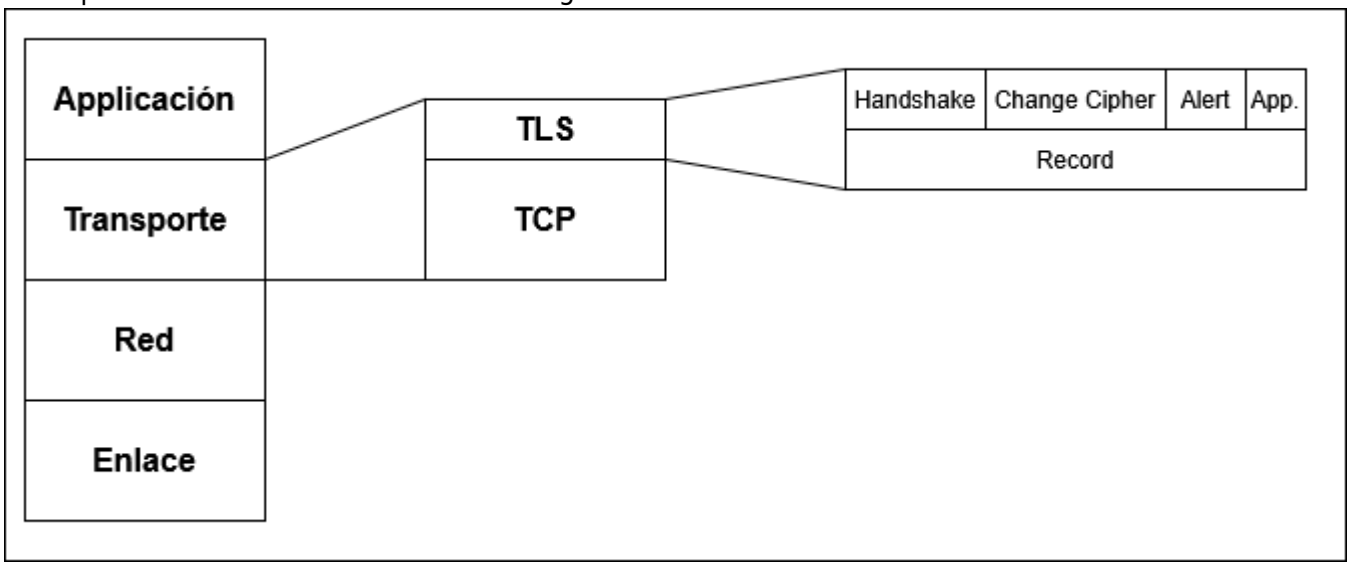
Seguridad a nivel de transporte [L4]

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:

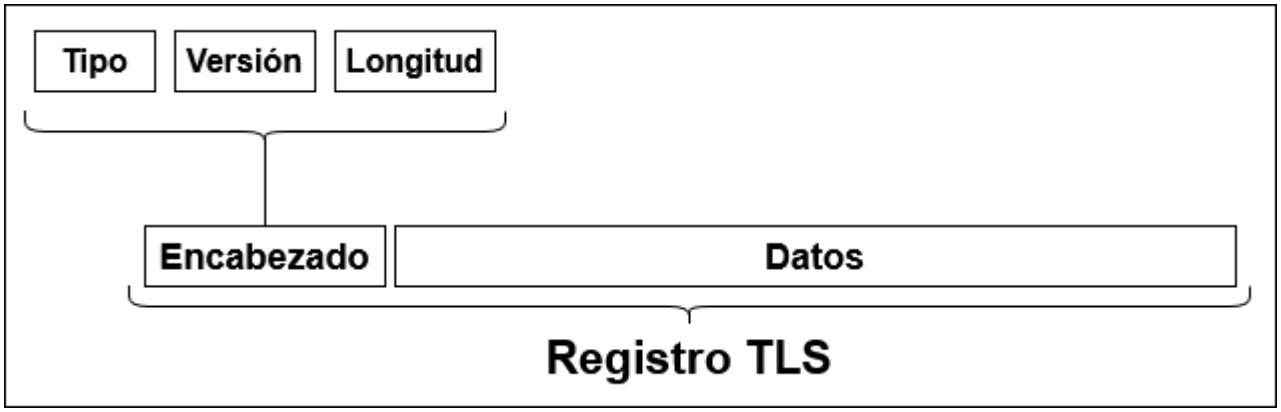


Competidores de TLS

- 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

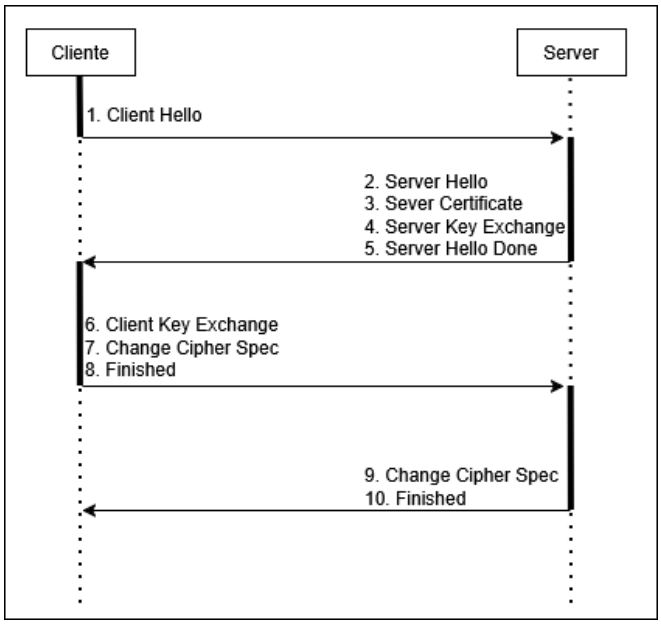
```
struct {
  HandshakeType msg_type; //1 Byte
  uint24 length;
  HandshakeMessage message; //Depende del tipo de mensaje
} Handshake;
```

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:

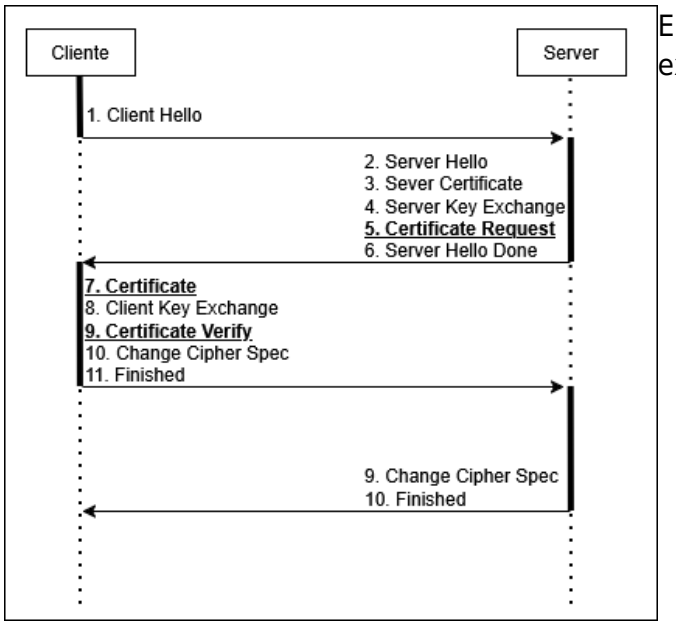


1. [Cliente] solicita sesión TLS y se envían las capabilities
    1. Client Hello: Compueto por los siguientes campos
        1. Random: Previene ataques de replay y asegura la integridad
        2. Session ID: Vacío para la primera conexión
        3. Cipher Suites: Ordenadas por preferencia.
2. [Server] 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

---

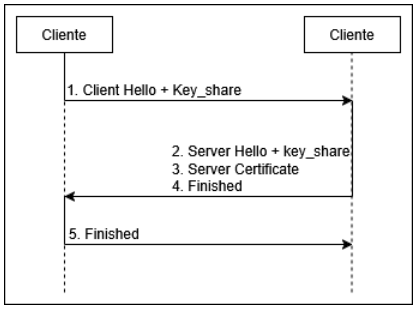
Handshake completo con autenticación mutua



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.

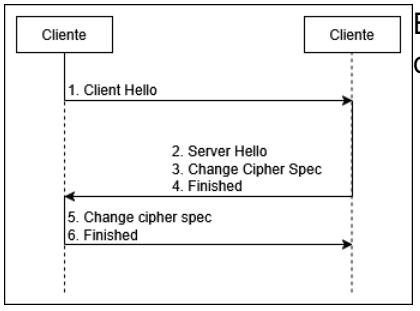
Caso típico de Handshake



- 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

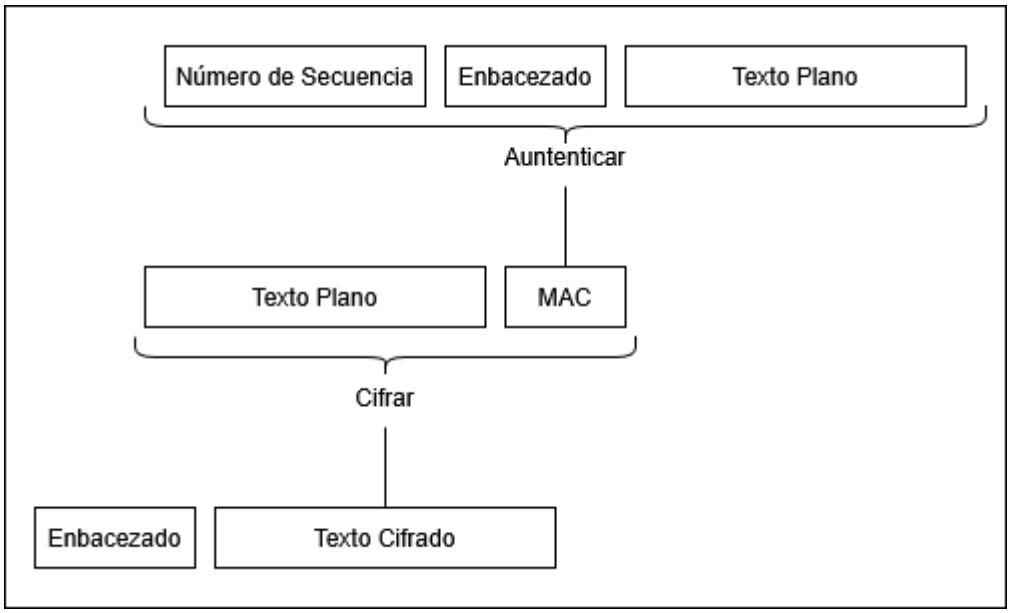
Normalmente se usa altún tipo de clave pública criptográfica. Normalmente se usa RSA, pero también el Algoritmo de Firma Digital de Curva Elíptica (ECDSA):

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.

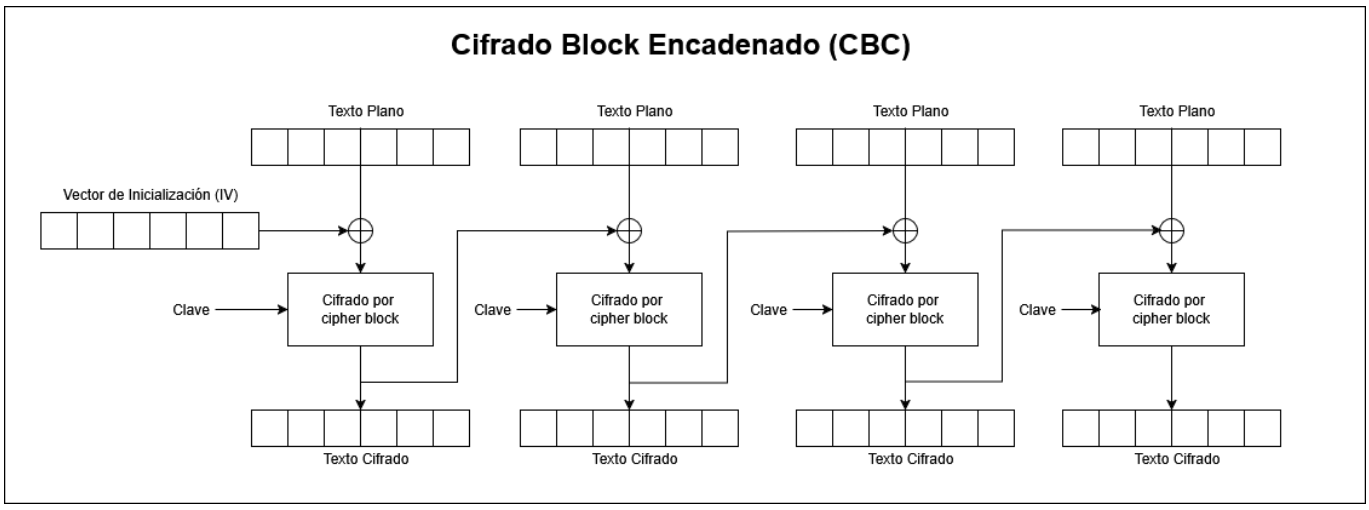
Cifrado

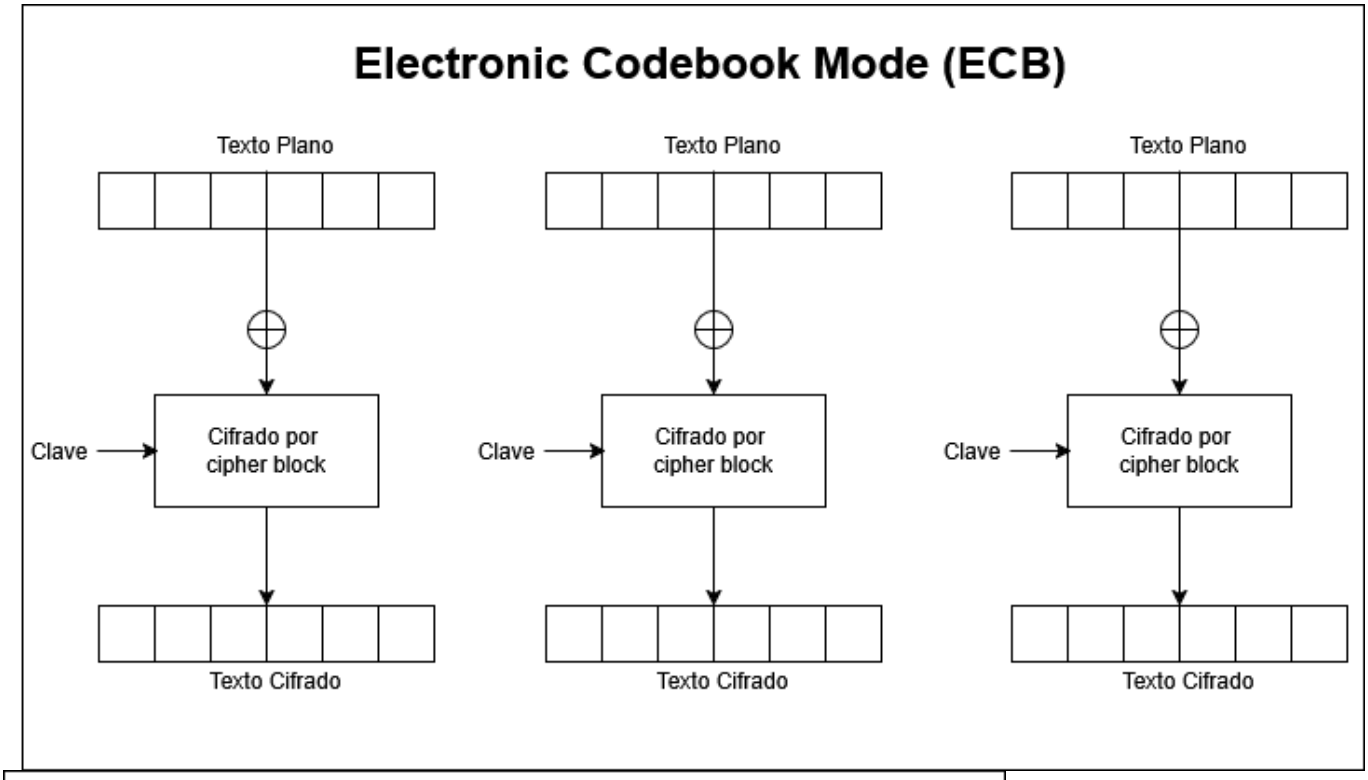
- Cifrados disponibles
  - 2DES
  - AES
  - ChaCha20
  - ARIA
  - CAMELIA
  - RC4
  - –broken–
  - SEED
- Tipos de cibrado
  - Stream (para RC4 y ChaCha20)
  - Block
  - Auntenticado

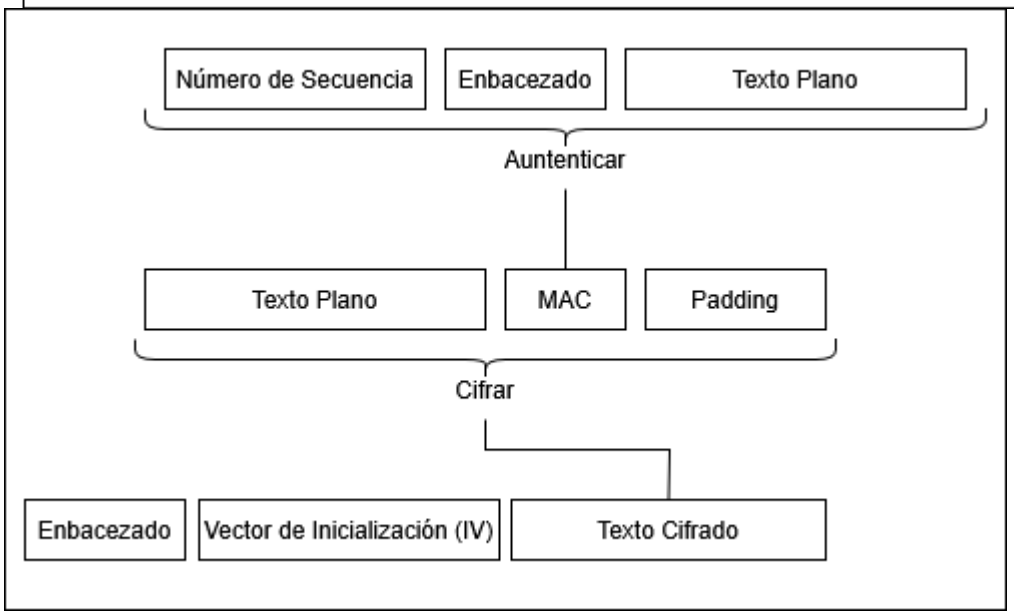
Cifrado Stream



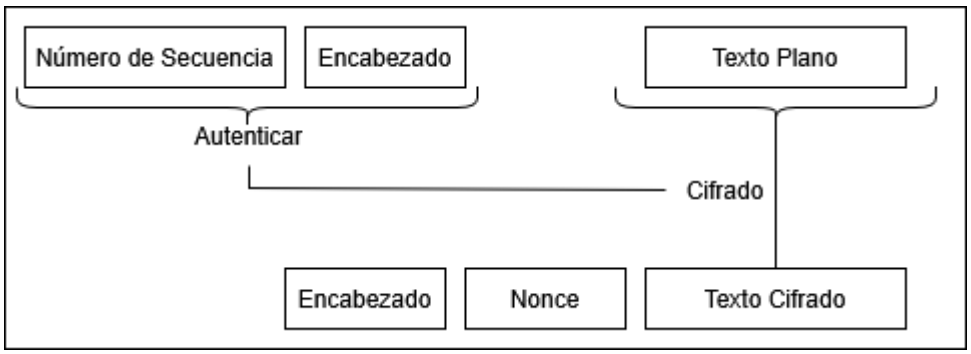
Cifrado Block







Cifrado Autenticado con Datos Asociados (AEAD)



Cierre de la conexión

- Subprotocolo de alerta: Mecanismo de notificación con dos niveles:
  - Fatal: Conexión cerrada de golpe
  - Warning: Hay una descripción, el receptor puede enviar una alerta “Fatal” en respuesta. El mensaje Close Notify“ es una alerta warning para cerrar una conexión ordenadamente. Evita ataques de truncado.

Operaciones Criptográficas

Una funcion Pseudo-Aleatoria (PRF) genera cantidades arbitrarias de datos pseudoaleatorios. en TLSv1.2 se basa en HMAC (Hash-Based MAC) y SHA256 (Secure Hash Algorithm). TLSv1.3 usa HKDF (Hashed Message Authentication Code)

```
PRF(Secret, label, seed) = P_hash(secret, label⊕seed)
```

Siendo ⊕ la concatenación de la operación.

```
$$ P_{\text{hash}}(secret, seed) = \underset{i=1}{\bigoplus} \mathrm{HMAC}_{\text{hash}}
\left(secret, A(i) \oplus seed\right) $$
```

P_hash es una función de expansión de datos donde:

- A(1) = seed
- A(i) = HMAC_hash(secret, A(i-1))

Internet Public Key Infraestructure (PKI)

La meta de la PKI es permitir a personas que nunca se han conocido comunicarse de forma segura. Los objetivos que tiene PKI son los siguentes:

- Acceso a claves públcias
- Información sobre la validez de las claves públicas (Revocación)
- Escalabilidad

Se basa en terceras partes, en este caso, Autoridades Certificadoras (CA), estas emiten certificados con claves públicas. X.509 Es un estándar internacional para PKI adaptado para ser usando en internet.

Normalmente los certificados tienen los siguientes campos:

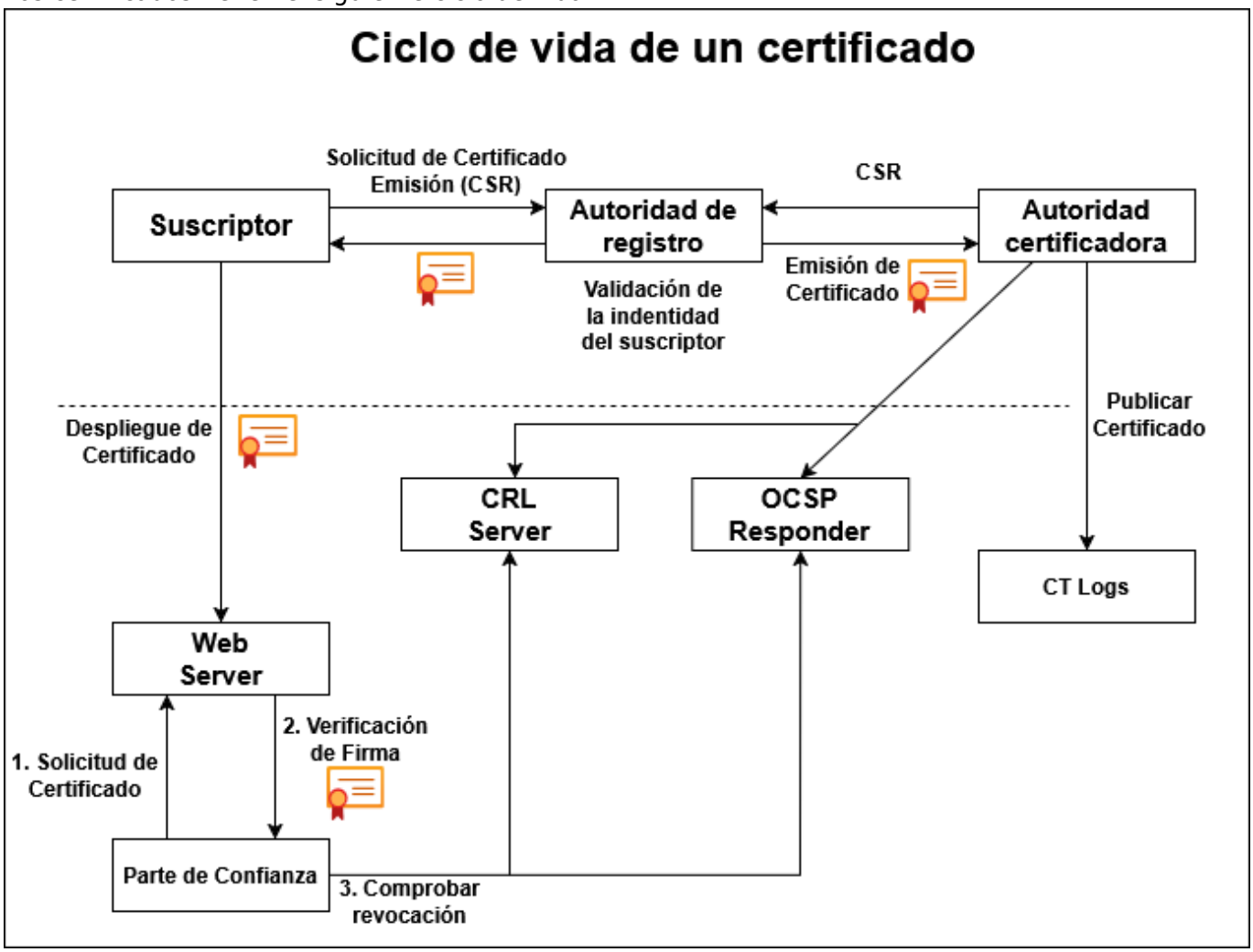
- Versión: 1, 2 o 3. En la actualidad solo la versión 3 es válida
- Número de Serie: ID único, no secuencial, no predecible con al menos 64 bits de entropía.

- Algoritmo de firma: Se usa para firmar el certificado, es parte del certificado para protegerlo
- Emisor: Nombre distinguible (DN) del emisor del certificado
- Validez: Inicio y fin del período de validez del certificado
- Subject: DB de la entidad a la que pertenece la clave del certificado. (Obsoleto)
- Public Key: Contiene el ID del algoritmo, parámetros opcionales y la propia clave pública.

Se puede extender el certificado con las siguientes extensiones:

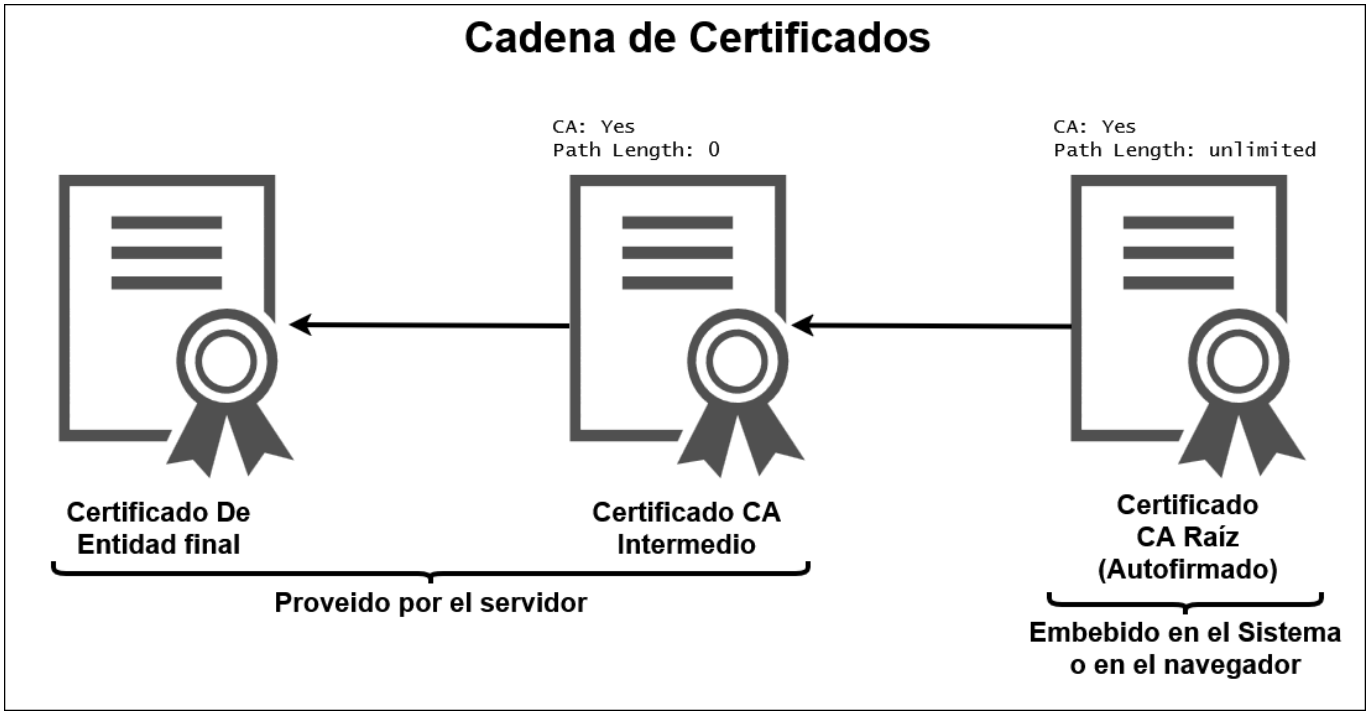
- Subject Alternative Name: Sustituye el campo subject, soporta múltiples identidades especificadas por nombre DNS, dirección IP o URI.
- Name Constraints: Limita las identidades para que la CA puede emitir certificados
- Basic Constraints: Limita la profundidad de la cadena de certificados suboridnada.
- Key Usage: usa el certificado de un un set restringido conocido.
- Extended Key Usage: Puede detallar temas ás específicos con propósitos arbitrarios
- Certificates Policies: Liste de enlaces donde se puede obtener todo el texto de la política.
- CRL Distribution Points: Ubicación de la Lista de certificados revocados (CRL)
- Authority information Access: Varias URI, entre ellas la localicación del OCSP responder.
- Subject Key Identifier: Identificador único para la clave pública, normalmente un hash.
- Authority Key Identifier: Identificador único de la clave que firmó el certificado. Debe coincidir con el SKI del que firmó el certificado.

Los certificados tienen el siguiente ciclo de vida:



Infraestructura PKI

- Solicitud de firma de certificado: Lleva la clave pública del suscriptor. Prueba de propiedad de la correspondiente clave privada. La clave privada no debe ser NUNCA comunicada. Puede llevar metadatos adicionales, pero suelen ser sobreescritos por la CA.
- Validación de indentidad del Suscriptor: Hay varias estrategias de validación:
  - Validación de Dominio (DV): Prueba de control de un nombre de dominio dado.
    - Envío de email a direcciones bien conocidas
    - Adición de registro a la zona del dominio
    - ACME challenge.
    - Puede ser automatizada y realizada en segundos.
  - Validación de Organización (OV): Requiere verificación de identidad y verificación. Puede haber inconsistencias en el procedimiento y en el codificado de la información.
  - Validación Extendida (EV): Requiere identidad y autenticidad con requerimientos muy estrictos. Alternativa a Certificados OV. Puede tomar días o semanas.
- Revocación: Se realiza cuando se sospecha que la clave privada ha sido comprometida o el certificado ya no es necesario.
  - Certification Revocation List (CRL): Lista el número de serie de certificados revocados no expirados.
  - Online Certificate Status Protocol (OCSP): Permite a las partes interesadas revisar el estado de un certificado. Tiene problemas de rendimiento y privacidad. El OCSP stapling permite al servidor adjuintar una respuesta OCSP al handshake TLS.
- Let's Encrypt
  - Los puntos de distribución CRL ahora son certificados hoja.
  - URLs OCSP eliminadas de nuevos certificados
  - CRLs particionados: 128 archivos por CA intermediaria
  - Los buscadores usan CRLs agregados, no suelen buscar directamente.



- Seguridad de la clave Root de la CA: Es crítico, si se revoca, todos los certificados que dependen de esta deben ser emitidos de nuevo. Los buscadores requieren que las CA sean operadas manualmente.
- Cross Certification: Una nueva raiz firmada por una vieja raiz mientras es desplegado en

sistemas y navegadores
- Compartimentación: Dividir la raiz entre múltiples CA subordinadas reduce riesgos. Las CA subordinadas pueden ser usadas para la emisión de certificados automatizada.
- Delegación: Una organización puede querer emitir sus propios certificados. Una CA piede emitir certificados subordiandos para ello.

Partes de Confianza

Se debe confiar en una serie de certificados CA raiz por sistemas operativos o desarrollador, por ejemplo:

- Apple: mismas raices para IOS y MacOS. La CA debe pasar una auditoría anual.
- Microsoft: Igual que apple pero para productos de microslop
- Mozilla: Opera un programa de certificados raiz transparente.
- Chrome: Usado en la tienda de Chrome OSm tiene restricciones adicionales:
  - Blacklist de raices no confiables
  - Whitelist de CAs que pueden emitir certificados EV
  - Se requiere transparencia de certificados para todos los certificados públicos.

Problemas con la infraestructura PKI actual

- Emisión de certificados del dueño del controlador de dominio:
  - Cualquier CA puede emitir un certificado para una CA sin permiso del dueño ya sea por negligencia o malicia
  - Hay cientos de CA, uno solo comprometido puede afectar a la seguridad.
- Dificultades actualizando Trust Stores:
  - Muchas CA son demasiado grandes como para fallar:
    - Eliminar una CA tiene consdecuencias demasiado grandes para el ecosistema
    - Normalmente una CA no puede ser eliminada
  - Las CA raíz son de confianza o no
  - Procedimientos alternativos:
    - Prohibir certificados de CA a de una fecha específica
    - Eliminar privilegios EV.
- Revocación rota
  - Retrasos en la propagación: La información en CRL y OCSP puede mantenerse válida hasta 10 días
  - Soft fail policy: Los navegadores ignoran fallos en comunicaciones CRL o OCSP. Un atacante puede suprimir dichas comunicaciones para usar certificados revocados
  - Browser Vendors: Por defecto no realizan revocaciones.
  - Una medida temporal es usar una blacklist propietaria en los navegadores.
- Otras debilidades:
  - Weak Domain Validation: En email inseguro o usando datos WHOIS inseguros.
  - Certificate Warnings: Muchas apps y librerías se saltan la verificación de los certificados. Los buscadores permiten a los usuarios aceptar certificados no verificados.

Mejoras de infraestructura

- Notarías: Repositorios públicos de certificados conocidos. Puede prevenir ataques basados en

CA intermediarias comprometidas. Logs de transparecnia de certificados (CT Logs)
- DNS Information: DANE (DNS-Based Authentication of Named Entities) y CAA (Certification Authority Authorization)
- HPKP (HTTP Public Key Pinning): Permite a los usuarios de un sitio restringir que CA pueden emitir certificados de sus sitios
    - Cabecera HTTP especial: Respetado por firefox, tras varias visitas el navegador va a rechazar certificados de otras CA
    - Mecanismos propietarios: Google chrome tiene un mecanismo propietario para que los dueños de los witios web puedan mandar sus requisitos.
    - HPKP tiene un alto riesgo de inutilizar una web si no se maneja correctamente
    - HPKP esta obsoleto en la actualidad.
    - Expected-CT header es una práctica recomendada,
- Transparencia de certificados (CT): Identifiación rápida de certificados frauduletos. Es un framework para las partes interesadas para verificar la emisión de los certificados.
    - Bajo CT, todas las CA participantes deben enviar tiodos los certificados emitidos en un log público de forma que todo el mundo puede monitorizar la emisión de certificados.
    - La CA obtiene una prueba digital firmada de envío al log público.