



Protocolos de Seguridad en la capa de Transporte

Secure Socket Layer SSL

Secure Socket Layer

- SSL (*Secure Socket Layer*) es un protocolo criptográfico de la capa de aplicación
 - Proporciona autenticación, integridad y confidencialidad
 - No proporciona “No repudio”
 - Utiliza TCP
 - Transparente para las capas superiores (aplicaciones)
- Es el protocolo más utilizado en Internet para proporcionar servicios de seguridad
- Utiliza criptografía simétrica y asimétrica

Secure Socket Layer

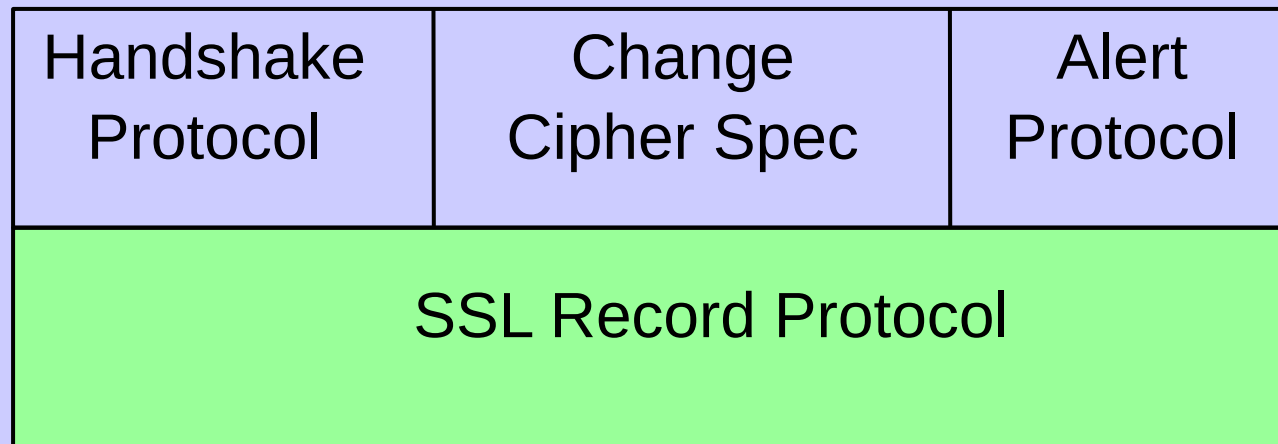
- Desarrollado por Netscape hasta la versión 3.0
 - En el año 1996, en plena Guerra de Navegadores con Microsoft
- SSLv3.0 sirve de base al IETF para TLS
 - *Transport Layer Security*, RFC 2246 (actualizado en la RFC 3546)
- Problemas de seguridad
 - Restricción en la longitud de las claves utilizadas

Secure Socket Layer

- Fácil de utilizar e implementado en muchas aplicaciones
- Solo se aplica extremo a extremo
- Otras opciones
 - implementada por las aplicaciones
 - Nivel de red seguro (IPSEC)

Protocolos de SSL

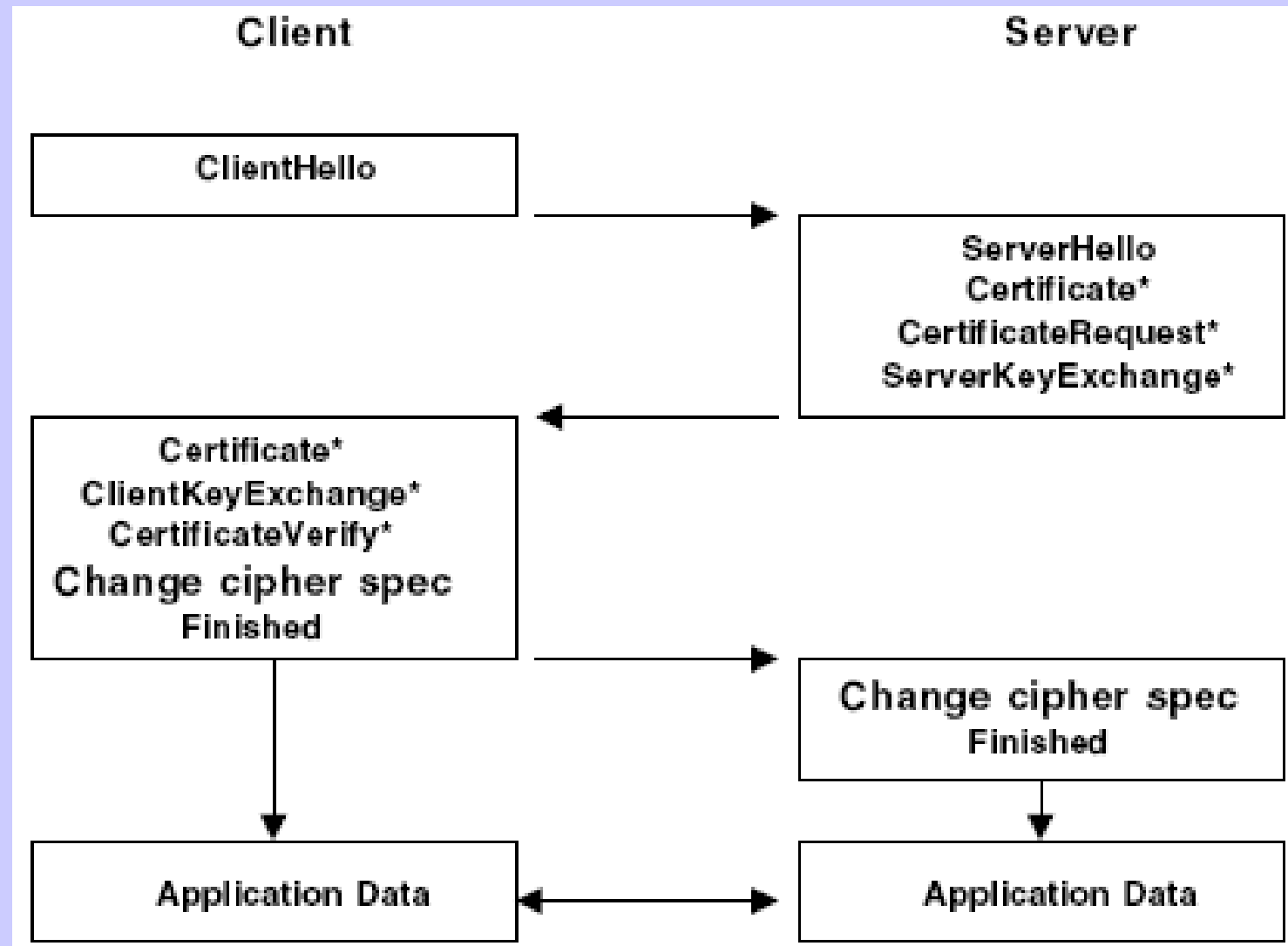
- SSL internamente consta de varios protocolos



Fase de Negociación (Handshake)

- Se negocian los algoritmos
 - Algoritmo de cifrado simétrico y asimétrico
 - Método de intercambio de claves
 - Funciones resumen
- Autenticación del servidor (obligatoria)
 - Opcionalmente se autentica al cliente

Mensajes del Handshake



ClientHello

- Versión del protocolo
- Número aleatorio
- Identificador de sesión
- Algoritmos criptográficos y de compresión

```
SSL_NULL_WITH_NULL_NULL = { 0, 0 }  
SSL_RSA_WITH_NULL_MD5 = { 0, 1 }  
SSL_RSA_WITH_NULL_SHA = { 0, 2 }  
SSL_RSA_EXPORT_WITH_RC4_40_MD5 = { 0, 3 }  
SSL_RSA_WITH_RC4_128_MD5 = { 0, 4 }  
SSL_RSA_WITH_RC4_128_SHA = { 0, 5 }  
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 = { 0, 6 }  
SSL_RSA_WITH_IDEA_CBC_SHA = { 0, 7 }  
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA = { 0, 8 }  
SSL_RSA_WITH_DES_CBC_SHA = { 0, 9 }  
SSL_RSA_WITH_3DES_EDE_CBC_SHA = { 0, 10 }
```


ServerHello

- Versión del protocolo
- Número aleatorio
- Identificador de sesión
- Conjunto de algoritmos de cifrado
- Algoritmo de compresión

Certificate

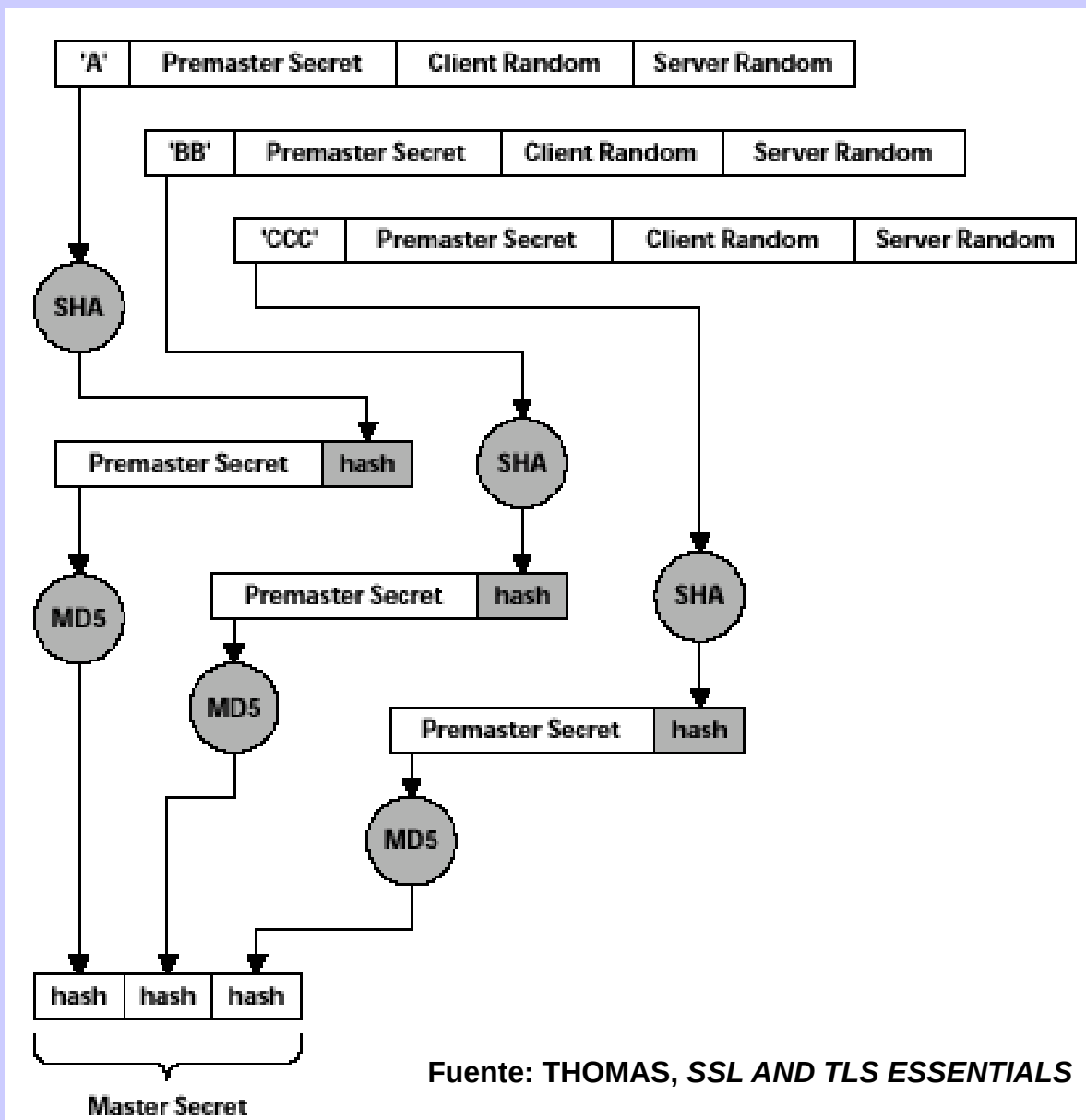
- Se envía una cadena de certificados
 - Servidor
 - Autoridades de Certificación, ...
- Para validar el certificado la otra parte:
 - debe reconocer alguna de las ACs de la cadena
 - debe comprobar que el certificado no ha sido revocado

ClientKeyExchange

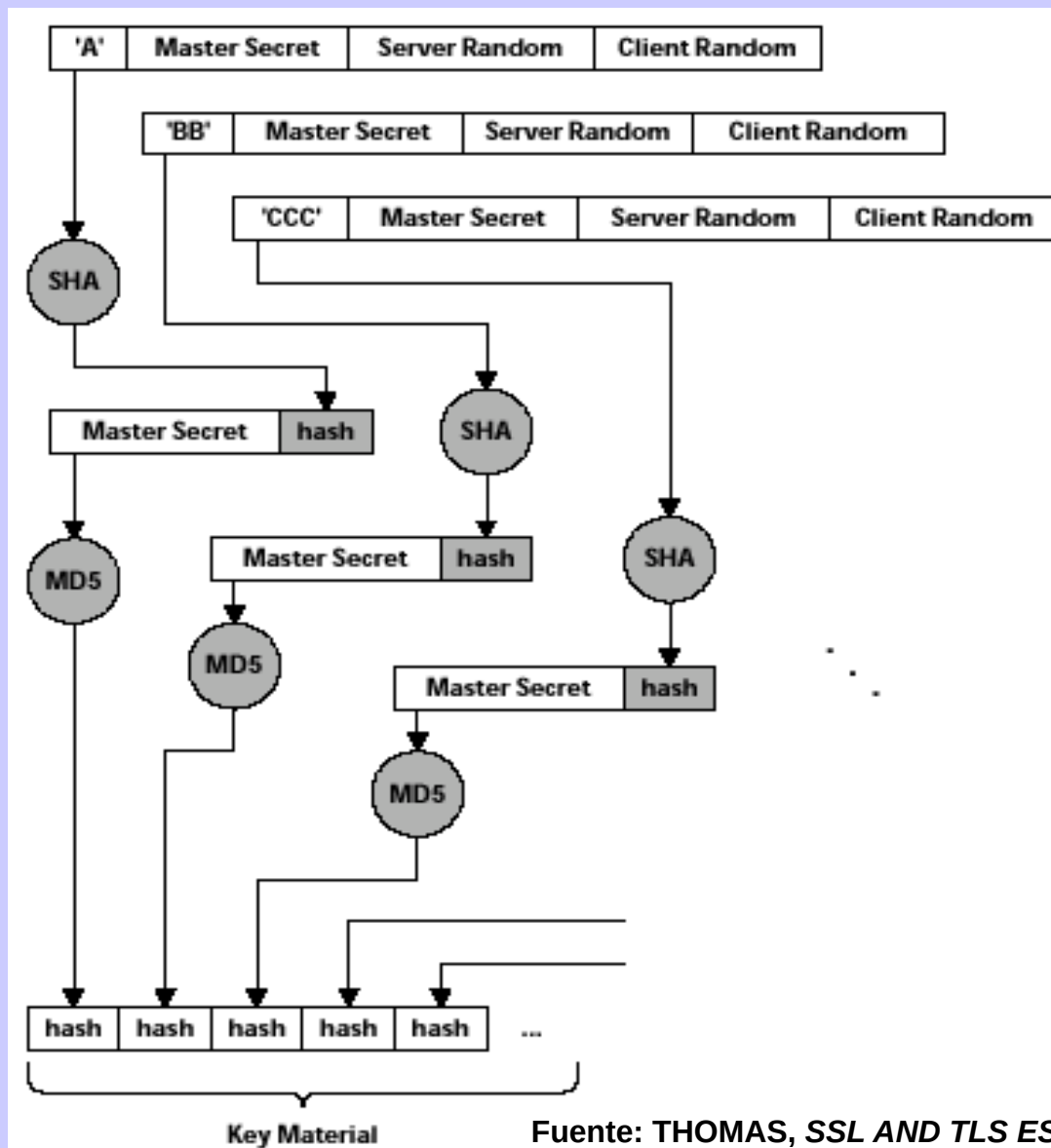
- Envía cifrado el secreto premaestro
 - Calculado por el cliente
 - Es la semilla de los algoritmos criptográficos que se utilizarán después
 - Se cifra con la clave pública del servidor
- Se genera el secreto compartido master_secret
 - basado en el intercambio pre_master_secret.

$$\begin{array}{c} \text{master_secret} = \text{hash} \left(\begin{array}{c} \text{pre_master_secret} \mid \text{N_cliente} \mid \text{N_servidor} \\ \text{48bytes} \qquad \qquad \text{32bytes} \end{array} \right) \\ \text{32bytes} \end{array}$$

ClientKeyExchange

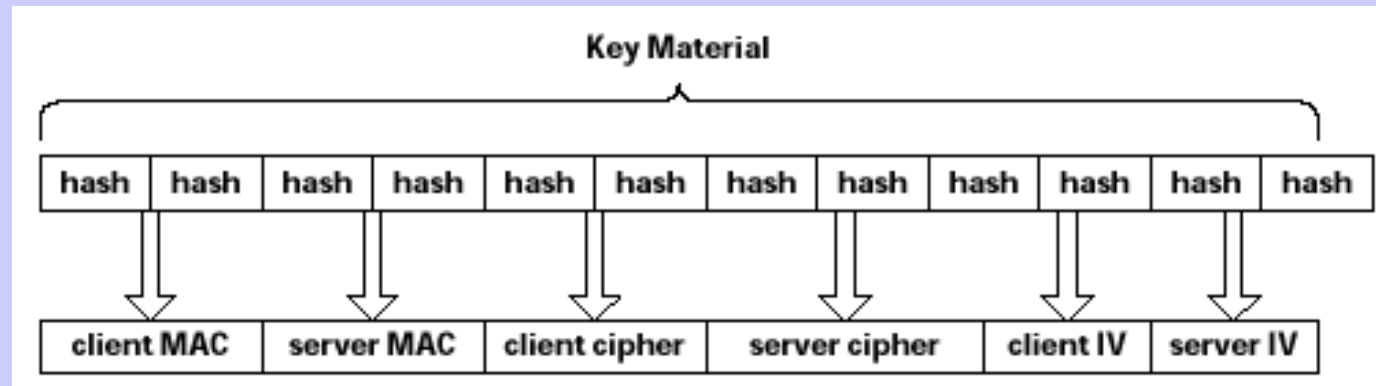


ClientKeyExchange



Fuente: THOMAS, SSL AND TLS ESSENTIALS

ClientKeyExchange



Fuente: THOMAS, *SSL AND TLS ESSENTIALS*

ChangeCipherSpec

- El cliente pasa a utilizar los algoritmos y claves negociados
 - Algoritmos aceptados por el servidor en el mensaje *ServerHello*
 - Claves calculadas a partir del secreto premaestro

Finished (cliente)

- El cliente ha terminado la fase de negociación
- Primer mensaje cifrado
 - Incluye, entre otros, un resumen del secreto maestro, los mensajes anteriores y una constante
 - El servidor puede verificar que descifra los mensajes correctamente

Alert Protocol

• Alert Protocol

- Gestiona la sesión SSL y los mensajes de error y advertencias (warning, critical, fatal):
 - *Unexpected message*
 - *Bad record MAC*
 - *Decompression failure*
 - *Bad certificate*
 - *Certificate revoked*
 - *Certificate expired*
 - *Etc...*

Datagram TLS

• Motivación

- Éxito de aplicación de TLS
- Extensión de protocolos sobre UDP (SIP, RTP, ...)
- Dificultades de aplicación para IPSEC
 - Empleo de nombres de dominio
 - Implementación en la pila de protocolos (KERNEL)
- No incrementar funciones de TLS
 - Sólo adaptarlas al entorno sin conexión
- “*Bang for the buck*”

Datagram TLS

- **Aspectos de diseño**
 - Negociación de clave sin conexión
 - Realizado sobre la misma conexión UDP
 - Deben implementarse los controles que ofrece TCP
 - Establecimiento de sesión fiable
 - Control de retransmisiones
 - Confidencialidad y control de integridad
 - Debe aplicarse a los datagramas independientes
 - Contemplar limitaciones de tamaño de los paquetes

Datagram TLS

- **Diferencias con TLS**

- Los registros (records) no se fragmentan
 - Registros de menor tamaño que en TLS

- **Control de retransmisiones**

- tiempo(16 bits) y secuencia (48 bits)
- Ventana antirepetición

- **Modos de cifrado**

- No es viable la utilización de RC 4
- Propone el empleo de CBC con IV explícito

Datagram TLS

- **Diferencias con TLS**

- Timeout y retransmisiones

- Cada nodo mantiene un reloj desde su última retransmisión
 - Dependen del RTT. Por defecto establecido entre los 500 y 1000 ms

- Cookies para evitar ataques DoS

- Consumición de recursos
 - Ataque de amplificación

-

Datagram TLS

- Ejemplo de DTLS handshake con RSA

