Autoridades de certificación y certificados digitales

Pilar Navarro Ramírez, Antonio Gámiz Delgado, Silvia González Rodríguez

20 de diciembre de 2020

Índice

1. ¿Qué es una Autoridad de Certificación (CA)?

Una **autoridad de certificación, certificadora, o certificante** (AC o CA por sus siglas en inglés) es una entidad de confianza responsable de validar las identidades de las entidades (como sitios web, direcciones de correo electrónico, empresas o personas individuales) y vincularlas a claves criptográficas mediante la emisión de documentos electrónicos conocidos como certificados digitales. Estos documentos, concretamente los certificados de clave pública, asocian una clave pública a unos datos que representan la identidad de una entidad que posee la clave privada asociada a dicha clave pública. Cuando se usa la clave privada asociada a la clave pública se puede asegurar, atendiendo a la autoridad de la Autoridad de Certificación, que ha sido usada por el legítimo propietario del certificado. De esta forma, otras entidades pueden confiar en las firmas o en afirmaciones asociadas a la clave privada correspondiente a la clave pública incluida en el certificado. La autoridad de certificación actúa pues como un tercero en el que confía tanto el propietario del certificado como la entidad que confía en el certificado.

Una tarea común de las autoridades de certificación es firmar los certificados usados en el protocolo HTTPS para garantizar la seguridad de las comunicaciones digitales. También son usadas para resguardar documentos digitales, por ejemplo utilizando firmas electrónicas.

2. Modo de funcionamiento

2.1. Solicitud de un certificado

El mecanismo habitual de solicitud de un certificado consiste en que la entidad solicitante completa ciertos datos identificativos propios (nombre de dominio, organización, dirección de correo electrónico, país, ciudad, etc.), que varían dependiendo del nivel de validación o el propósito del certificado, y genera una pareja de claves pública/privada, manteniédose la clave privada en secreto (no se le muestra a la CA ni a nadie más, a diferencia de la llave pública). A continuación, se crea una petición de firma de certificado (CSR, Certificate Signing Request), que es un archivo de texto codificado que incluye la llave pública y los datos identificativos de la entidad que se quieren incluir en el certificado. La generación de las llaves y la CSR se lleva a cabo normalmente en el servidor o estación de trabajo donde se quiere instalar el certificado.

Una vez generada la CSR, el solicitante se lo envía a la CA elegida. Esta tiene entonces la obligación de verificar las credenciales del solicitante de manera que otros usuarios y entidades puedan confiar en la información que aparece en el certificado emitido por ella. Tras realizar la verificación, la CA firma digitalmente el certificado con su clave privada y lo envía al solicitante.

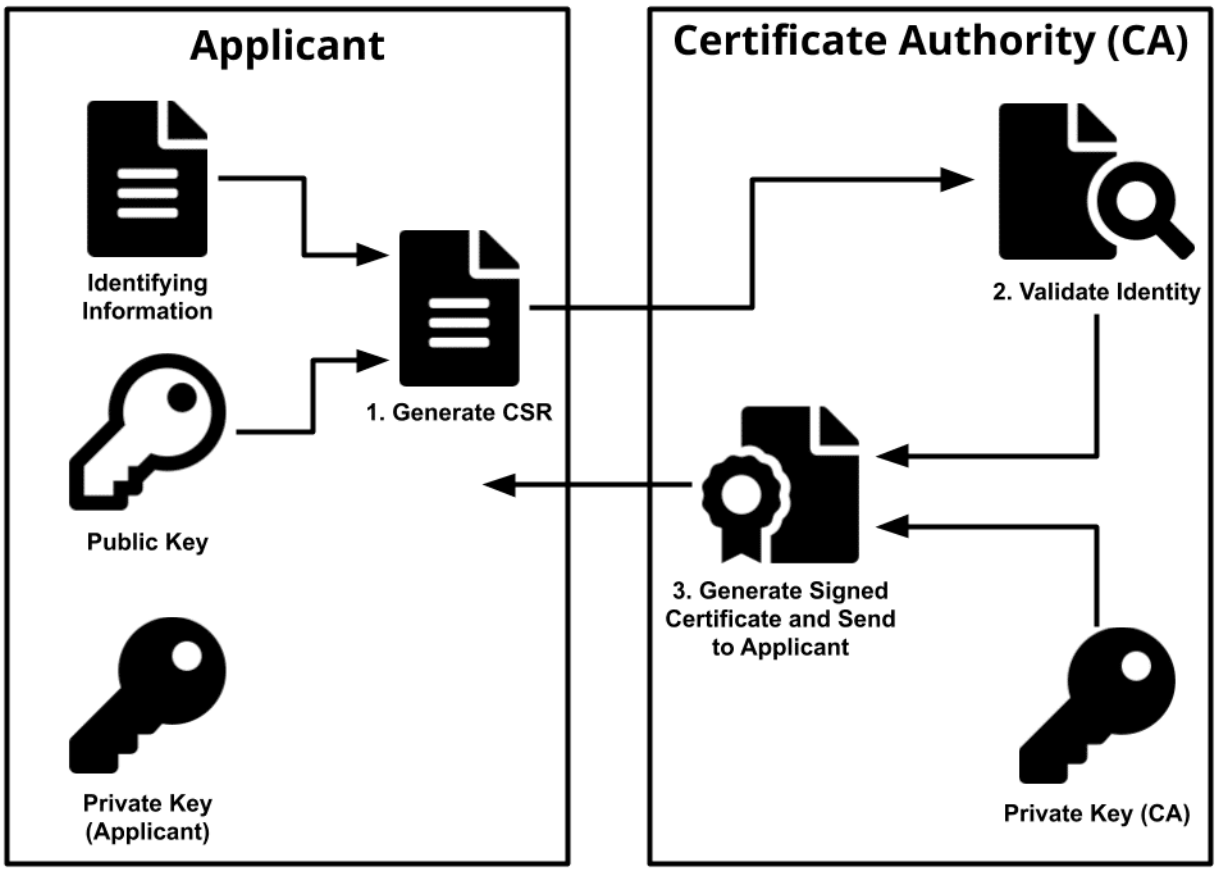

Figura 1

2.2. Validación de la autenticidad de un certificado

Cuando se presenta un certificado firmado por una CA a un tercero , el destinatario puede confirmar la firma de la CA a través de su llave pública. Si un usuario confía en la autoridad de certificación y puede verificar su firma, puede asumir también que la llave pública incluida en el certificado emitido por dicha CA corresponde a quien aparece como propietario en el certificado y que la información no ha sido alterada desde que se firmó.

En la validación de la autenticidad de un certificado digital tienen lugar los siguientes pasos:

1. El primer paso es averiguar si la CA es de confianza. Para ello, se mira el nombre de la misma en el certificado y se compara con una lista de CA's de confianza, por ejemplo provista por el navegador web. En el caso de que el nombre se encuentre en esta lista, el destinatario del certificado (que está llevando a cabo la validación) obtiene la llave pública correspondiente a la CA.

2. Se valida la firma digital que aparece en el certificado, la cual es básicamente el hash de la llave pública de la CA.

3. Para validar la firma digital, el destinatario obtiene el hash de la clave pública de la CA usando el mismo algoritmo que utilizó la CA para obtener la firma digital.

4. Si los dos hashes coinciden la firma digital es válida y el certificado es auténtico. Si no coinciden el certificado no es válido y no se puede autenticar.

5. Se comprueba la fecha de expiración del certificado, la cual también es importante para saber si el certificado sigue siendo válido.

6. Se debe determinar además si el certificado ha sido revocado, en cuyo caso no sería válido (en el siguiente apartado explicamos este proceso con más detalle)

7. Una vez que el certificado ha sido autenticado, la identidad de su propietario también puede ser autenticada.

2.3. Revocación de certificados

Las CAs también se encargan de la gestión de los certificados firmados. Esto incluye las tareas de revocación de certificados, que puede solicitar el titular del certificado o cualquier tercero con interés legítimo ante la CA, así como la gestión asociada a la renovación de certificados por caducidad o revocación.

2.3.1. Lista de Revocación de Certificados

Una **lista de revocación de certificados**, conocida por sus siglas en inglés **CRL** (Certificate Revocation List), contiene un registro de los números de serie de certificados digitales que han sido revocados por una CA y está firmada digitalmente por dicha CA. Es responsabilidad de la autoridad de certificación publicarla y actualizarla debidamente.

Cuando un tercero desea comprobar la validez de un certificado debe descargar una CRL actualizada desde los servidores de la misma autoridad de certificación que emitió el certificado en cuestión. A continuación debe comprobar la autenticidad de la lista gracias a la firma digital de la autoridad de certificación. Después se verifica si el número de serie del certificado cuestionado está en la lista. En caso afirmativo, no se debe aceptar el certificado como válido. Todo este proceso se incluye en los pasos para determinar la validez de un certificado enumerados en el apartado anterior.

No es necesario descargar una CRL cada vez que se verifica un certificado, sino solamente cuando no se dispone de la CRL de una entidad de certificación concreta o cuando dicha lista tiene una cierta antigüedad que aconseja su renovación.

Si una CA emite muchos certificados, corre el riesgo de que sus CRL sean de gran tamaño, lo que hace poco práctica su descarga para los terceros que confían. Además, si la lista no está actualizada, es posible que un certificado haya sido revocado y no aparezca en la CRL del

tercero que comprueba su validez. Por estos motivos y otros se han desarrollan mecanismos alternativos de consulta de validez de los certificados, como el protocolo OCSP.

2.3.2. Protocolo OCSP

El protocolo de comprobación del estado de un certificado en línea u Online Certificate Status Protocol (OCSP) en inglés, es un método para determinar el estado de vigencia de un certificado digital X.509 usando otros medios que no sean el uso de CRL.

Los mensajes OCSP se codifican en ASN.1 y habitualmente se transmiten sobre el protocolo HTTP.

Una petición de OCSP está compuesta por la versión del protocolo y los identificadores de los certificados que se quiere que sean validados (número de serie, hash de las credenciales del emisor del certificado y hash de la clave pública del mismo). En una petición se puede solicitar la consulta del estado de varios certificados, incluso pertenecientes a diferentes CAs.

Un servidor OCSP puede devolver una respuesta firmada, lo cual significaría que el certificado indicado en la petición es "bueno"(good), revocado"(revoked) o "desconocido"(unknown). Estas respuestas serán para cada uno de los certificados de los que se ha solicitado la consulta. También puede devolver un código de error, en cuyo caso la respuesta no tendría que estar firmada.

A diferencia de las CRL, OCSP proporciona información más reciente sobre el estado de revocación de un certificado y elimina la necesidad de que los clientes tengan que obtener y procesar las CRL. Por otra parte, una consulta sobre el estado de un certificado sobre una CRL, debe recorrerla completa secuencialmente para decir si es válido o no, mientras que un servidor OCSP usa un motor de base de datos para consultar el estado del certificado solicitado, con todas las ventajas y estructura para facilitar las consultas. De esta manera, cuando el tamaño de la CRL es muy grande, la búsqueda es mucho más eficiente en un servidor OCSP. Sin embargo, una CRL puede estar disponible sin conexión si está almacenada localmente, pero para usar OCSP se requiere conexión con el servidor OCSP.

3. La Jerarquía de Certificación

Los certificados digitales son validados usando una cadena de confianza. Las CA disponen de sus propios certificados públicos, cuyas claves privadas asociadas son usadas por las CA para firmar los certificados que emiten. Un certificado de CA puede estar auto-firmado cuando no hay ninguna CA de rango superior que lo firme. Este es el caso de los certificados de CA raíz, el elemento inicial en esta cadena de confianza.

Una jerarquía de certificación es una estructura de certificados que permite a los usuarios verificar la validez de un emisor de certificados. Los certificados son emitidos y firmados por

certificados que se encuentran más arriba en la jerarquía de certificación, de forma que la validez y fiabilidad de un cierto certificado está determinada por la correspondiente validez del certificado que lo firmó.

En esta jerarquía se parte de un certificado auto-firmado, correspondiente a la CA raíz, que se encargará de validar todos y cada uno de los certificados emitidos por la CA. Este certificado, sin embargo, no firmará los certificados finales (por ejemplo certificados de persona física, persona jurídica, SSL, etc.), sino que se empleará únicamente para firmar los denominados Certificados Subordinados o Certificados Intermedios (que representan a CAs intermedias), y estos últimos son los que firmarán los certificados finales. Así pues, en la cadena de confianza jerárquica nos encontramos con un certificado raíz, seguido de uno o varios certificados intermedios y en último lugar tenemos los certificados de los usarios finales. Se muestra un diagrama ejemplo del modelo jerárquico para ilustrar el funcionamiento de la cadena de confianza de certificados:

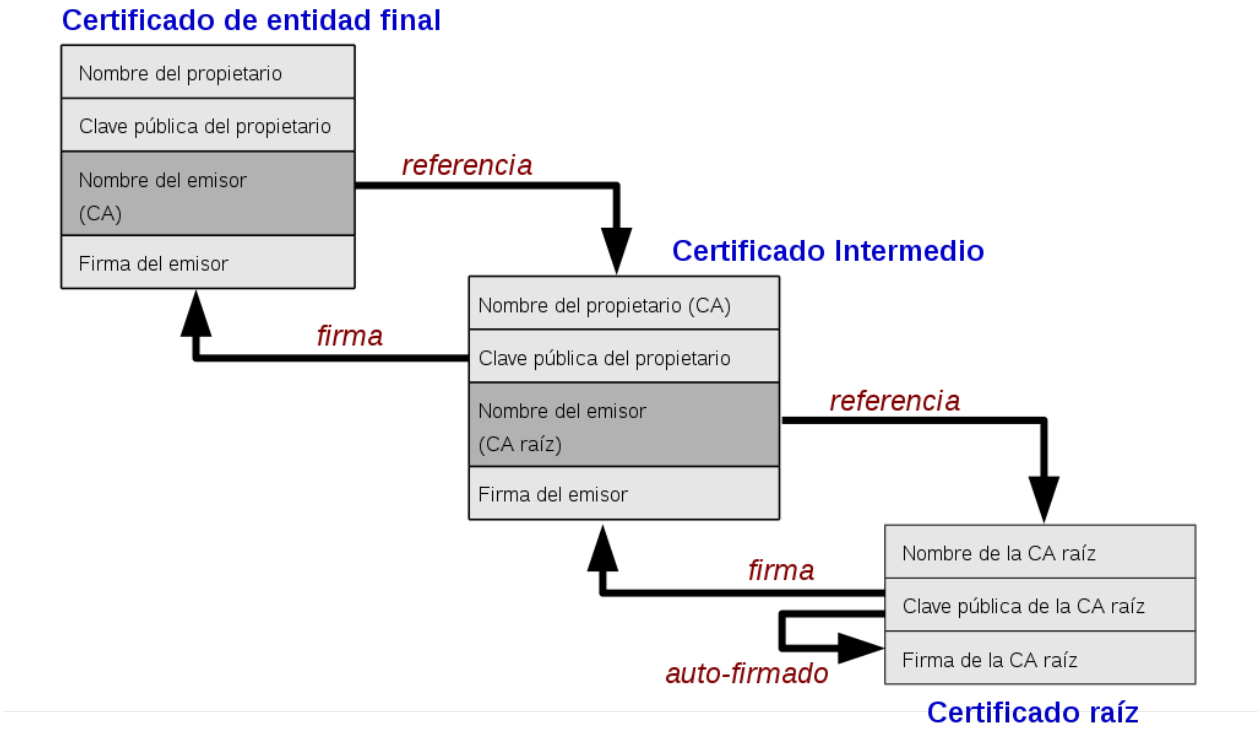

Figura 2

3.1. Tipos de jerarquías

3.1.1. Jerarquía de un solo nivel

En este tipo de jerarquía interviene una única CA que actúa como Autoridad raíz y como emisora. Este tipo no está recomendado para ningún escenario de producción, pues si esta única CA se ve comprometida se pone en riesgo toda la PKI (infraestructura de clave pública).

Figura 3

3.1.2. Jerarquía de dos niveles

Este diseño consta de una CA raíz que se encuentra offline y una CA emisora subordinada que está online. Aquí el nivel de seguridad es mayor porque la CA raíz y la CA emisora están separadas y, lo que es más importante, la raíz se encuentra offline, de forma que la clave privada de esta está más protegida contra cualquier tipo de ataque. Además, este tipo de jerarquía proporciona una mayor escalabilidad y flexibilidad puesto que puede haber más de una CA emisora subordinada a la CA raíz. Por lo tanto, pueden exister varias CAS en diferentes localizaciones geográficas, así como con diferentes niveles de seguridad.

La jerarquía de dos niveles es la más común y satisface las necesidades de la mayoría de las empresas. Es el tipo de jerarquía que venimos considerando hasta ahora.

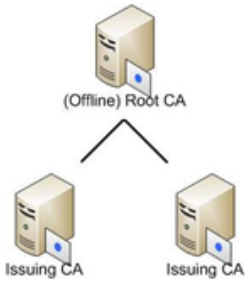

Figura 4

3.1.3. Jerarquía de tres niveles

La diferencia entre la jerarquía de dos niveles y la de tres niveles, es que esta última introduce un segundo nivel entre la CA raíz y la CA emisora. Hay dos motivos principales por los que se realiza esto:

- La primera razón sería para usar esta CA de segundo nivel como una *Policy CA*, la cual está configurada para emitir certificados a la CA emisora y está restringida en el tipo de certificados que puede emitir. Por ejemplo, una Policy CA emite certificados que requieren que el usuario aparezca en persona y otra los emite a cualquier persona jurídica autenticada.
- La segunda razón es que si se necesita revocar un determinado número de certificados debido a que una llave se vio comprometida, la revocación se puede llevar a cabo en

este segundo nivel introducido, dejando otras ramas desde la raíz disponibles (las otras CAs del segundo nivel que no están en riesgo).

Cabe destacar que las CAs del segundo nivel pueden mantenerse offline como la CA raíz, de manera que aumenta así la seguridad de las mismas.

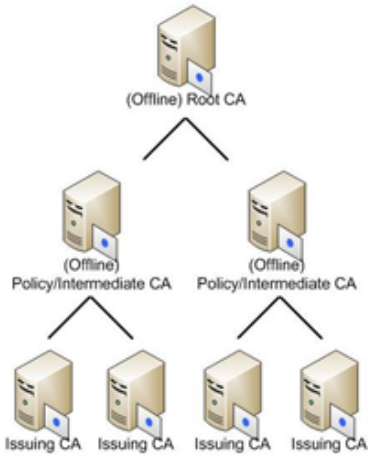

Figura 5

A medida que se incluyen más niveles intermedios, incrementa la seguridad, flexibilidad y escalabilidad (por los motivos ya explicados), pero empeora el rendimiento en la creación de la cadena de certificados, pues hay que verificar el certificado y la información del estado del certificado tanto para las CAs emisoras como para las Policy CAs. Debido a esto, la jerarquía de tres niveles generalmente no está recomendada, salvo para casos concretos en los que se necesite este segundo nivel por alguna de las razones ya comentadas.

3.2. Confianza en una CA

Una de las formas por las que se establece la confianza en una CA para un usuario consiste en la 'instalación' en el ordenador del usuario (tercero que confía) del certificado autofirmado de la CA raíz de la jerarquía en la que se desea confiar. El proceso hay que repetirlo por cada uno de los navegadores que existan en el sistema. Así, un navegador web o dispositivo establece su confianza en una CA al aceptar su correspondiente certificado raíz en su almacén de CAs de confianza, una base de datos de CAs, muchas de las cuales viene preinstaladas en el navegador o dispositivo.

Si está instalada una CA en el repositorio de CAs de confianza de cada navegador, cualquier certificado firmado por dicha CA se podrá validar, ya que se dispone de la clave pública con la que verificar la firma que lleva el certificado. Cuando el modelo de CA incluye una jerarquía, es preciso establecer explícitamente la confianza en los certificados de todas las

cadenas de certificación en las que se confíe. Para ello, se puede localizar sus certificados mediante distintos medios de publicación en internet, pero también es posible que un certificado contenga toda la cadena de certificación necesaria para ser instalado con confianza.

Hay un número relativamente pequeño de CAs autorizadas, desde compañías privadas hasta gobiernos, y, por lo general, cuanto más tiempo ha estado la CA operativa, más probable es que los navegadores y dispositivos tengan sus certificados preinstalados. El número de navegadores web, dispositivos móviles y aplicaciones que confían en un certificado concreto se conoce como ubicuidad y es una de las características más importantes que una CA puede ofrecer a sus clientes.

3.3. El riesgo de las CA

Las autoridades de certificación no deben emitir certificados digitales directamente desde la raíz, sino a través de sus CAs intermedias, como ya se ha comentado. El principal motivo de tomar esta decisión es proteger al máximo el Certificado Raíz de la jerarquía, minimizando lo máximo posible la exposición de este a los atacantes. Si un tercero consiguiera la clave privada de un Certificado Raíz, se comprometerían todos y cada uno de los certificados emitidos con la misma e incluso sería capaz de emitir certificados en nombre de la CA raíz. Este hecho propicia que se considere normalmente la clave privada del certificado raíz como la información de mayor valor y más crítica de la CA. Debido a ello, es habitual que la clave privada del certificado raíz se encuentre offline, en un dispositivo de seguridad cifrado y con medidas de alta seguridad.

Un **caso destacable de subversión de una CA** tuvo lugar en 2001, cuando la CA VeriSign emitió dos certificados a una persona que afirmaba representar a Microsoft. Los certificados tenían el nombre de *Microsoft Corporation*, con lo que fueron usados para engañar a la gente haciéndole creer que actualizacions del software de Microsoft provenían de Microsoft cuando en realidad no era así. El fraude fue detectado a principios de 2001 y ambos, Microsoft y Verisign, tomaron medidas para intentar limitar el impacto del problema.

4. Certificados TLS/SSL

Para encriptar y verificar la comunicación entre un cliente y un servidor, el protocolo TLS (así como su antecesor, SSL) emplea certificados de claves públicas, generalmente en formato X.509. El uso más notable de TLS es en el protocolo HTTPS, que habilita la transmisión segura de datos en páginas y aplicaciones web.

A la hora de establecer una conexión mediante TLS o SSL, el servidor debe presentar un certificado firmado por una autoridad de certificación; el cliente, opcionalmente, también puede presentar uno como método de autenticación.

4.1. En servidores

Para verificar que el certificado del servidor al que se conecta es válido, el cliente comprueba que su nombre de dominio coincide con el que aparece como nombre común en el campo Sujeto del certificado, y que éste ha sido firmado por una CA de confianza.

En el caso de querer utilizar un mismo certificado con múltiples dominios, se pueden listar en el campo Sujeto, o hacer uso de la extensión *Subject Alternative Name* al estándar X.509, que añade un campo de nombres alternativos en el que se listan todos los dominios al que se aplica el certificado. Estos certificados se suelen llamar UCC (*Unified Communications Certificates*), o SAN, por las iniciales de la extensión en inglés. Por motivos de retrocompatibilidad, ciertas CA listan los dominios mediante ambos métodos en un mismo certificado.

Donde es de utilidad, un servidor TLS puede usar un certificado firmado por sí mismo, por ejemplo en entornos de prueba o en redes locales, pero sin desactivar el comprobado de certificados en el cliente, éste no podrá verificar el certificado y la conexión fallará.

4.2. En clientes

El uso de certificados por el lado del cliente es menos común, y al contrario que los certificados del servidor, los del cliente suelen contener el nombre del usuario o su correo electrónico en el campo Sujeto, en lugar de un nombre de dominio. Debido a su uso para autenticar usuarios, normalmente se generan en una CA interna, propia del servicio al que se quiere acceder, en lugar de una CA pública.

Un uso más común de estos certificados es en sistemas de llamada a procedimiento remoto (RPC), para autenticar dispositivos y comprobar si están autorizados o no para realizar ciertas llamadas.

5. Formato X.509

X.509 es un estándar del Sector de Normalización de las Telecomunicaciones de la UIT (UIT-T), publicado junto con el estándar X.500, para definir el formato de los certificados de claves públicas, tanto en entornos *online* como *offline*, además de las listas de revocación de certificados y el algoritmo empleado para validar la ruta de certificación.

Este formato opera asumiendo una jerarquía estricta de autoridades de certificación que emiten los certificados, al contrario que PGP (pretty good privacy), donde cualquiera puede firmar y avalar la validez de otros certificados. Es posible utilizar X.509 de esa manera, pero es muy poco común.

5.1. Características de los certificados

La estructura de los certificados se establece mediante un lenguaje de descripción de interfaz diseñado también por el UIT-T, *Abstract Syntax Notation One* (ASN.1). Esta estructura, en la versión 3 del estándar, es la siguiente:

- **Certificado:**
  - **Versión**
  - **Número de serie;** número entero positivo de 64 bits como mínimo, debe ser único para cada certificado de una misma CA.
  - **ID del algoritmo con el que ha firmado la CA;** suele ser RSA o DSA
  - **Emisor;** nombre de la CA que emite el certificado, en notación *Distinguished Name* (DN), de acuerdo con el estándar X.500.
  - **Validez;** para que el certificado se considere válido, la fecha en la que se evalúa debe caer entre las fechas mínima y máxima.
    - No antes de
    - No después de
  - **Sujeto;** nombre del usuario, dispositivo o servicio al que se emite el certificado, en notación DN.
  - **Información de la clave pública del sujeto**
    - Algoritmo de la clave
    - Valor de la clave
  - (opcional) **ID único del emisor**
  - (opcional) **ID único del sujeto**
  - (opcional) **Extensiones;** añadidas en la versión 3, cada una debe tener un ID propio e indicar si es crítica o no. Si el sistema no puede reconocer una extensión crítica, el certificado entero es rechazado; si no es crítica, simplemente se ignora.
    - ...
- **Firma de la CA**

5.2. Codificación

La sintaxis ASN.1 define el contenido de los certificados y su estructura; existen varios sistemas para codificarlos y almacenarlos en archivos compatibles con la plataforma en la que se va a utilizar el certificado. Generalmente, están basados en las codificaciones BER (*Basic Encoding Rules*) y DER (*Distinguished Encoding Rules*), definidas en el estándar X.690 del UIT-T, que definen cómo convertir información en sintaxis ASN.1 a un formato binario.

Existen múltiples extensiones de archivo de certificados, pero las más usadas son las siguientes:

- **.pem** – de las iniciales de *Privacy-enhanced Electronic Mail*, almacena los datos DER en texto plano, codificados en base 64, entre una cabecera y un pie de página que indican que el archivo es un certificado.
- **.cer, .crt, .der** – contienen los datos DER en formato binario.
- **.p7b, .p7c** – de PKCS#7, un estándar de RSA para firmar o cifrar mensajes, así como distribuir certificados, y que forma la base de S/MIME.
- **.p12** – de PKCS#12, otro estándar de RSA, éste generalmente usado para almacenar el certificado junto a su clave privada en ficheros protegidos por clave simétrica.

6. Un poco de historia

En 1995, Mark Shuttleworth, más conocido actualmente como director ejecutivo de Canonical, fundó la empresa Thawte, la primera autoridad de certificación en emitir certificados públicos fuera de los Estados Unidos. Ese mismo año se fundó Verisign, a partir de la rama de servicios de certificación de RSA; ambas entidades acabaron ocupando el 50 % del mercado cada una, y sus certificados fueron incluidos en las primeras versiones de Netscape.

Verisign acabó adquiriendo Thawte en 1999 por 575 millones de dólares, por un lado convirtiendo a Shuttleworth en el segundo turista espacial de la historia, y por otro dando inicio a la hegemonía de Verisign en la industria.

Sin embargo, en 2001, un incidente con un atacante haciéndose pasar por un empleado de Microsoft consiguió que Verisign le emitiera dos certificados que le permitirían suplantar la identidad de la compañía, para disgusto de Microsoft, que aún no había implementado CRLs en Windows. El año siguiente, GeoTrust se convirtió en la primera CA en emitir certificados validados por dominio, con un precio más barato y un proceso más rápido que llevó a la empresa a ocupar el 26,7 % del mercado en 2006.

Hacia finales de los 2000, Melih Abdulhayoğlu, de Comodo Security Solutions, dio comienzo al *CA/Browser Forum*, un consorcio de CAs y desarrolladores de navegadores web y sistemas operativos, con el objetivo de establecer estándares para la emisión y el manejo de certificados públicos, llevando a la formación del *CA Security Council* en 2013.

Volviendo atrás un par de años, en 2011 un atacante obtuvo acceso administrativo a los sistemas de una CA holandesa, DigiNotar, poniendo a más de 300.000 usuarios de Gmail en Irán en riesgo de un ataque *man-in-the-middle* y provocando la bancarrota de la CA.

Después de este incidente, tras haber adquirido la división de autenticación y certificación de Verisign por 1280 millones de dólares, fue Symantec, una compañía mucho mayor, la que

hizo sonar las alarmas de la industria al descubrirse en 2015 que había estado emitiendo certificados para 76 dominios distintos, sin la autorización de sus propietarios. Esto llevó a la revocación de sus certificados en la mayoría de navegadores, y a la venta por 950 millones de dólares de su rama de certificación a DigiCert, que en octubre de 2018 consiguió reemplazar más de 5 millones de certificados afectados.

Volviendo a 2015, este es el año en el que IdenTrust, una CA establecida en 1999 con una buena reputación entre instituciones financieras, agencias gubernamentales y grandes empresas, firmó los certificados intermedios de Let's Encrypt, una CA de código abierto y sin ánimo de lucro, dando comienzo a su rápida expansión hasta convertirse, hoy en día, en la mayor autoridad de certificación del mundo.

7. Práctica: Servidor HTTPS

Enlace al video.

En la demostración práctica vamos a generar un certificado SSL para crear una página web que use HTTPS. Para esto primeramente necesitamos algunas herramientas para crear un servidor web: usaremos NodeJS y ExpressJS. Además necestamos un dominio, luego usaremos `noip.com` ya que nos permite crear dominios gratis.

Como veremos, para probar nuestra identidad a certbot, necesitaremos que el servidor empiece en el puerto 80, por lo que tendremos que usar un servidor externo en lugar de nuestro ordenador. Para ello, usaremos un droplet en Digital Ocean con NodeJS ya instalado para mayor comodidad. Una vez creado el droplet simplemente tenemos que configurar una conexión SSH, conectarnos e instalar las siguientes dependencias:

```
npm init
npm install express
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install certbot
```

Después generamos un certificado SSL usando *certbot*: `certbot certonly --manual`. Cuando escribamos nuestro dominio, *certbot* nos pedirá que creemos una estructura de directorios específica: *http:dominio/.well-known/acme-challenge/archivo*. Ese *archivo* tendrá que contener un *challenge* determinado, proporcionado por *certbot*. Antes de continuar, necesitamos crear esos directorios, ese archivo, y arrancar el servidor para que esté disponible. Para crear el servidor web simplemente tenemos que crear un archivo *server.js* con el siguiente contenido:

```
const express = require('express');
const app = express();
```

```
app.use(express.static(__dirname, { dotfiles: 'allow' } ));

app.listen(80, () => {
  console.log('HTTP server running on port 80');
});
```

El objetivo de la tercera línea es servir los archivos estáticos empezando desde *__dirname*, es decir, desde el directorio en el que el servidor se está ejecutando. Además, los archivos cuyo nombre empiezan por punto, es decir, los dotfiles, ya que están deshabilitados por defecto, luego es necesario activarlos. Una vez el archivo se ha creado, nos conectamos desde otra terminal y ejecutamos el servidor con la orden `node server.js`. Ahora ya sí podemos darle a continuar y deberíamos ver el siguiente mensaje:

```
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/trabajospsi.ddns.net/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/trabajospsi.ddns.net/privkey.pem
   Your cert will expire on 2021-03-19. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot
   again. To non-interactively renew *all* of your certificates, run
   "certbot renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Como vemos, se han creado varios archivos: el certificado (*cert.pem*), las cadenas de certificación (*fullchain.pem* y *chain.pem*) y la llave privada (*privkey.pem*). Esos archivos tendremos que usarlos en nuestro servidor para poder usar HTTPs. En 1, se puede ver cómo cambia el código para usar esos archivos. Simplemente hay que leerles y pasárselos a ExpressJS. Con esto ya tenemos un servicio web usando HTTPs!

Listing 1: server.js

```
const fs = require('fs');
const http = require('http');
const https = require('https');
const express = require('express');

const app = express();

const privateKey = fs.readFileSync(
        '/etc/letsencrypt/live/trabajospsi.ddns.net//privkey.pem', 'utf8');
const certificate = fs.readFileSync(
        '/etc/letsencrypt/live/trabajospsi.ddns.net//cert.pem', 'utf8');
const ca = fs.readFileSync(
        '/etc/letsencrypt/live/trabajospsi.ddns.net/chain.pem', 'utf8');

const credentials = {
        key: privateKey,
        cert: certificate,
        ca: ca
};

app.use((req, res) => {
        res.send('Trabajo SPSI :D!');
});

const httpServer = http.createServer(app);
const httpsServer = https.createServer(credentials, app);

httpServer.listen(80, () => {
        console.log('HTTP Server running on port 80');
});

httpsServer.listen(443, () => {
        console.log('HTTPS Server running on port 443');
});
```

Referencias

[1] https://es.wikipedia.org/wiki/Autoridad_de_certificaci%C3%B3n

[2] https://en.wikipedia.org/wiki/Certificate_authority

[3] https://es.wikipedia.org/wiki/Certificado_de_clave_p%C3%BAblica

[4] https://en.wikipedia.org/wiki/Public_key_certificate

[5] https://www.globalsign.com/es/ssl-information-center/what-are-certification-authorities-trust-hierarchies

[6] https://www.ssl.com/es/faqs/%C2%BFQu%C3%A9-es-una-autoridad-de-certificaci%C3%B3n%3F/

[7] https://www.encryptionconsulting.com/certificate-authority-and-hierarchy/

[8] https://es.wikipedia.org/wiki/Cadena_de_confianza

[9] https://www.securityartwork.es/2013/06/12/fundamentos-sobre-certificados-digitales

[10] https://sites.google.com/site/ddmwsst/digital-certificates

[11] https://es.wikipedia.org/wiki/Online_Certificate_Status_Protocol

[12] https://es.wikipedia.org/wiki/Lista_de_revocaci%C3%B3n_de_certificados

[13] https://en.wikipedia.org/wiki/X.509

[14] https://en.wikipedia.org/wiki/X.690

[15] https://en.wikipedia.org/wiki/PKCS

[16] https://www.ssldragon.com/blog/the-history-of-certificate-authorities/