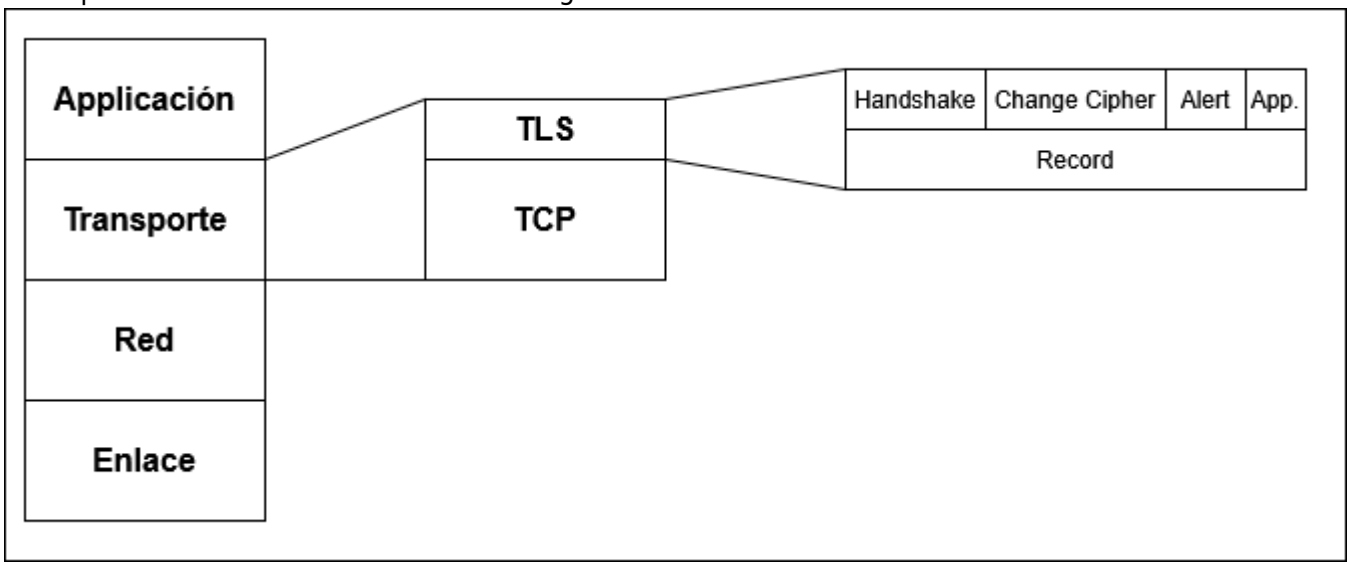
Seguridad a nivel de transporte [L4]

TLS (Transport Layer Security)

Es una evolución de SSL (Secure Socket Layer) para proveer comunicaciones seguras a través de infraestuctura insegura. Provee un canal seguro a un servicio arbitrario de internet. Garantiza autenticación, confidencialidad e integridad, tiene los siguientes objetivos:

- Seguridad Criptográfica
- Interoperabilidad
- Extensibildiad
- Eficiencia

La arquitectura de TLS en internet es la siguiente:

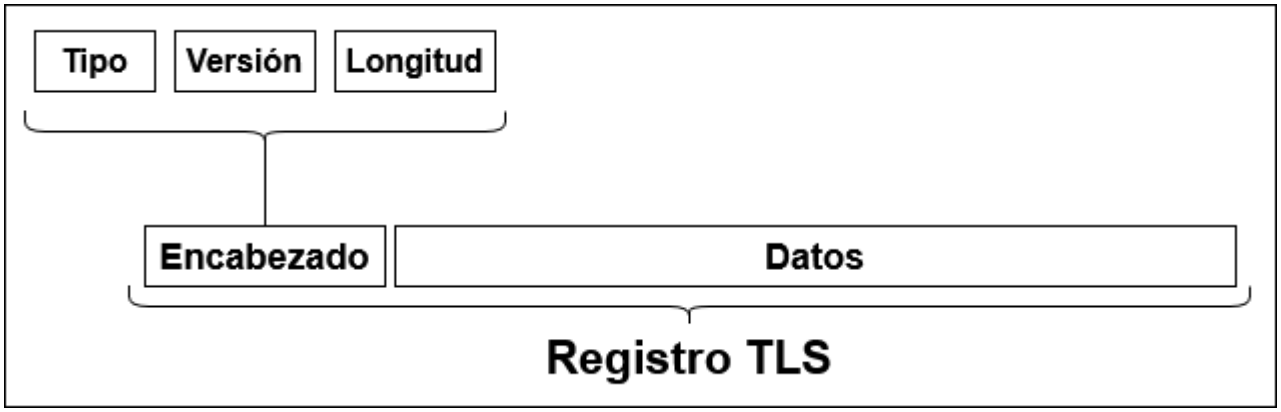


Competidores de TLS

- SSH (Secure Shell)
  - También en capa de aplicación
  - Usa cifrado de clave pública para la autenticación pero también puede usar contraseñas
  - Confiada basada en hosts conocidos e intercambio de claves en vez de PKI
  - Se suele usar para acceso remoto a servidores, transferencia de archivos o tunelado de otros protocolos.
- PGP(Pretty Good Privacy):
  - Confianza de web descentralizada en vez de usar PKI jerárquico.
  - Se suele usar para email, archivos y verificación de paquetes de software.

Protocolo TSL (1.2 y 1.3)

Transporta y, opcionalmente, cifra cada mensaje TLS entre 2 aplicaciones. Un registro TLS tiene la siguiente estructura:



- Trasnsporte de mensaje: Se transportan buffers opacos enviados por subcapas del protocolo superarioes. Puede fragmentar mensajes mayores de 16384 bytes y combinar varios mensajes pequeños en un solo regisotr.
- Cifrado y validación de integridad: Los primeros mensajes se transmiten en claro, una vez finaliza el handshake, se cifra y valida de acuerdo a los parámetros negociados
- Compresión: Ya no se usa, sujetoa a ataques de compresión de canal lateral.
- Extensibilidad: El protocolo de registro solo trata con el transporte y el cifrado, las demás tareas son llevadas a cabo por un subprotocolo. Hay 4 subprotocolos principales:
  - Handshake
  - Change cipher spec
  - Datos de aplicación
  - Alert.

Protocolo de Handshake

Responsable de la negociación de los parámetros de conexión y realizar la autenticación. Intercambia entre 6 y 10 mensajes, dependiendo de las características. Suelen haber 3 flujos comunes:

- Handshake completo con autenticación de servidor
- Handshake abreviado continuando una sesión anterior
- Handshake completo con autenticación mutua.

especificación_del_mensaje

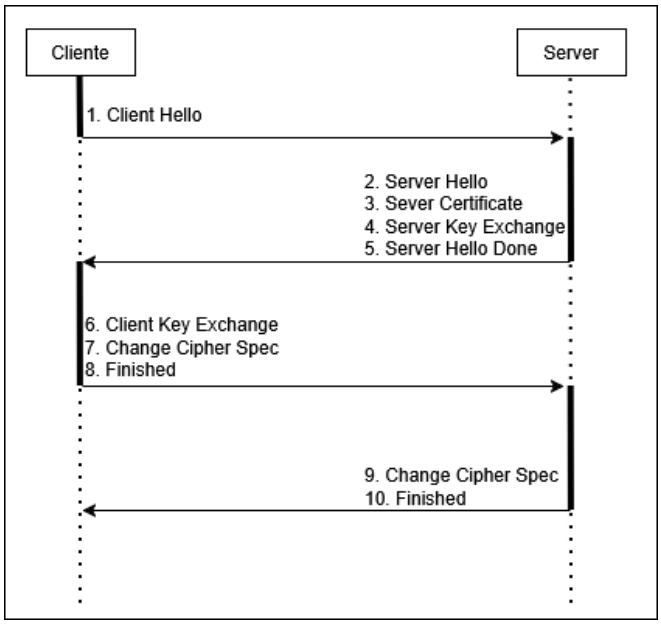
```
struct {
  HandshakeType msg_type; //1 Byte
  uint24 length;
  HandshakeMessage message; //Depende del tipo de mensaje
} Handshake;
```

El funcionamiento de un handshake con autenticación de servidor sería el siguiente:

1. Intercambio de capacidades y negociación de parámetros
2. Autenticación, se validan los certificados presentados
3. Se establece una clave secreta maestra para proteger la sesión
4. Se verifica la integridad de los mensajes de handshake

Handshake completo con atenticación de servidor

Siguen los siguientes pasos:

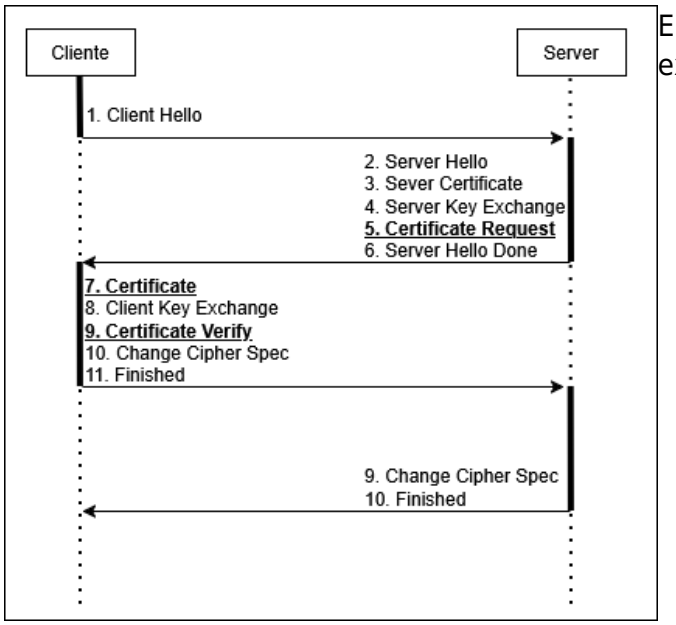


1. [Cliente] solicita sesión TLS y se envían las capabilities
    1. Client Hello: Compueto por los siguientes campos
        1. Random: Previene ataques de replay y asegura la integridad
        2. Session ID: Vacío para la primera conexión
        3. Cipher Suites: Ordenadas por preferencia.
2. [Server] selecciona los parámetros de conexión
    1. Server Hello: La versión puede ser inferior a la del cliente. Contiene solo una opción por campo.
3. [Server] envía la cadena de certificados si es necesaria autenticación
    1. Certificate Message: Sirve un certificado X.509
        1. Primero viene el certificado principal y luego los intermedios. No se debe enviar el certificado raid
        2. El certificadod epende de la cipher suite
        3. Un server puede ser configurado con múltiples certificados
4. [Server] envía información adicional para generar la clave maestra
    1. Key Exchange: Los contenidos dependen de la cipher suite.
        1. El ClientKeyExchange es obligatorio y el SeverKeyExchange es opcional.
5. [Server] Indica la finalización de la negociación
6. [Cliente] Envía información adicional para generar la clave maestra
7. [Cliente] Cambia el cifrado e informa al servidor
8. [Cliente] Envía un MAC (Message Authentication Code) para todos los mensajes intercambiados
9. [Server] Cambia el cifrado e informa al

cliente
10. [Server] Envía un MAC para todos los mensajes intercambiados

---

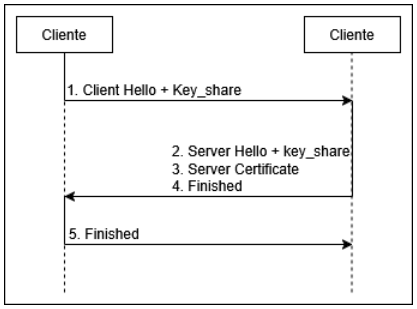
Handshake completo con autenticación mutua



En este caso tenemos 2 tipos de comunicación extra:

- Certificate Request: Se necesita que el server se auntentique. Puede usar algoritmos firmados, algoritmos de clave pública o autoridades de certificación
- Certifiate Verify: Una firma de un mensaje de handshake previo para verificar la posesión de la clave privada del certificado.

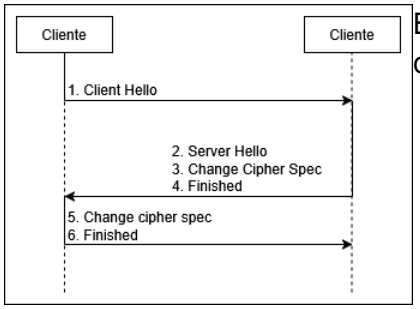
Caso típico de Handshake



- El Client Hello incluye parte de la clave de cliente. Solo DHE^a (Diffie-Hellman Ehemeral) y PSK^d (Pre-Shared Key) son soportados para la derivación de la clave

---

Continuación de Sesión



Esto reduce el tiempo necesitado para el establecimiento de conexión. El procedimiento para el ID de sesión sigue estos pasos:

1. El servidor almacena en cache prámetros de conexión de viejas conexiones
2. El cliente manda un ID de sesión con el Client Hello
3. El server confirma enviando el ID de sesión de vuelta en el Server Hello

4. Se derivan un nuevo set de claves y se activa el cifrado.

Intercambio de claves

Se usa para derivar la clave premaestra:

- Intercambio de claves RSA: El cliente eliges una clave preamestra aleatoria, la cifra con la clave pública del servidor y la enviá en un ClientKeyExchange. No provee Forward Secrecy, no usadod esde antes de TLS 1.3.
- Intercambio de claves Diffie-Hellman y de curva elíptica: Puede derivar una clave sobre un canal inseguro. Las partes deben estar autenticadas para evitar ataques Man in the Middle. Hay 2 variatnes:
  - Estático (DH): El server reutiliza parámetros, por lo que la clave compartida es siempre la misma, no provee Forward Secrecy. No usado desde antes de TLS 1.3.
  - Efímero (DHE): Los parámetros del servidor cambian entre conexiones, tiene forward secrecy.

Autenticación

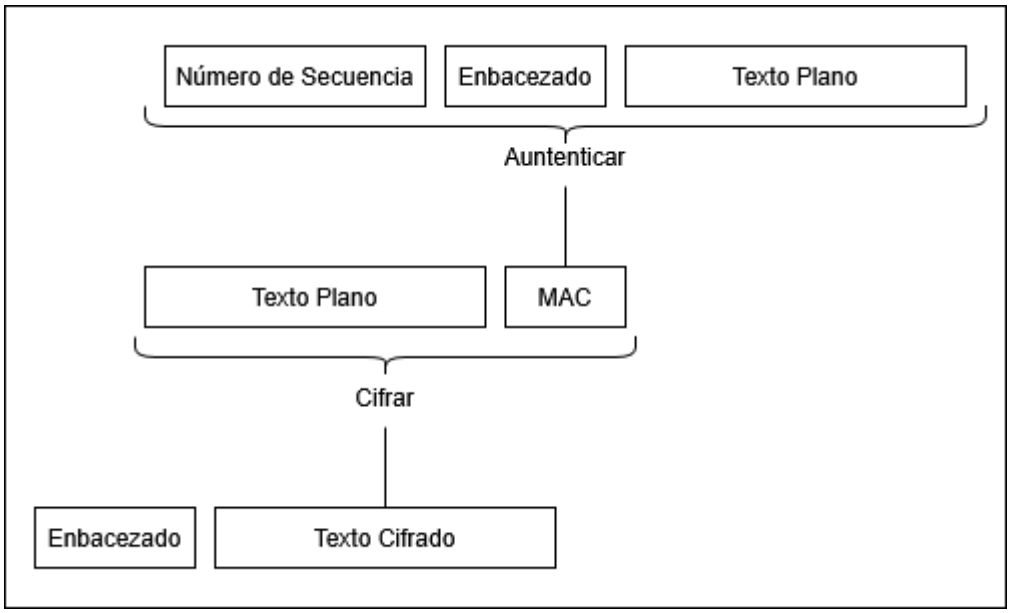
Normalmente se usa altún tipo de clave pública criptográfica. Normalmente se usa RSA, pero también el Algoritmo de Firma Digital de Curva Elíptica (ECDSA):

1. El cliente obtiene y valida el certificado del servidor.
2. Dependiendo de si es RSA o ECDSA:
   1. RSA: El cliente cifra la clave premaestra con la clave pública del servidor. El server se autentica si se recibe el mensaje “Finished” correcto.
   2. ECDSA: El Server comunica los parámetros firmados con su propia clave provada. Los parámentros se concatenan con valores aleatorios para evitar ataques replay.

Cifrado

- Cifrados disponibles
  - 2DES
  - AES
  - ChaCha20
  - ARIA
  - CAMELIA
  - RC4
  - –broken–
  - SEED
- Tipos de cibrado
  - Stream (para RC4 y ChaCha20)
  - Block
  - Auntenticado

Cifrado Stream



From:
https://knoppia.net/ - **Knoppia**

Permanent link:
https://knoppia.net/doku.php?id=master_cs:secom:tm1_v2&rev=1779655396

Last update: **2026/05/24 20:43**