Universidad de Oviedo

Departamento de Informática

Campus de Gijón

Protocolos de Seguridad: SSL/TLS

Presentación

Daniel F. García

Introducción a los protocolos de seguridad 1

Definición

Un protocolo de seguridad define la secuencia de funciones necesarias para garantizar la protección de la información usando algoritmos criptográficos

Usado con un protocolo de comunicaciones permite el intercambio seguro de información entre dos entidades

Un protocolo de seguridad incluye al menos alguna de estas funciones:

- Acordar o establecer claves
- Cifrar información
- Autenticar entidades
- Autenticar mensajes

Hay 4 enfoques básicos para proporcionar seguridad a las comunicaciones:

1) Usar un protocolo de seguridad separado
2) Integrar la seguridad en el protocolo de aplicación
3) Integrar la seguridad en el protocolo de transporte/red
4) Usar un protocolo de seguridad paralelo

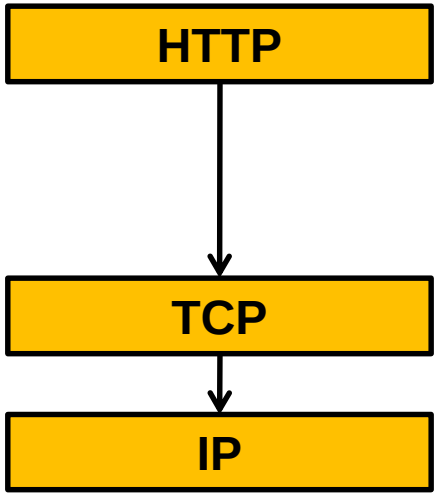
Introducción a los protocolos de seguridad 2

1) Usar un protocolo de seguridad separado

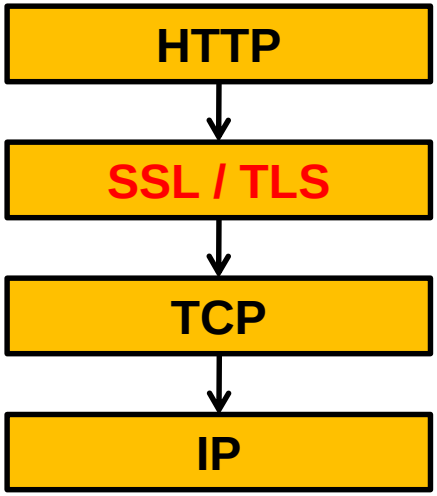
La idea es insertar una capa o protocolo intermedio entre los protocolos de aplicación (HTTP) y los de transporte (TCP/IP) que proporcione los servicios de seguridad requeridos

Ejemplo típico: protocolo SSL/TLS →
- SSL → Secure Sockets Layer
- TLS → Transport Layer Security

(Evolución directa de SSL)



SIN seguridad



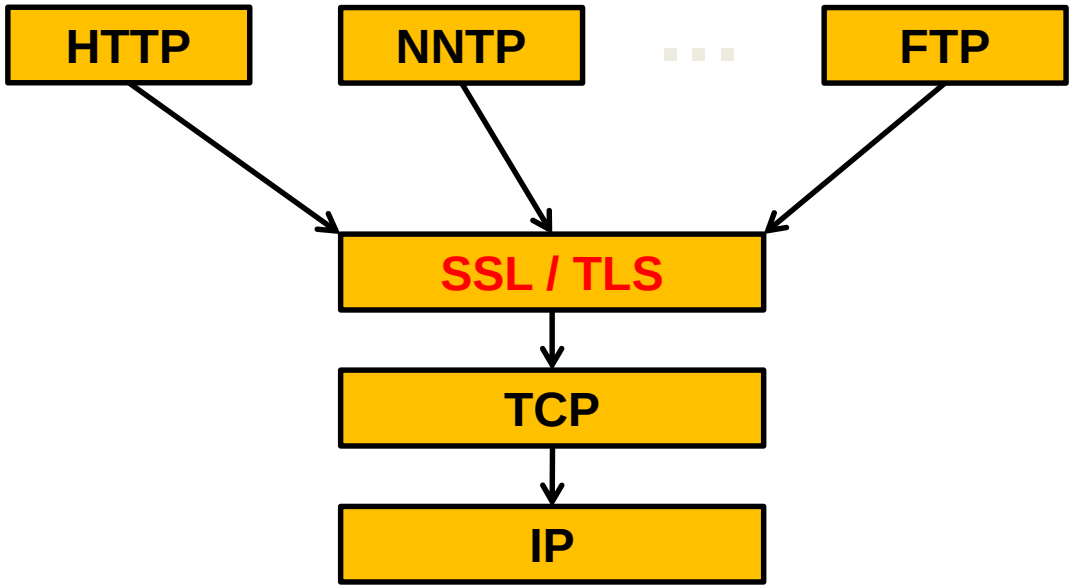
CON seguridad

Introducción a los protocolos de seguridad 3

La capa SSL se inserta en la pila de protocolos

Para la capa TCP, el protocolo SSL es un nuevo usuario de sus servicios

Aspecto importante: SSL puede proporcionar servicios de seguridad a cualquier aplicación y no solo a las basadas en HTTP



Introducción a los protocolos de seguridad 4

Las aplicaciones basadas en HTTP utilizan la capa SSL de modo similar al que utilizan la capa TCP en ausencia de seguridad

PERO …

Debido a las diferencias que existen entre usar la capa TCP o la capa SSL se modifica ligeramente el protocolo HTTP y se obtiene el HTTPS

HTTPS = HyperText Transfer Protocol Secure

Importante

El protocolo HTTPS
- **NO** proporciona la seguridad por si mismo
- Permite el uso de SSL que provee la seguridad

Diferencias

	URLs empiezan por	Puerto por defecto
HTTP	http:// …	80
HTTPS	https:// …	443

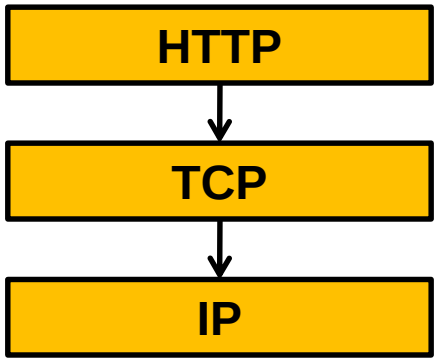
Introducción a los protocolos de seguridad 5

2) Integrar la seguridad en el protocolo de aplicación

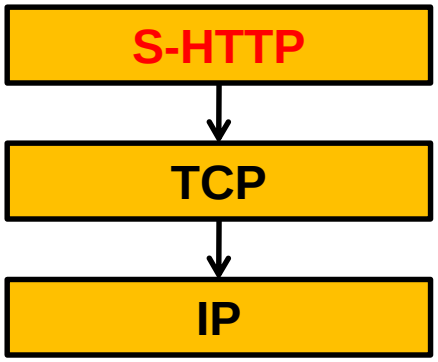
En el caso del protocolo HTTP consiste en añadir todo un conjunto de funciones de seguridad al propio protocolo que da lugar a un nuevo protocolo:

S-HTTP Secure HyperText Transfer Protocol

Fue diseñado por Rescorla y Schiffman en EIT (Enterprise Integration Technologies)
S-HTTP Tiene una difusión muchísimo menor que HTTPS
Está especificado en la RFC 2660 (Agosto 1999)



SIN seguridad



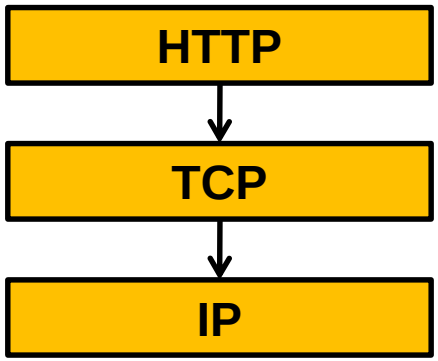
CON seguridad

3) Integrar la seguridad en el protocolo de transporte/red

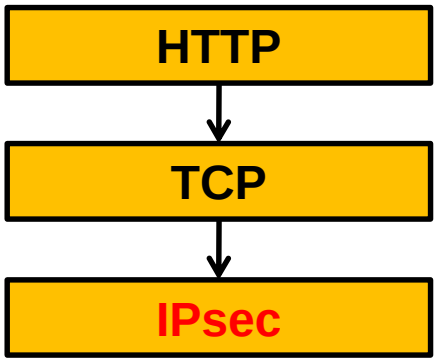
Con este enfoque los servicios de seguridad se convierten en una parte opcional del protocolo de transporte

Ejemplo típico: protocolo IPsec { Provee un conjunto completo de servicios de seguridad en el propio protocolo IP

Fue diseñado conjuntamente con IPv6 y es una extensión para IPv4
Su gran ventaja es que es independiente de las aplicaciones
Está especificado en múltiples RFCs: 4301, 4309, …



SIN seguridad



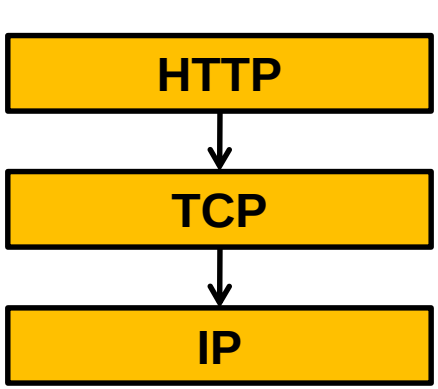
CON seguridad

4) Usar un protocolo de seguridad paralelo

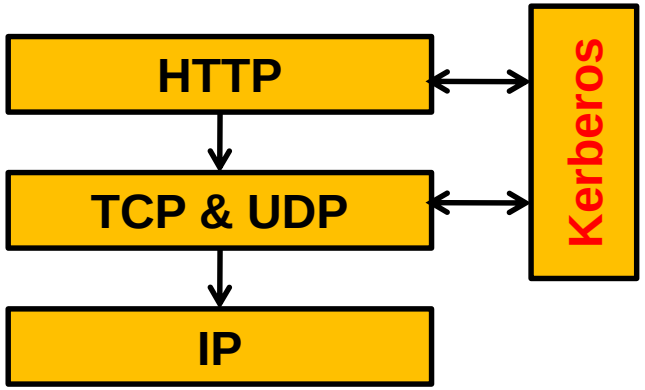
Este enfoque consiste en ejecutar un protocolo de seguridad en paralelo con los protocolos de red

Ejemplo típico: Kerberos { Desarrollado por el MIT (Massachussets Institute of Technology)

Kerberos = Toolkit usado por otros protocolos para obtener servicios de seguridad
Ej. Telnet puede usar Kerberos para identificar a un usuario con seguridad



SIN seguridad



CON seguridad

Protocolo SSL/TLS: Introducción

El protocolo SSL/TLS consiste de un conjunto de mensajes predefinidos y reglas sobre cuando enviarlos (y no enviarlos)

SSL/TLS = Secure Sockets Layer / Transport Layer Security

Objetivos

1) Establecer un canal de comunicación cifrado entre un cliente y un servidor
2) Autenticar al servidor ante el cliente
3) Autenticar al cliente ante el servidor

Desarrollo

	SSL 1.0	Desarrollado por Netscape, nunca fue publicado
1995 Feb	**SSL 2.0**	Contenía errores y se paso rápidamente a la siguiente versión
1996 Nov	**SSL 3.0**	Versión estable y operativa del protocolo
1999 Ene	**TLS 1.0**	Especificado en RFC 2246 Similar al SSL 3.0 pero no son interoperables entre ellos
2006 Abr	**TLS 1.1**	Especificado en RFC 4346 Mejora la versión anterior
2008 Ago	**TLS 1.2**	Especificado en RFC 5246
2018 Ago	**TLS 1.3**	Especificado en RFC 8446

TLS es un estándar de la IETF

RFC DataBase:
https://www.rfc-editor.org/

Protocolo SSL/TLS: Roles de los computadores

Roles de los computadores con el protocolo SSL/TLS:

UN computador es siempre un **cliente** (inicia la comunicación segura)

OTRO computador es siempre un **servidor** (responde a peticiones del cliente)

El uso más común de SSL es la navegación web segura:

El navegador web (Web browser) ←→ Cliente SSL

El sitio web (Web site) ←→ Servidor SSL

En todas las aplicaciones que usan el protocolo SSL debe haber
Un cliente y Un servidor

La distinción de cliente y servidor es muy importante →
→ Sus comportamientos son muy diferentes

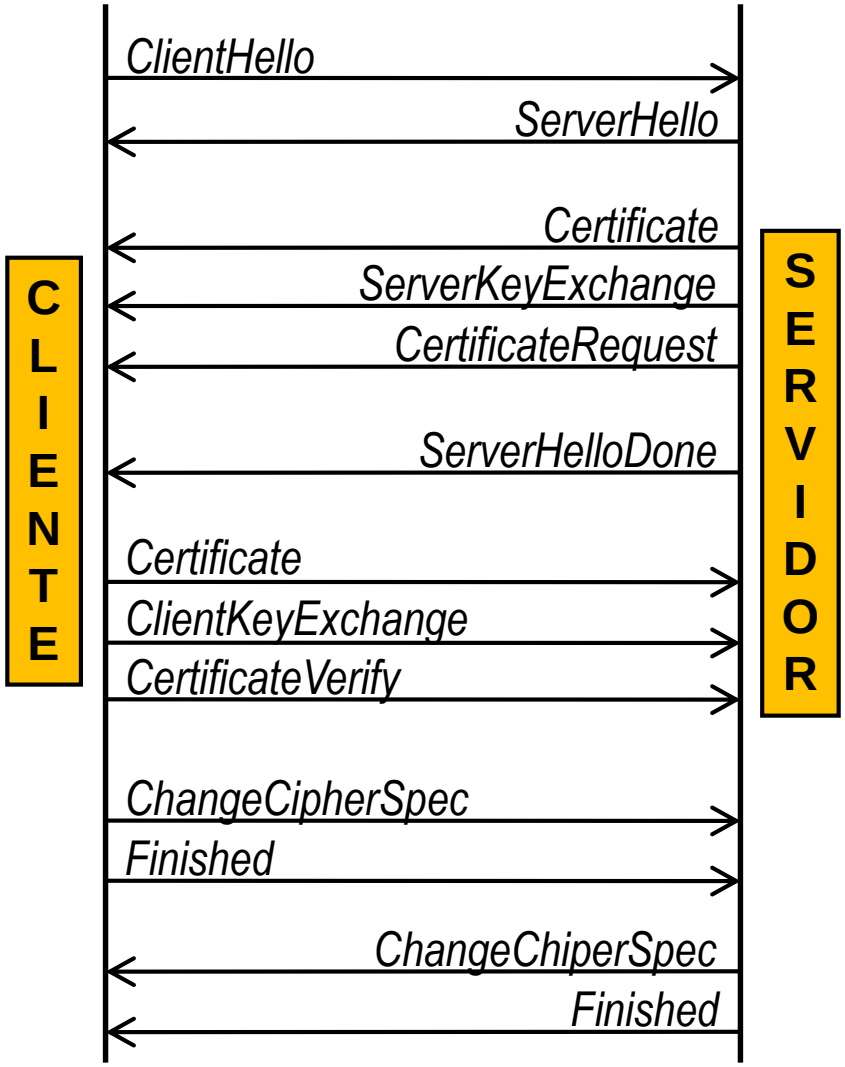
El **cliente** propone un conjunto de opciones SSL para la futura comunicación

El **servidor** elije una de las opciones propuestas por el cliente

Aunque es el servidor el que toma la decisión final

Solo puede elegir entre las opciones proporcionadas por el cliente

SSL/TLS: Todos los mensajes del **handshake**



← Aquí están todos los mensajes posibles

¡Pero NO se usan todos siempre!

Consideramos ⋯

Los 4 escenarios de operación típicos:

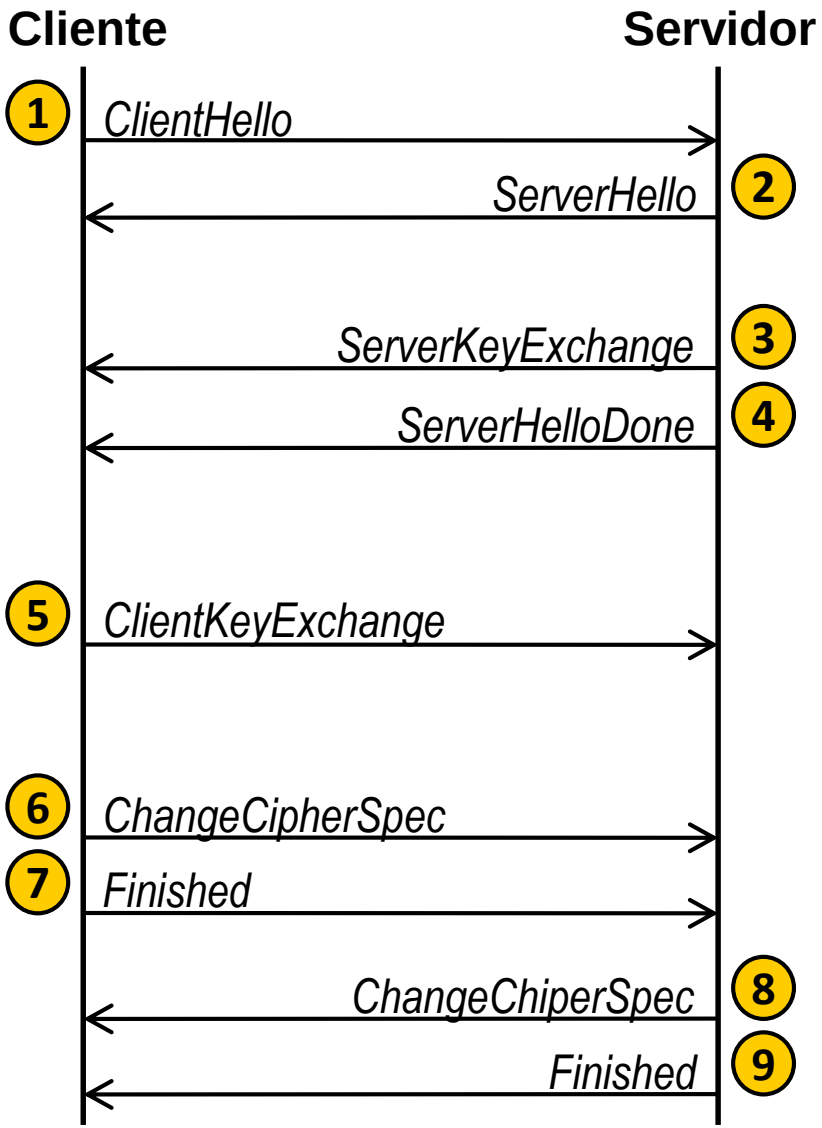
1) Comunicaciones cifradas solamente

2) Coms cifradas + Autenticar servidor
(misma clave para autenticar y cifrar)

3) Coms cifradas + Autenticar servidor
(distinta clave para autenticar y cifrar)

4) Coms cifradas + Autenticar servidor
+ Autenticar cliente

SSL/TLS handshake Solo coms cifradas



Estados del protocolo SSL/TLS

En el protocolo SSL el cliente y el servidor tienen 2 estados → { WRITE, READ }

- Estado WRITE: Define los algoritmos y clave usados para procesar los datos que envía (write) un computador

- Estado READ: Define los algoritmos y clave usados para procesar los datos que recibe (read) un computador

Además, SSL define 2 estados WRITE y 2 estados READ → { PENDING, ACTIVE }

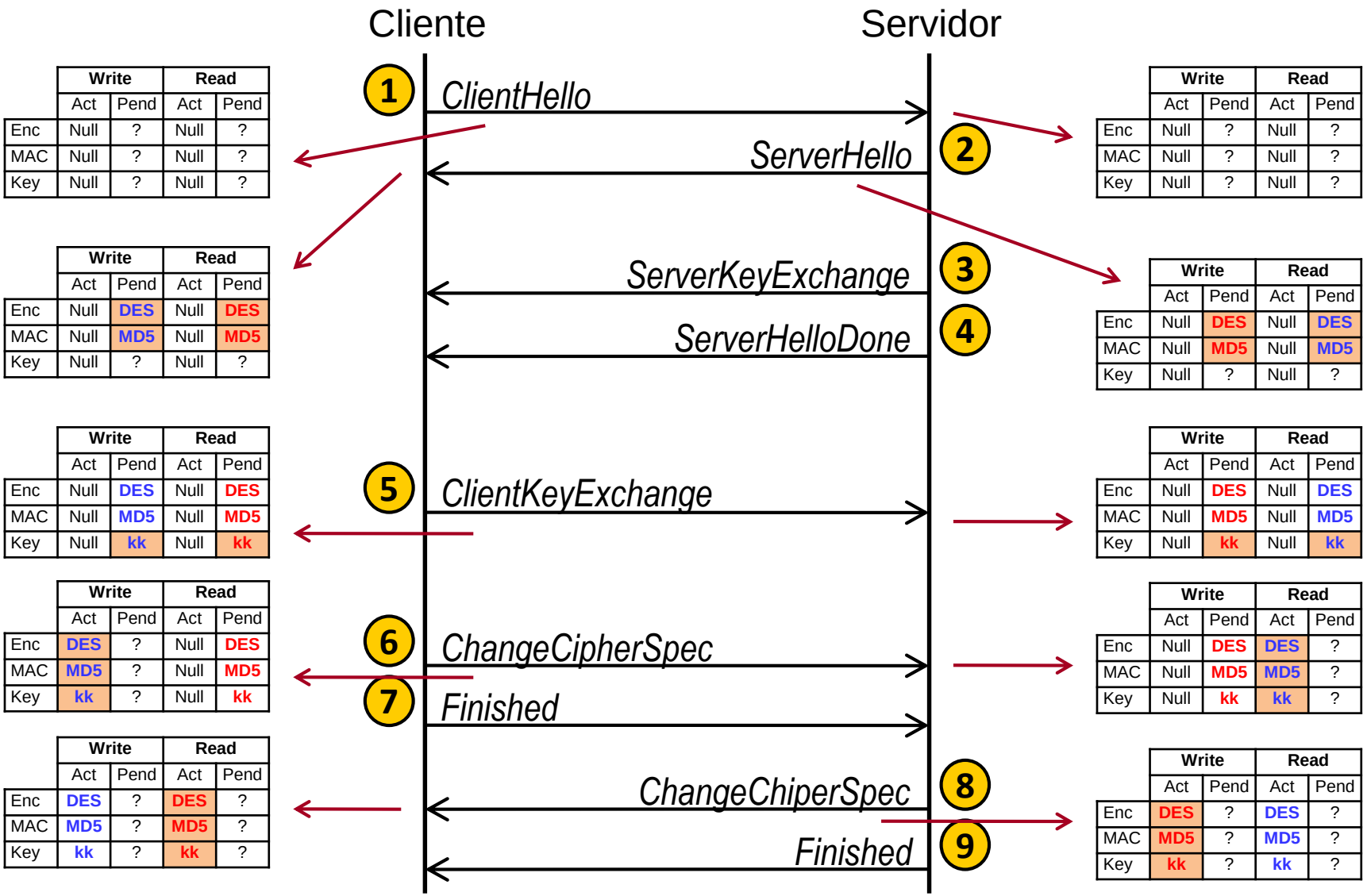
- Estado PENDING: Define los algoritmos y clave que un computador ya tiene ASIGNADOS para cifrar/descifrar los datos que envía/recibe pero cuya activación esta aún PENDIENTE

- Estado ACTIVE: Define los algoritmos y clave que un computador ya tiene ACTIVADOS para cifrar/descifrar los datos que envía/recibe

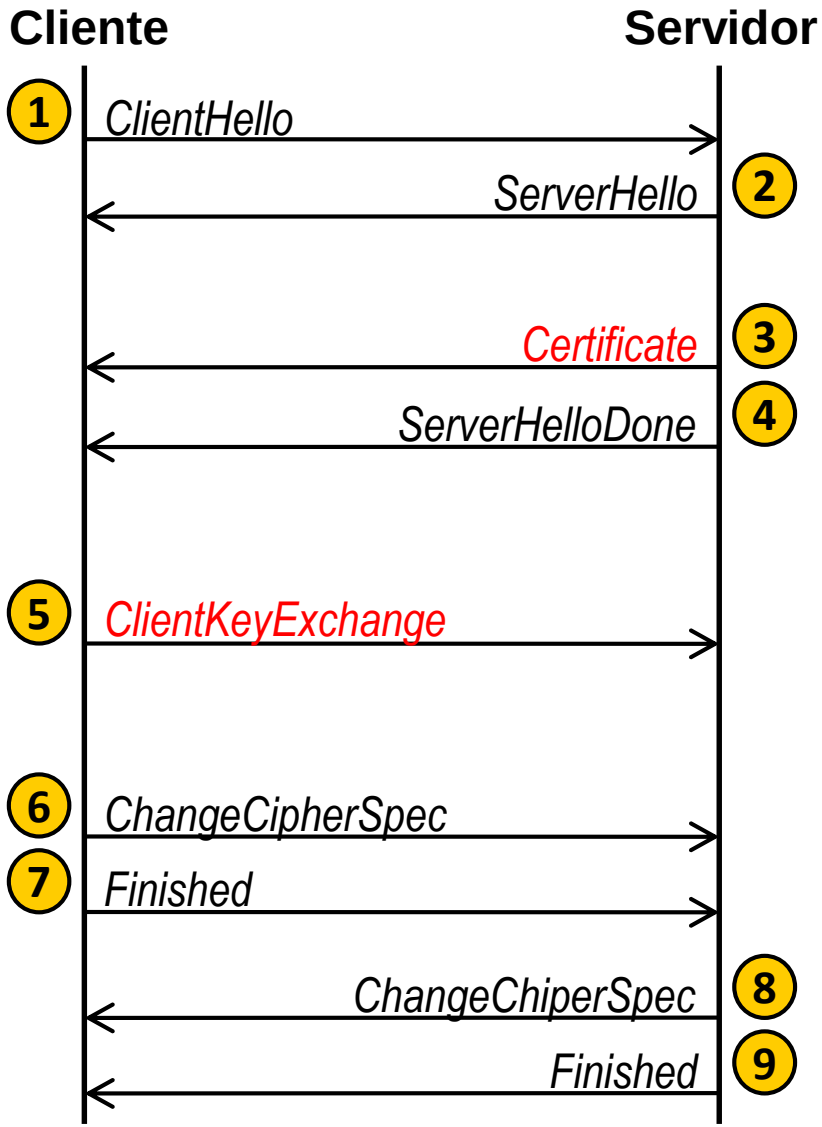
Resumen

En SSL un computador tiene 4 estados → { WRITE Pending, WRITE Active, READ Pending, READ Active }

SSL/TLS handshake Solo coms encriptadas DETALLE



SSL/TLS: Coms cifradas + Autentica servidor



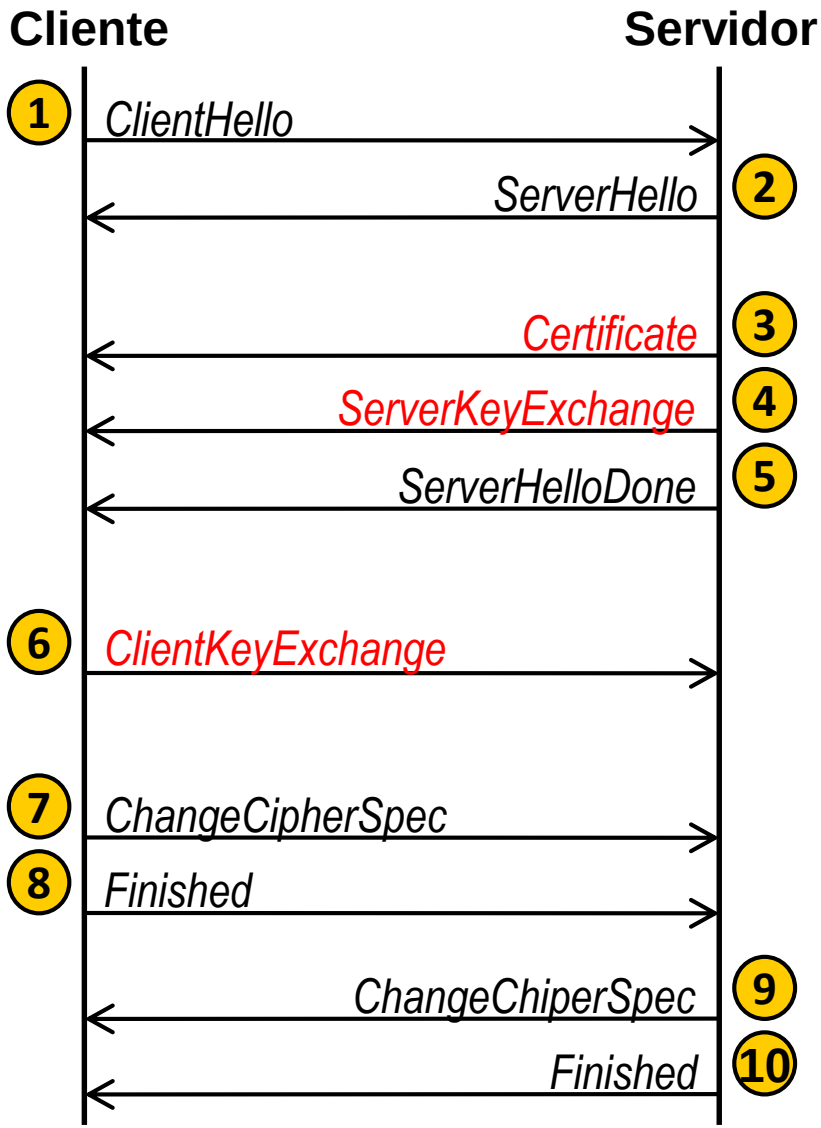
Se utiliza la **misma clave** para:
- Autenticar al servidor
- Cifrar la clave de sesión

Diferencias con el escenario previo:

1) Cambia el mensaje 3
Certificate por ServerKeyExchange
(la clave va integrada en un certificado)

2) Se modifica el mensaje 5
ClientKeyExchange es diferente
(La clave de sesión se ha cifrado con la clave pública del certificado)

SSL/TLS: Coms cifradas + Autentica servidor



Se utiliza **distinta clave** para:
- Autenticar al servidor
- Cifrar la clave de sesión

Razones para usar 2 claves públicas:

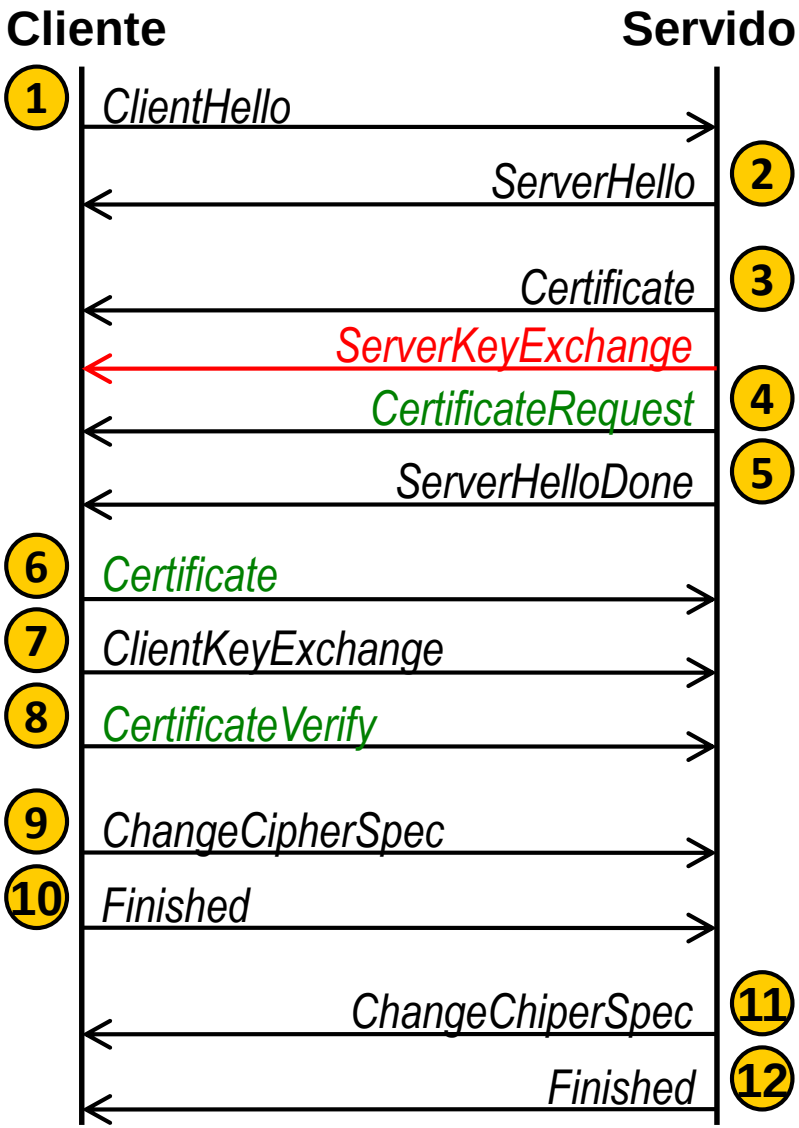
Alg de autenticación solo permite firmar *(DSA = Digital Signature Algorithm)*

Usar claves largas para autenticar y cortas (restringidas) para cifrar

Diferencias con el escenario previo:

Añade el mensaje 4 *ServerKeyExchange* conteniendo la clave pública para cifrar la clave de sesión

SSL/TLS: Coms cifradas + Autentica servidor & cliente



El servidor determina si necesita la autenticación del cliente

El cliente NO puede decidir si se autentica o no → No puede rechazar la solicitud

El servidor NO puede pedir la autenticación al cliente sin antes autenticarse el mismo

Esta propiedad del protocolo asegura que el cliente conocerá la identidad del servidor antes de revelarle la suya

Diferencia con los escenarios previos:

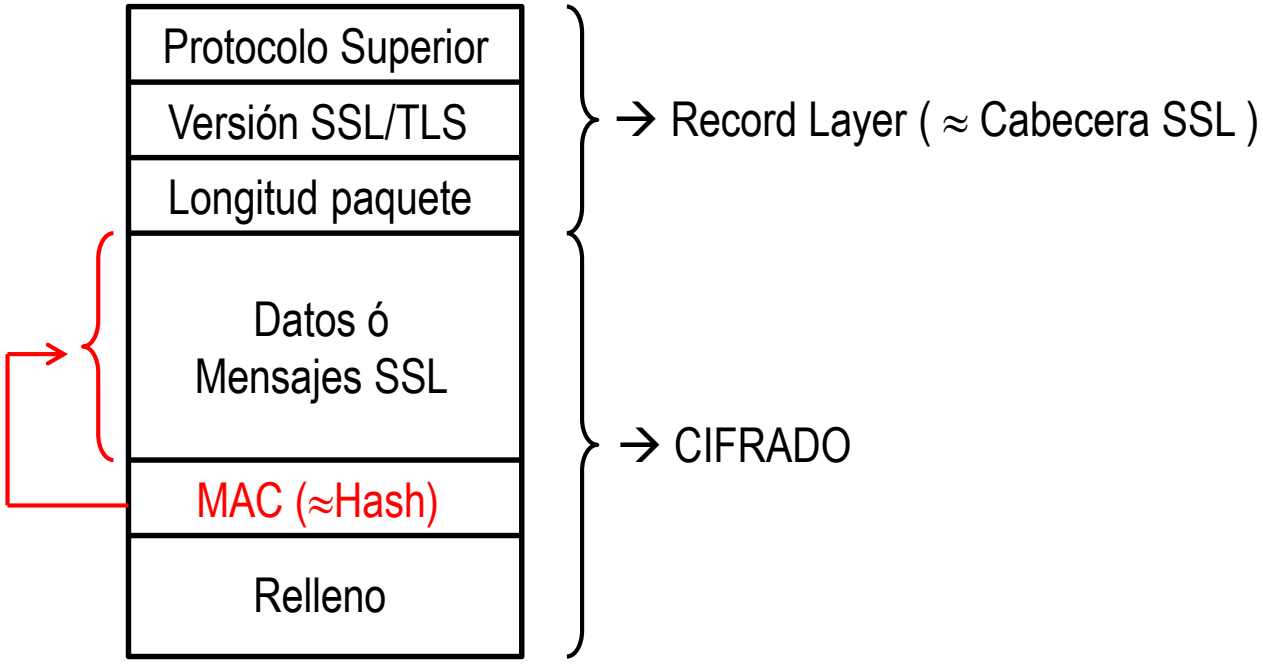
3 mensajes de autenticación del cliente

SSL/TLS: Intercambio seguro de datos

Una vez establecida la sesión segura, SSL/TLS protege los datos intercambiados mediante:

1. Generación de MAC (*Message Authentication Code*) para los datos
   Código hash MD5 o SHA a añadir a los datos para proteger su integridad
2. Cifrado de: los datos + el Hash + posible relleno (para alcanzar el tamaño del alg. de cifrado)

Formato de los paquetes intercambiados:



SSL/TLS: Finalización de una sesión

El protocolo SSL **no** dispone de un procedimiento para **finalizar una sesión** de comunicación segura entre dos computadores

NO OBSTANTE, ⋯

Cualquier computador puede enviar un mensaje ***ClosureAlert*** al otro computador

- Notifica al receptor que el transmisor NO enviará más mensajes por la conexión segura
- Cualquier mensaje recibido después de *ClosureAlert* debería ser ignorado

El mensaje ***ClosureAlert*** ayuda a detectar "Ataques por Truncación"

Por ejemplo, el transmisor envía el mensaje:

1ª Parte → Destruye todos los documentos | si no hablo contigo mañana ← 2ª Parte

Un atacante elimina la 2ª parte, el receptor solo recibe la 1ª y se cierra la comunicación

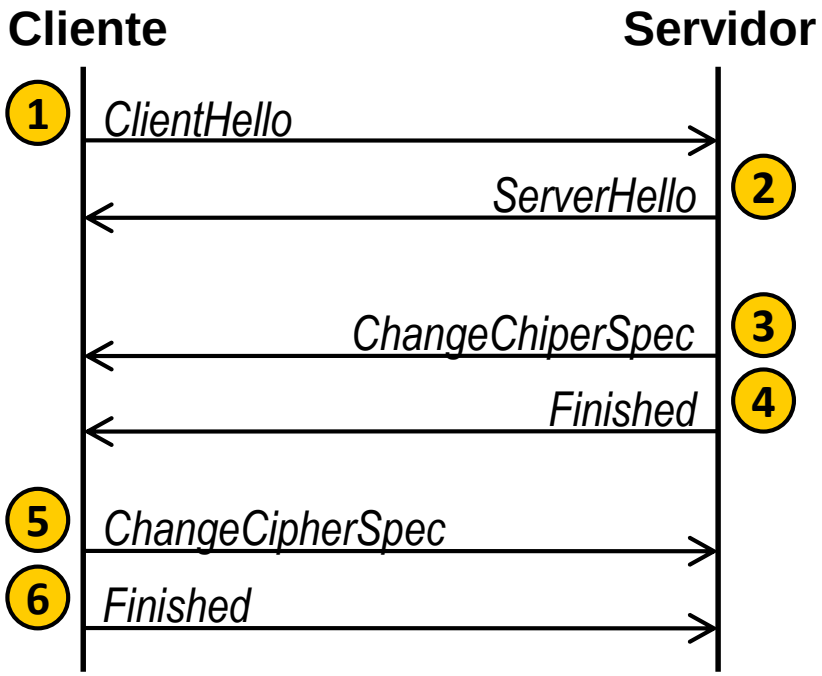
- SI el receptor NO ESPERA por un *ClosureAlert*
  No detecta problemas y destruye inmediatamente los documentos
- SI el receptor ESPERA por un *ClosureAlert*
  Detecta que puede faltar información

SSL/TLS: Reutilización de sesiones 1/2

El protocolo SSL define un mecanismo para que dos computadores reutilicen los parámetros SSL negociados en una sesión anterior

Objetivo → Minimizar la sobrecarga de mensajes y cálculos que implica la negociación SSL

La reutilización de una sesión SSL tan solo requiere de 6 mensajes:



SSL/TLS: Reutilización de sesiones 2/2

Análisis de los mensajes clave:

ClientHello

Contiene en el campo SessionID el identificador de una sesión SSL establecida anteriormente
Cuando se establece una nueva sesión el campo SessionID se deja vacío

ServerHello

Si el servidor decide …

ACEPTAR la propuesta del cliente de reusar la sesión lo indica incluyendo en ServerHello el mismo identificador de sesión que le ha enviado el cliente

NO ACEPTAR la propuesta del cliente lo indica incluyendo en ServerHello un nuevo identificador de sesión, iniciándose una nueva negociación completa

CACHE SSL

Las sesiones se pueden reusar porque sus parámetros permanecen durante un tiempo en las caches SSL del cliente y del servidor

Cuanto más tiempo pasa una sesión en la cache SSL más insegura es su reutilización

TLS 1.3 Handshake