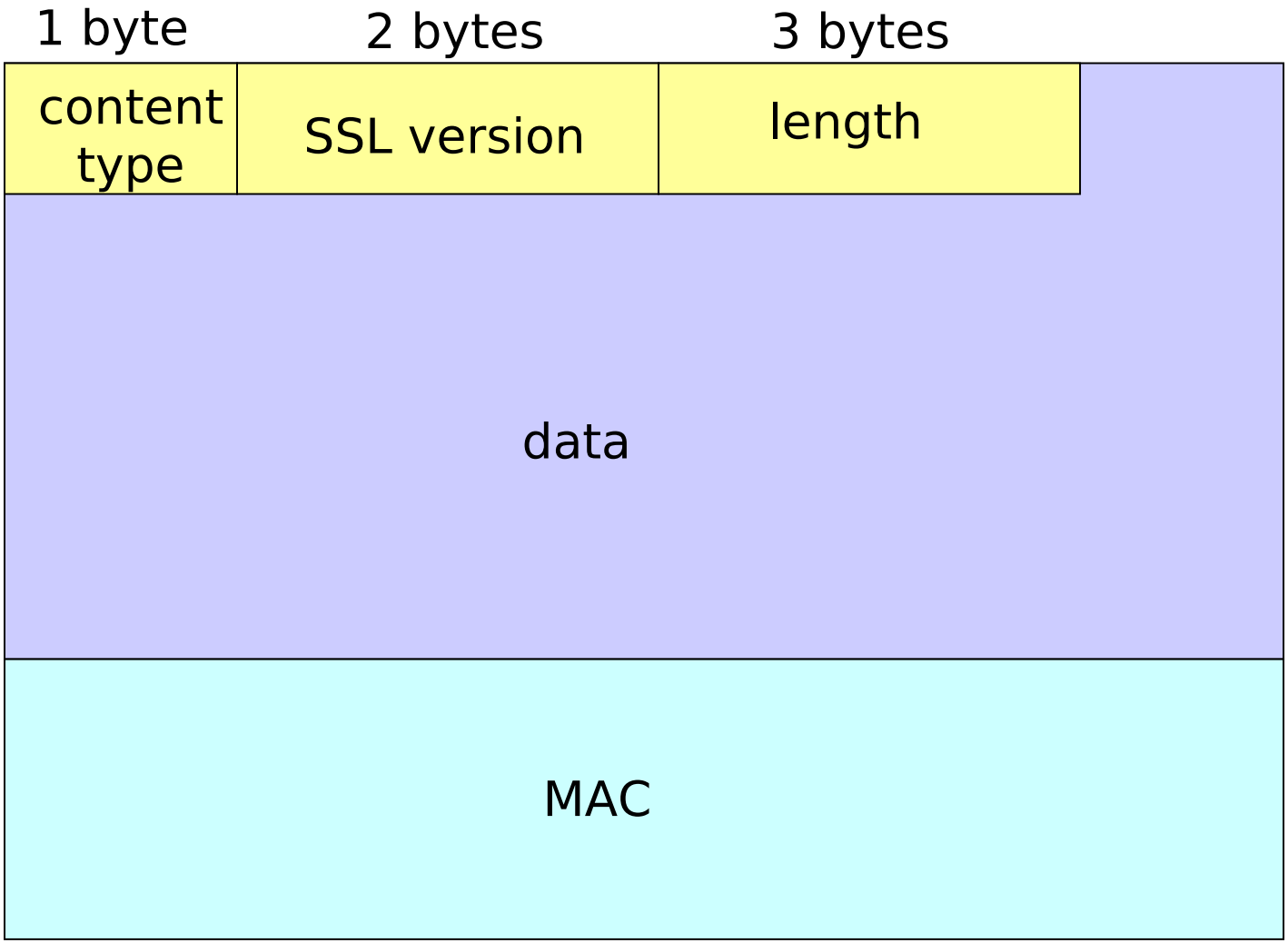


Record header: contiene: tipo, versión, largo

MAC: incluye número de secuencia, clave MAC M_x

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

SSL: Formato del registro



Data y MAC van cifradas (algoritmo simétrico)

Conexión Real

handshake: ClientHello

handshake: ServerHello

handshake: Certificate

handshake: ServerHelloDone

handshake: ClientKeyExchange

ChangeCipherSpec

handshake: Finished

ChangeCipherSpec

handshake: Finished

application_data

application_data

Alert: warning, close_notify

Desde aquí todo va encriptado con Claves de sesión

Sigue intercambio de Fin de TCP

Capítulo 8 contenidos

8.1 ¿Qué es la seguridad en la red?

8.2 Principios de criptografía

8.3 Integridad de mensajes

8.4 Autenticación extremo a extremo

8.5 Dando seguridad a e-mail

8.6 Conexiones TCP seguras: SSL

8.7 Seguridad en capa de Red: IPsec

8.8 Seguridad en redes locales inalámbricas

8.9 Cortafuegos y Sistemas de detección de intrusión (IDS)