

Escuela Técnica Superior de Ingeniería Informática

Ingeniería Informática

TRABAJO FIN DE CARRERA

Autoridad de Certificación: WebCA

Autor/es:

Jesús Fernández Sáez

Tutor/es:

Pablo Neira Ayuso

Curso 2017/2018

Índice

[Índice de figuras ii](#_Toc508655438)

[Índice de tablas iii](#_Toc508655439)

[1. Introducción 1](#_Toc508655440)

[2. Solución propuesta 3](#_Toc508655441)

[3. Estado del arte 5](#_Toc508655442)

[3.1. Suites criptográficas 5](#_Toc508655443)

[3.2. Autoridades de certificación 5](#_Toc508655444)

[4. Criptografía de clave pública 7](#_Toc508655445)

[4.1. Glosario 7](#_Toc508655446)

[4.2. Generación de claves 8](#_Toc508655447)

[4.3. Estándares PKCS 10](#_Toc508655448)

[5. Infraestructuras de clave pública 12](#_Toc508655449)

[5.1. Estándar X.509 para certificados digitales 13](#_Toc508655450)

[5.2. Certificados digitales 13](#_Toc508655451)

[5.2.1. Estructura de un certificado 14](#_Toc508655452)

[5.2.2. Extensiones 15](#_Toc508655453)

[5.2.3. Formatos de archivo para certificados y PKI 16](#_Toc508655454)

[5.3. Cadenas de certificados 17](#_Toc508655455)

[5.3.1. Comprobación de la validez de un certificado 19](#_Toc508655456)

[5.3.2. Certificados autofirmados 20](#_Toc508655457)

[5.4. Validación cruzada de certificados 20](#_Toc508655458)

[5.5. Comprobación de la revocación de un certificado 21](#_Toc508655459)

[5.5.1. Listas de revocación de certificados – CRL 22](#_Toc508655460)

[5.5.2. Online Certificate Status Protocol – OCSP 23](#_Toc508655461)

[5.5.3. OCSP Stapling 24](#_Toc508655462)

[5.6. Autoridades de certificación 25](#_Toc508655463)

[5.6.1. Proceso de emisión de un certificado 26](#_Toc508655464)

[5.6.2. Tipos de CSR 27](#_Toc508655465)

[5.6.3. Distribución del certificado generado 28](#_Toc508655466)

[5.6.4. Seguridad relacionada con una CA 29](#_Toc508655467)

[5.7. Confianza en una PKI 29](#_Toc508655468)

[6. Planificación 31](#_Toc508655469)

[7. Referencias 32](#_Toc508655470)

# Índice de figuras

[Figura 1. Arquitectura de alto nivel de la solución propuesta. 4](#_Toc508655471)

[Figura 2. La generación de claves RSA usa dos números primos muy grandes y, después de aplicar un algoritmo matemático, genera una pareja de claves. 9](#_Toc508655472)

[Figura 3. Ejemplos de jerarquías PKI con un certificado intermedio (izquierda) y con dos certificados intermedios (derecha). 13](#_Toc508655473)

[Figura 4. Información del certificado digital de www.openssl.org. 14](#_Toc508655474)

[Figura 5. Ejemplo de extensión SAN en el certificado de www.itu.int. 16](#_Toc508655475)

[Figura 6. Cadena de certificados para www.itu.int. 18](#_Toc508655476)

[Figura 7. El certificado final está firmado por un certificado intermedio COMODO RSA Extended Validation. 18](#_Toc508655477)

[Figura 8. El certificado intermedio está firmado por COMODO RSA Certification Authority. 18](#_Toc508655478)

[Figura 9. COMODO RSA Certification Authority es un certificado raíz ya que es autofirmado. 19](#_Toc508655479)

[Figura 10. Ejemplo de validación de una cadena de certificados. 20](#_Toc508655480)

[Figura 11. Validación cruzada entre PKI 1 y PKI 2. 21](#_Toc508655481)

[Figura 12. CRL de Starfield Secure Certificate Authority - G2. Detalles y certificados revocados. 23](#_Toc508655482)

[Figura 13. Esquema parcial de la jerarquía de certificados CA de la Fábrica Nacional de Moneda y Timbre (FNMT). 26](#_Toc508655483)

# Índice de tablas

[Tabla 1. Implementaciones de Autoridades de Certificación disponibles. 6](#_Toc508655484)

# ¿Resumen?

# Introducción

Los certificados digitales son una parte fundamental de las comunicaciones seguras hoy en día. Casi cualquier servicio web ya usa el protocolo HTTPS para servir su contenido de una manera segura. Además, los certificados digitales tienen más aplicaciones que ayudar a hacer que un canal de comunicación sea seguro. La distribución o instalación de software habitualmente usa algún tipo de firma digital para que el usuario receptor de dicho software pueda confiar en que ha recibido una copia exacta a como el fabricante la haya publicado.

Así mismo, con la evolución de las sedes electrónicas de la Administración, todos los ciudadanos españoles disponen de su propio certificado digital en su DNI electrónico. En él se dispone de una serie de claves criptográficas y de varios certificados que permite al ciudadano identificarse ante la Administración para hacer diferentes trámites como el pago de multas de circulación, obtención del certificado de nuestra vida laboral o hacer la declaración de la Renta.

Con el desarrollo de las tecnologías en la nube y servicios híbridos, muchas empresas también están haciendo uso de certificados digitales para la autenticación de sus usuarios. Por ejemplo, una empresa puede proporcionar a sus empleados un usuario y contraseña, pero también puede introducir un segundo factor de autenticación para asegurarse de que es el empleado el que realmente el que está intentando autenticarse. Este segundo factor puede ser un certificado digital en modo de certificado integrado en una smartcard de empleado o una smartcard virtual en el dispositivo del empleado. En ambos casos se trata de un certificado de usuario firmado habitualmente por la Autoridad de Certificación de la propia organización.

Un certificado digital es solo la parte más visible de las infraestructuras de clave pública. Dentro de una infraestructura de clave pública o PKI por sus siglas en inglés (Public Key Infrastructure) se tienen más conceptos como: clave privada, clave pública, firma digital o Autoridad de Certificación (CA). Estos conceptos se explicarán más adelante.

Este proyecto consiste en la implementación de un software de Autoridad de Certificación que permite la implementación de una infraestructura de clave pública. Esta implementación de una CA usará los estándares habituales PKCS para la solicitud y emisión de los certificados, así como aquellos usados para comprobar la validez de un certificado. Esta implementación se centrará en la parte de usuario final y del administrador, es decir, en el front-end que permitirá realizar estas acciones de una manera sencilla y no se centrará en la implementación de todos los protocolos criptográficos usados. Para esto último se usará alguna de las suites criptográficas ya disponibles.

Existen varias implementaciones de suites criptográficas actualmente, algunas de código abierto y otras propietarias: OpenSSL, GnuTLS, NSS, CAPI2. Para este proyecto se ha elegido la librería OpenSSL ya que es una librería bien mantenida y usada frecuentemente en otras aplicaciones criptográficas por lo que asegura estar bajo revisión constantemente y soluciones relativamente rápidas para posibles vulnerabilidades que puedan surgir en el futuro. Además, existen adaptadores de esta librería para ser usada con una gran cantidad de lenguajes de programación.

El software que se va a desarrollar será una Autoridad de Certificación web, es decir, permitirá que un usuario pueda solicitar un certificado digital mediante una interfaz web, de manera que se facilite la solicitud, renovación o revocación de un certificado digital.

Los navegadores actuales soportan la creación de claves privadas directamente en el propio navegador por lo que la creación de una autoridad de certificación que solo necesite el uso de un navegador web sin ningún tipo de software adicional por parte de los usuarios simplifica enormemente estas tareas y está al alcance de cualquier usuario con un navegador que soporte estas operaciones. Por supuesto, se proporcionarán también funciones más avanzadas para la creación de certificados en el caso de que el usuario no disponga de un navegador que soporte estas funciones de clave privada.

Por otra parte, el uso de una aplicación web permite que la administración y configuración de un sistema sea más fácil al poderse usar desde cualquier tipo de dispositivo, ya sea fijo o móvil, además de ser independiente del sistema operativo del operador con lo que se imponen menos restricciones de este tipo. Además, es fácil escalar una aplicación web por lo que en caso de necesitar más capacidad para atender más peticiones de usuarios finales debería ser relativamente sencillo aumentar esta capacidad escalando horizontalmente, es decir, añadiendo más servidores web.

Finalmente, una parte importante de la comprobación de validez de los certificados digitales es poder contactar con la CA según una serie de protocolos como CRL y OCSP. CRL se simplifica cuando se usa un servidor web ya que estas listas se publican habitualmente en archivos descargables y es sencillo cuando ya se dispone de un servidor web. OCSP es un protocolo puramente HTTP.

# Solución propuesta

La solución que se propone es una aplicación web que permitirá a un usuario final realizar las siguientes acciones principales, entre otras:

* Solicitar un certificado digital. La CA debe realizar algún tipo de comprobación de los datos que el usuario proporciona, según el tipo de certificado digital que se solicite.
* Renovar un certificado digital. Dado que los certificados digitales tienen un periodo de validez, una vez que este expire, el usuario puede solicitar que el certificado se vuelva a emitir para un nuevo periodo de validez.
* Revocar un certificado. En determinadas circunstancias se podría necesitar que un certificado deje de ser válido por lo que se tiene que permitir que un usuario solicite dar de baja un certificado y la CA debe incluirlo dentro de sus listas de revocación de certificados como certificado no válido.

Un operador que administre la CA podrá hacer las siguientes acciones:

* Realizar tareas durante la creación de los certificados. Estas tareas pueden ser aprobar o denegar la emisión de un certificado o revocar un certificado unilateralmente.
* Administrar las plantillas de certificados que los usuarios pueden usar. Dado que los certificados tienen diferentes usos, tiene sentido la creación de plantillas que especifiquen los diferentes certificados que la CA puede emitir. Ejemplos de plantillas pueden ser: certificados de usuario para autenticación, firmado o cifrado de correos electrónicos, certificados para servidores (web, correo, etc) o certificados para firmado de software.

La arquitectura de la solución será:

* Una aplicación web para los usuarios. Esta será el punto de acceso público a la Autoridad de Certificación.
* Una aplicación web para los administradores. Al tener su propia aplicación web, se puede restringir el acceso a esa aplicación de una manera diferente a la pública, ya sea por restricción de IP, método de autenticación, etc.
* Una aplicación web para la publicación de listas de revocación de certificados (CRL), así como un respondedor OCSP para atender este tipo de peticiones de validación.
* Un proceso emisor (servicio en modo demonio) que se encargará de realizar las tareas de cifrado/firmado de certificados pendientes. Este servicio leerá de una cola de acciones pendientes y actualizará las acciones pendientes con el resultado. Al tener un proceso separado del servicio web, se puede restringir el acceso a la clave privada de la CA que es la parte más importante de toda la CA. Este servicio puede ejecutarse en las mismas máquinas que las aplicaciones web o no.
* Un sistema de almacenamiento para los usuarios finales de la CA. Se podría usar un directorio LDAP si esta se despliega en una organización con un directorio LDAP.
* Un sistema de almacenamiento para los usuarios de administración. Igual que en el caso anterior, también se podría usar un directorio LDAP para identificar estos usuarios.

La arquitectura de la aplicación está descrita en la siguiente figura.

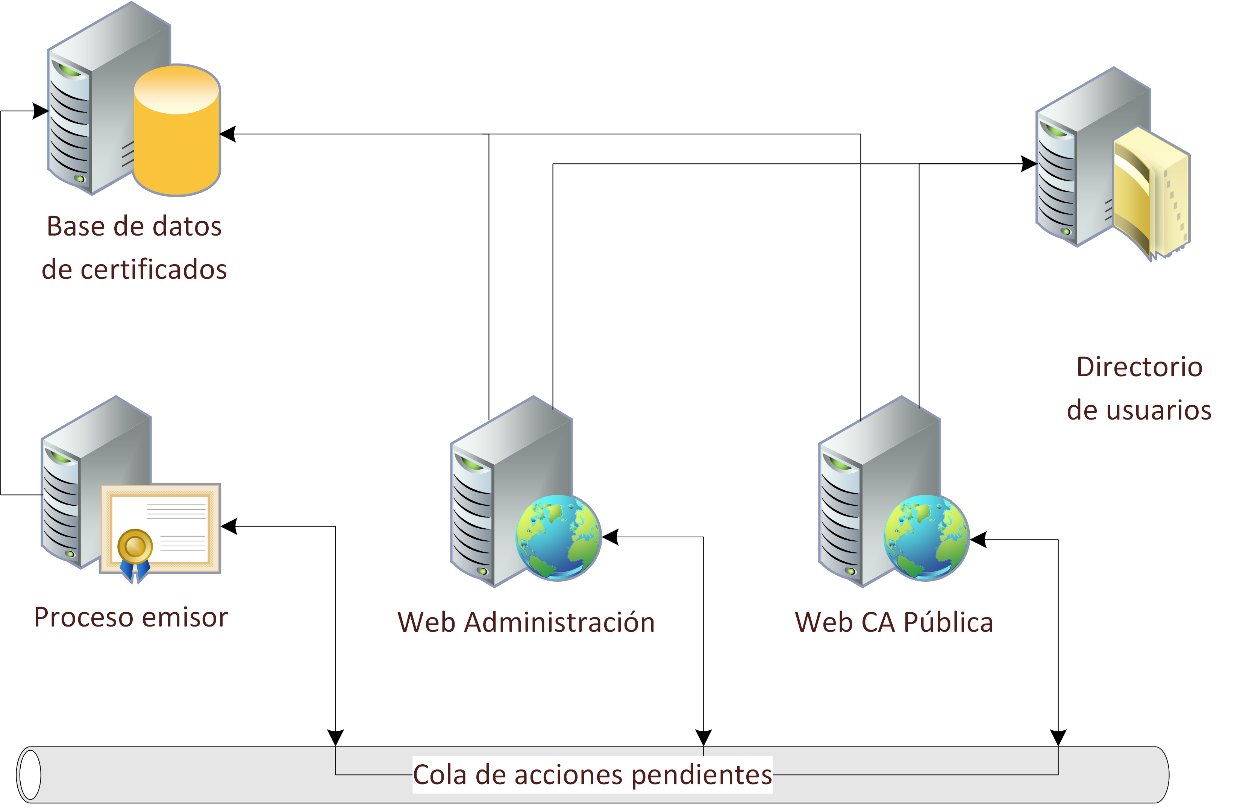


Figura 1. Arquitectura de alto nivel de la solución propuesta.

Se ha elegido Python como lenguaje de programación para el desarrollo de la parte de servidor de la aplicación. Python es un lenguaje de programación versátil y potente, pero también sencillo de leer y fácil de usar. Ha crecido mucho en popularidad en los últimos tiempos y, al ser un lenguaje derivado de C, es un lenguaje sencillo al que adaptarse.

Python dispone de uno de los framework de desarrollo web más conocidos: Django. Django dispone de una serie de características que lo hacen muy atractivo:

* MVC: sigue el principio Modelo-Vista-Controlador que permite separar la aplicación en estas tres conocidas capas.
* ORM: Object Relational Mapper, interactuar con los modelos de base de datos es extremadamente sencillo usando el ORM que incorpora Django. Esta capa de abstracción permite desarrollar la aplicación sin necesidad de hacer consultas a las bases de datos directamente.
* Migración de modelos: dado que una aplicación puede ir evolucionando con el tiempo, así mismo lo pueden hacer sus modelos (entidades que se guardan en base de datos). Django incorpora un sistema de migración y actualización de modelos que hace los cambios necesarios en los esquemas de las bases de datos.

# Estado del arte

A continuación, se muestran primero las diferentes implementaciones de librerías para trabajar con funciones criptográficas y certificados y posteriormente se muestran algunas implementaciones de software para crear una autoridad de certificación.

## Suites criptográficas

Todos los lenguajes de programación habituales tienen implementaciones para trabajar con certificados X.509, ya sea mediante implementaciones directas de o envolviendo otras librerías como OpenSSL.

OpenSSL es una implementación de código abierto de los protocolos SSL y TLS hecha en C. Muchas de las librerías disponibles para el resto de los lenguajes de programación se basan en llamar al código de OpenSSL para realizar operaciones. Además de ser una librería, OpenSSL incluye una serie de aplicaciones de consola para administrar claves, certificados, generar CSR e incluso cifrar datos. LibreSSL y BoringSSL son otros proyectos derivados de OpenSSL.

Otra librería libre es GnuTLS, que tiene una lista de funcionalidades similar.

Para la plataforma Windows y sus lenguajes de programación, Microsoft tiene su propia implementación de los protocolos y funciones de criptografía llamada CryptoAPI o CAPI, aunque es una librería propietaria.

## Autoridades de certificación

Existen varias implementaciones de código abierto. La mayoría implementan los servicios más habituales como emisión, revocación y administración de certificados digitales, es decir, X.509, CSR y CRL.

Debido a que OpenSSL es una de las librerías de código abierto más utilizadas, muchas de las implementaciones disponibles usan esta librería para las operaciones criptográficas.

En la Tabla 1 se comparan algunas de las implementaciones disponibles más notables. Como se podrá ver, la fecha de las últimas versiones estables de la gran mayoría no es realmente reciente. Todas las implementaciones escogidas son multiplataforma.

De las diferentes Autoridades de Certificación de código abierto que existen, sin duda, EJBCA es la CA que aparentemente está más usada. Por una parte, tiene una versión Community que cualquiera puede descargar y crear su propia CA. Existe también una versión comercial llamada Enterprise orientada a organizaciones donde ya se da soporte por parte de una empresa dedicada a seguridad informática. Es además un software que puede ser extendido por el usuario y existe diferente software complementario para que EJBCA pueda ampliar sus funcionalidades como el uso de tokens hardware o poder usarlo con smartcards.

XCA ha sido actualizada muy recientemente después de 3 años de inactividad. Además del uso básico para generar certificados digitales, se puede usar también con smartcards o tokens USB si se instala una librería compatible. Conviene decir que esta CA es una aplicación de escritorio y no tiene una versión web, por lo que su uso queda limitado al PC donde está instalada.

Igual que XCA, gnoMint es otra CA cuya interfaz es una aplicación de escritorio o una interfaz de consola, por lo que tampoco se puede interactuar con ella mediante una web. gnoMint soporta el únicamente la administración básica de una CA y la creación/revocación de certificados. Y de nuevo, al igual que XCA, gnoMint no publica CRLs o tiene un respondedor OCSP, aunque sí soporta la configuración de rutas CRL/OCSP en los certificados que genera. Para estas funcionalidades depende software externo.

OpenCA es un conjunto de proyectos relacionados con autoridades de certificación y en el que uno de estos proyectos es una CA. La arquitectura de esta CA es distribuida y soporta tanto una CA offline como una RA online (Registration Authority). Esta CA sí tiene una interfaz web y soporta CRLs y OCSP.

Finalmente, la única CA digna de mención cuyo lenguaje es Python es pyCA. Esta es una CA offline y cuyas funcionalidades están basadas en el uso de scripts de consola y tareas programadas. Se podría decir que el proyecto está muerto ya que la última versión disponible data de 2003.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Licencia | Lenguaje | Versión estable | Notas |
| EJBCA | LGPL v2.1 | Java /J2EE | Diciembre 2017 | Implementa OCSP, CMP,  XKMS, SCEP, EC y CVC  (pasaportes digitales)  Clusterizable  Tiene un modo, usar y tirar, donde nada queda registrado y es capaz de generar una gran cantidad de certificados muy rápidamente.  CC, PP EAL4+[[1]](#footnote-2) |
| gnoMint | GPL v3 | C | Marzo 2016 |  |
| OpenCA PKI | GPL | Perl | Agosto 2013 | Basada en OpenSSL.  OCSP, SCEP.  Distribuible.  Interfaz web. |
| r509 | Apache 2.0 | Ruby | Diciembre 2016 | Basada en OpenSSL.  OCSP, SPKI/SPKAC.  Interfaz web. |
| TinyCA | GPL | Perl | Marzo 2007 | Basada en OpenSSL. |
| XCA | BSD | C++ | Enero 2018 |  |
| pyCA | GPL | Python | Noviembre 2003 |  |
| newPKI | GPLv2 | C++ | Diciembre 2005 | Basada en OpenSSL |

Tabla 1. Implementaciones de Autoridades de Certificación disponibles.

# Criptografía de clave pública

La criptografía de clave pública también se conoce como criptografía asimétrica [1]. La criptografía simétrica usa una única clave conocida y compartida por las dos partes de una comunicación. En cambio, la criptografía asimétrica usa algoritmos criptográficos que requieren dos claves para su funcionamiento, una para cifrar y otra descifrar. En este tipo de algoritmos, una de las claves es de dominio público, llamada clave pública, mientras que su pareja, la clave privada, se debe mantener, como su nombre indica, de manera privada. La clave pública se usa para cifrar o comprobar la validez de una firma digital, mientras que la clave privada se usa para descifrar o para generar una firma digital.

La criptografía de clave pública tiene dos usos habituales:

* Confidencialidad: solo el receptor del mensaje puede ver el contenido de este. Como se explica en el párrafo anterior, un mensaje se cifra con la clave pública y solo el poseedor de la clave privada puede descifrar dicho mensaje.
* Autenticación: el receptor sabe a ciencia cierta que el emisor del mensaje es quien dice ser. Es decir, existe un mecanismo que permite al receptor saber quién envió el mensaje. En el caso de certificados digitales se trata de una firma digital. La firma se crea con la clave privada, de manera que cualquiera que posea la clave pública puede confirmar que la firma es correcta. Además, la firma depende del contenido del mensaje, por lo que validar que la firma es correcta asegura que el mensaje solo puede haber sido firmado por quien tiene la clave privada y que el contenido del mensaje no ha sido modificado.

## Glosario

Para entender el funcionamiento de una PKI se necesita conocer el significado de algunos conceptos básicos, aunque alguno se explicará con más detalle en posteriores epígrafes:

* Clave privada

Clave que se mantiene en secreto y que se usa para descifrar un mensaje o para crear una firma digital.

* Clave pública

Clave conocida públicamente que se usa para cifrar un mensaje o para comprobar la validez de una firma digital.

* Pareja de claves

Una pareja de claves está formada por una clave pública y por una clave privada. Estas se generan habitualmente en el mismo momento según un determinado algoritmo. Dan lugar a lo que se llama criptografía asimétrica: con una clave se cifra y solo con la otra se puede descifrar y viceversa.

* Función hash

Función hash o función resumen. Es un algoritmo al que se le pasa como entrada la representación de un objeto de longitud indeterminada y habitualmente devuelve una cadena de caracteres con una longitud determinada que se puede usar como resumen de dicho objeto. Cualquier mínima modificación de la cadena original y, por tanto, del objeto original, debe hacer que la función hash tenga una salida diferente. Son funciones de un único sentido, es decir, una vez computado el resumen de un objeto, debe ser muy difícil conseguir el objeto original.

* Firma digital

Mecanismo en el que se usa la clave privada para cifrar el hash de un objeto. Con la firma se proporciona el hash original y el hash cifrado (firmado) de manera que cualquier usuario dispone de la clave pública para poder descifrar el hash firmado y comprobar que es el mismo que el original. Dado que solo el firmante tiene la clave privada, se puede comprobar así la autenticidad de la firma.

* Certificado digital

Documento digital que incluye una clave pública, una serie de datos sobre el propietario de la clave pública, un periodo de validez y la firma digital de dicho certificado realizada por una determinada entidad. Esta firma digital asegura que una Autoridad de Certificación da los datos que presenta el certificado digital como válidos.

Un certificado también puede ser autofirmado, es decir, que está firmado por la propia clave privada correspondiente a la clave pública del certificado.

* Autoridad de Certificación

El cometido principal de una Autoridad de Certificación o CA es firmar un certificado digital después de comprobar la validez de los datos del solicitante. Los usuarios de ese certificado digital, al confiar esta CA darán por válido los certificados que esta CA firme.

Una CA tiene su propia pareja de claves y certificado firmado por otra CA diferente. Si el certificado de una CA es autofirmado, entonces se dice que es una Autoridad de Certificación Raíz.

* Cadena de confianza

Simplificando los conceptos, un certificado digital está firmado por el certificado de una CA. El certificado de esta CA está firmado por otra CA superior. Esta cadena de firmas digitales forma una cadena de confianza. Las CA que están arriba del todo en la jerarquía son las CA que tienen certificados autofirmados, es decir, son las CA Raíz. Los usuarios habitualmente confían en las CA Raíz de manera que, al confiar en la que está arriba del todo, un usuario puede confiar en todos los certificados intermedios o finales cuya cadena de confianza empiece por ese certificado raíz.

## Generación de claves

Existen varios algoritmos para generar parejas de claves. Estos algoritmos se basan en problemas matemáticos que tienen una solución costosa de obtener en un tiempo razonable. Debe ser fácil generar la pareja de claves, pero debe ser muy costoso computacionalmente deducir la clave privada a partir de la clave pública ya que esta va a ser distribuida y no es secreta.

Es necesario que las claves sean generadas con la mayor aleatoriedad y entropía posible. En el RFC 4086[[2]](#footnote-3) describe una serie de requerimientos necesarios de garantizar un nivel de aleatoriedad suficiente. En la práctica, las claves son números primos de muchos dígitos. Estos se podrían generar de diferentes maneras:

* Usando un generador de números aleatorios o pseudoaleatorios.
* Con una *frase de paso* a la que se le aplica una función de *hash*.

Las implementaciones de criptografía asimétrica más usadas actualmente se pueden encontrar en la siguiente lista. Existen otros métodos para generar una pareja de claves, pero los descriptos en la lista son los que se usan con más frecuencia. Cada protocolo usa un algoritmo propio para generar una pareja de claves.

* RSA. Se basa en la dificultad de factorizar el producto de dos números primos grandes. Se hizo público por primera vez en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman y de ahí su nombre. Las recomendaciones para su implementación están recogidas en PKCS #1[[3]](#footnote-4).

Una pareja de claves RSA se puede usar tanto para cifrar y como para crear firmas digitales.

Para RSA, las claves actuales suelen tener un tamaño entre 1024 bits y 4096 bits. Menos de esto se considera inseguro. Dado que una clave de 768 bits fue rota en 2009[[4]](#footnote-5), en la actualidad se recomienda usar claves de 2048 bits o mayores ya que debido a los avances actuales no está claro si algunas agencias de inteligencia son ya capaces de romper una clave de 1024 bits.

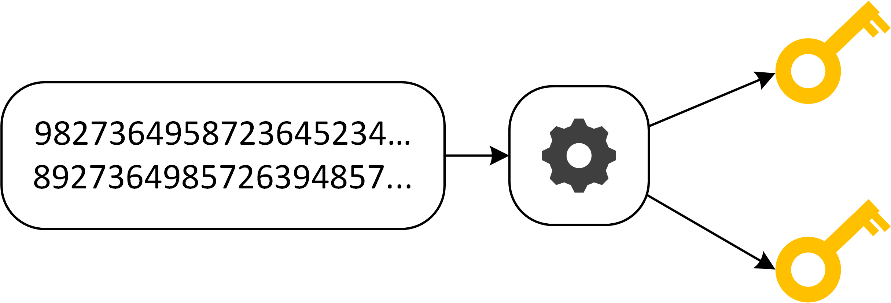


Figura 2. La generación de claves RSA usa dos números primos muy grandes y, después de aplicar un algoritmo matemático, genera una pareja de claves.

* ElGamal. Otro sistema de cifrado asimétrico que se basa en un problema matemático diferente al sistema RSA. En este caso, la seguridad se basa en la dificultad de computar logaritmos discretos. ElGamal usa un intercambio de claves Diffie-Hellman. Publicado por Taher ElGamal en 1985[[5]](#footnote-6). Existe otro algoritmo llamado ElGamal del mismo autor, pero usado para firmas digitales y muy poco usado en la práctica, pero es en el que se basa el protocolo DSA descrito en el punto siguiente.

Es uno de los algoritmos que se puede usar en GnuPG[[6]](#footnote-7), una implementación GNU de PGP, del que hablaremos más adelante.

* DSA, Digital Signature Algorithm. Es una variante del sistema de firma ElGamal. Este sistema se basa en la dificultad de computar logaritmos discretos. Desarrollado por la NSA y publicado por primera vez en 1991, fue patentado por el NIST (National Institute of Standards and Technology de EE. UU.) y adoptado como FIPS (Federal Information Processing Standards) 186 en 1993[[7]](#footnote-8).

FIPS-186 especifica que también se puede usar una pareja de claves RSA para firmas digitales con la restricción de que esta pareja de claves solo se use con el propósito de firma digital.

Al igual que en el caso de claves RSA para cifrado, se recomiendan claves de 2048 bits o mayores.

Como se ha visto, las claves tienen un tamaño que se expresa en bits. A mayor número de bits, la clave será más grande y, por consiguiente, más difícil de romper. La criptografía asimétrica necesita tamaños de clave mayores que la criptografía simétrica para ofrecer el mismo nivel de seguridad. Por ejemplo, NIST se estima que una clave RSA de 15360 bits es comparable a una clave simétrica AES de 256 bits[[8]](#footnote-9). Existen algoritmos asimétricos basados en curvas elípticas (ECC o Elliptic Curve Cryptography) que parecen ser seguros con claves más cortas[[9]](#footnote-10). Por ejemplo, una clave ECC de 224 bits se puede comparar con una clave simétrica de 112 bits, también según el NIST.

## Estándares PKCS

PKCS son las siglas de Public Key Cryptography Standard. En PKCS se recogen una serie de estándares relacionados con la criptografía de clave pública. RSA Security Inc. inició la publicación de estos estándares para promover el uso de los algoritmos criptográficos patentados por la compañía. Algunos de estos estándares se usan actualmente, aunque algunos hayan sido abandonados o sustituidos por otro. En la siguiente lista se describen brevemente aquellos estándares PKCS aún en uso [2]:

* PKCS #1, RSA Cryptography Standard, comentado anteriormente y donde se define la criptografía RSA. Las claves privadas se escriben usando la notación ASN.1 [3]. Publicado en el RFC 8017.
* PKCS #3, Diffie-Hellman Key Agreement Standard, donde se define el intercambio de claves Diffie-Hellman, un protocolo con el que dos interlocutores que no se conocen pueden conseguir establecer una clave secreta compartida en un canal de comunicación no seguro.
* PKCS #5, Password-based Encryption Standard, donde se define un estándar de cifrado basado en contraseñas. Publicado en el RFC 8018.
* PKCS #7, Cryptographic Message Syntax Standard. Precursor del S/MIME y usado para cifrar o firmar mensajes, así como para la distribución de certificados en una PKI. Publicado en el RFC 2315.
* PKCS #8, Private-Key Information Syntax Standard, donde se define un estándar usado para transportar parejas de claves, cifradas o sin cifrar. Publicado en el RFC 5958.
* PKCS #9, Selected Attribute Types, donde se definen tipos de atributos para usar en certificados PKCS #6, mensajes firmados PKCS #7, claves privadas PKCS #8 y solicitudes de firma de certificados PKCS #10. Publicado en el RFC 2985.
* PKCS #10, Certification Request Standard. Define el formato de los mensajes que se deben enviar a una autoridad de certificación para solicitar un certificado. Publicado en RFC 2986.
* PKCS #11, Cryptographic Token Interface. También conocido como "Cryptoki", es una API que define métodos genéricos para interactuar con un dispositivo token criptográfico, que son módulos criptográficos en hardware como, por ejemplo, una smart card (DNIe [4]).
* PKCS #12, Personal Information Exchange Syntax Standard, define un formato de archivo usado para almacenar claves privadas y sus certificados, protegidos por una clave simétrica. Es una evolución de PFX. Publicado en RFC 7292.
* PKCS #15, Cryptographic Token Information Format Standard, que define un mecanismo para que un usuario pueda usar un token criptográfico ante aplicaciones que sigan estos estándares, independientemente del proveedor o interfaz de este token.

# Infraestructuras de clave pública

Ahora que se sabe en qué se basa la criptografía de clave pública: se debe tener una pareja de claves, una privada y una pública. Si es necesario que una persona envíe un documento cifrado para que solo otra pueda leerlo, la persona que lo envía debe la clave pública de la que lo recibe para cifrar dicho documento. De esa manera, solo el que lo recibe con su clave privada puede descifrar ese mensaje que le llega.

Por otra parte, si una persona quiere enviar un documento asegurando a la que lo recibe que ha sido ella quien lo ha enviado y que nadie durante la transmisión ha modificado ese documento, el emisor firma digitalmente el mensaje con su clave privada y envía la firma junto al documento. De esa manera, cuando la otra persona reciba el documento y la firma, podrá comprobar con la clave pública del emisor la validez de la firma digital y podrá confiar en que el documento que le ha llegado es idéntico al que el otro había enviado.

Pero ¿cómo es posible estar seguro de que la clave pública que se recibe es efectivamente de la persona a la que se quiere enviar un documento secreto? Evidentemente una manera sería ir físicamente donde esté esa persona y que él mismo nos dé su clave pública. En la práctica, esta no es una solución viable ya que la persona no tiene por qué estar en la misma ciudad que nosotros, o ni siquiera en el mismo país.

Para solucionar este problema se introduce el concepto de confianza. ¿Qué permite a un usuario confiar en que la clave pública que tiene corresponde con la persona con la que se está comunicando o servicio al que quiere acceder? Hay dos respuestas a esta pregunta:

* Que las claves públicas que vayamos a usar estén firmadas por un ente u organismo central en el que el usuario confíe.
* Que las claves estén firmadas por otro individuo (o un grupo de individuos) en el que el usuario confíe.

Estas dos respuestas dan lugar a los dos grandes modelos de infraestructuras de clave pública que se usan en la actualidad: un modelo PKI centralizado o un modelo descentralizado, también conocido como *Web of trust* o anillo de confianza.

En el primer caso de una PKI centralizada, existen una o varias autoridades organizadas en una jerarquía que firman las claves públicas de los usuarios. Si se confía en la autoridad de nivel superior, por defecto, se confiará en todos los certificados que sean firmados por ella u otras autoridades que estén por debajo de la de nivel superior. Esto permite crear una relación de confianza con un gran número de certificados con tan solo confiar en una única entidad.

En el caso del anillo de confianza, esta autoridad de nivel superior no existe en un principio. Cada usuario tiene su pareja de claves y un anillo de claves públicas en las que confía. La confianza se crea cuando un usuario firma la clave pública de otro usuario. Así, se podría confiar en que una clave pública determinada corresponde con seguridad a su propietario sabiendo cuántas personas o entidades han firmado dicha clave.

Un usuario normal usará la mayor parte de su vida una infraestructura centralizada, sabiéndolo o no, ya que es la que se usa mayoritariamente para la navegación web. Para que un usuario normal no tenga que pasar por el proceso de confiar en las autoridades de certificación, actualmente los diferentes sistemas operativos y navegadores incluyen una serie de certificados en los que ellos confían y, por tanto, en las que sus usuarios confiarán. Esto no impide no obstante que un usuario pueda decidir si confiar o no en estas entidades. Dependiendo del sistema operativo, el usuario puede eliminar (o añadir) las entidades en las que confía.

## Estándar X.509 para certificados digitales

El estándar X.509 es un estándar ITU-T para PKI y para PMI (Privilege Management Infrastructure).

Pensado originalmente como complemento al estándar X.500 en su faceta de autenticación, este estándar fue adoptado fuera del estándar X.500En este estándar se definen formatos para certificados de clave pública, listas de revocación de certificados (CRL), certificados de autorización (relacionados con PMI), así como un algoritmo para comprobar la validez de la cadena de certificados. La última versión publicada de este estándar es la Versión 3, publicada en el RFC 5280 [5], siendo el RFC 3280 [6] la versión inicial y el RFC 6818 [7] la última corrección o actualización del estándar.

Al igual que el estándar X.500 para servicios de directorios, el X.509 se basa en una jerarquía estricta de autoridades de certificación (CA) para la emisión de certificados digitales. Por tanto, nos encontramos con una infraestructura centralizada como de la que hablábamos en el epígrafe anterior. Aquellas CA que están en lo más alto de la jerarquía se llaman Autoridades de Certificación Raíz y son las que suelen estar incluidas en los sistemas operativos y navegadores web. Si pensamos en esta jerarquía como un árbol, en la raíz de la jerarquía está el certificado de la CA Raíz y en las hojas están los certificaos finales que serán usadas por los servidores o lo usuarios para su autenticación. Los nodos intermedios en el árbol son las llamadas CA intermedias. Se considera una buena práctica que los certificados finales sean emitidos por CA intermedias, en vez que por una CA raíz. De esa manera, si un certificado intermedio se ve comprometido, solo es necesario cortar esa rama del árbol. Si un certificado raíz se viera comprometido, entonces habría que cortar todo el árbol con el consiguiente *trauma* que ello podría conllevar.

|  |  |
| --- | --- |
|  |  |

Figura 3. Ejemplos de jerarquías PKI con un certificado intermedio (izquierda) y con dos certificados intermedios (derecha).

## Certificados digitales

Un certificado digital consiste en asociar una clave pública con una serie de detalles adicionales como son:

* A qué persona u organización identifica dicho certificado.
* Quién firma el certificado, es decir, quién asegura que el certificado es del propietario.
* Las fechas desde la que es válido el certificado y la fecha en la que expira.
* Para qué usos está aprobado puede usar el certificado.

El uso más visible de los certificados digitales para un usuario es el de crear una conexión segura a la hora de navegar por Internet. Es usado en conjunto con el protocolo TLS (Transport Layer Security) y permite al usuario confiar en que se está conectando al servidor que pretendía conectarse y, además, tener relativa seguridad de que nadie puede interceptar esa comunicación con el servidor y que queda en secreto entre el usuario y el servidor.

Los certificados pueden estar asociados a nombres de Internet (DNS), en el caso de sitios web, pero también pueden estar asociados a direcciones de correo electrónico o cuentas de usuario de algún servicio de directorio determinado. Es lo que se conocen como certificados de usuario y pueden usarse para cifrar/firmar comunicaciones por correo electrónico o para firmar software, por ejemplo.

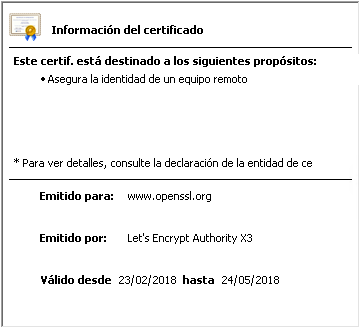


Figura 4. Información del certificado digital de www.openssl.org.

En el caso de una PKIX (PKI X.509), los certificados digitales están asociados a un nombre distinguido (Distinguished Name) del estilo X.500, o bien a un nombre alternativo como una dirección de correo electrónico o una entrada DNS. Una CA firma con su propio certificado un certificado que está expidiendo, de manera que cualquier usuario que confíe en la CA (o su CA raíz) confiará en el nuevo certificado.

### Estructura de un certificado

Los certificados X.509v3 se expresan según la notación ASN.1 definida en el PKCS #1. El RFC 5280 define la estructura de un certificado y una serie de extensiones básicas. Los atributos de lo que se compone un certificado se muestran en la siguiente lista:

* Certificado
* Versión
* Número de serie
* Algoritmo usado para firmar el certificado
* Emisor
* Periodo de validez
  + No antes de
  + No después de
* Sujeto
* Información de la clave pública del sujeto
  + Clave pública
  + Algoritmo de la clave pública
* Identificador único del emisor (opcional)
* Identificador único del sujeto (opcional)
* Extensiones (opcional)
* Firma del certificado
* Algoritmo usado para firmar el certificado

Todas las propiedades de los certificados son importantes, pero cabe resaltar las siguientes:

* El número de serie es un número entero mayor de cero y debe ser único para cada certificado emitido por una determinada CA.
* Emisor: es el nombre de la CA que firma el certificado
* Sujeto: a quién va dirigido el certificado. Por ejemplo, en certificados de usuario puede ser el nombre de la persona o en el caso de un certificado para un servidor web, puede ser la URL de un servicio web.

### Extensiones

Las extensiones en los certificados permiten añadir propiedades extra a los certificados. Estas extensiones pueden estar definidas de manera pública y ser usadas por cualquier sistema, pero también se permite la inclusión de extensiones privadas de manera que solo tienen sentido en el sistema que las define.

Dentro de un certificado una extensión está compuesta de:

* Un identificador único de extensión expresado como un OID (Object Identifier). Este OID se expresa en notación ASN.1 de X.690. En el estándar X.690 [8] se definen una serie de formatos de codificación como BER, CER y DER partes de la notación ASN.1. En este caso, cada OID representa casi todos los tipos de objetos posibles que se pueden incluir en un certificado X.509.
* Una estructura ASN.1 que se corresponde al valor de la extensión.

Una extensión sólo puede incluirse una única vez en un certificado.

Cada extensión se puede acompañar de una indicación para informar de la criticidad de la extensión:

* Extensión crítica: si el sistema que usa un certificado encuentra una extensión crítica que no reconoce o no reconoce la información contenida en la extensión debe rechazar el certificado.
* No crítica: el sistema podría ignorar una extensión no crítica si no la reconoce, pero debe procesarla si la reconoce.

Existe un conjunto de extensiones básicas que todo sistema debe reconocer. Estas extensiones permiten el funcionamiento básico de un sistema PKI, por ejemplo, identificando el uso autorizado del certificado o dónde se puede encontrar la información para comprobar la revocación del certificado. En la siguiente lista se nombran algunas de estas extensiones, con un nombre y su identificador OID en formato ASN.1:

* Basic constraints, { id-ce 19 }: indica que el certificado corresponde a una CA, una CA intermedia o si está al final de la cadena.
* Key Usage, { id-ce 15 }: indica el uso permitido del certificado, por ejemplo, cifrado o firma.
* Extended Key Usage, { id-ce 37 }: indica el propósito del certificado, mediante una lista de OID. Estos propósitos podrían ser: ser usado en una conexión TLS, enviar correo electrónico seguro o identificar a una persona.
* Certificate Policies, { id-ce 32 }: refleja las normas que se han seguido para expedir el certificado y los usos de este.
* Para indicar los puntos de acceso para la comprobación de la revocación del certificado (CRL u OCSP), existen dos extensiones:

CRL Distribution Point, { id-ce 31 }

Authority Information Access, { id-pe 1 }

* Subject Alternative Names, { id-ce 17 }: permite asociar el certificado a una lista de sujetos. Complementa la propiedad Sujeto del certificado, permitiendo que sea válido para más nombres. Una organización, por motivos de coste o técnicos, puede optar por un certificado SAN para publicar una web. Es un caso muy común que un certificado para servicios web tenga la extensión SAN para que acceder a una página web con www y sin ella. Por ejemplo, en la siguiente figura, un usuario puede acceder a <https://www.itu.int> o a <https://itu.int> con el mismo certificado:

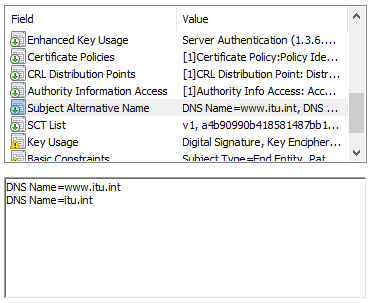


Figura 5. Ejemplo de extensión SAN en el certificado de www.itu.int.

### Formatos de archivo para certificados y PKI

Los formatos de archivo más comunes para almacenar certificados X.509 son los siguientes:

* Formato binario DER. Las extensiones más comunes son: cer, crt, der.
* Privacy-enhanced Electronic Mail. Consiste en un certificado codificado usando la notación DER y posteriormente codificado como Base64 y rodeado por una cabecera y un pie en texto plano. La extensión más común es pem, aunque también es común usar cer o crt.

-----BEGIN CERTIFICATE-----

MIIGOTCCBSGgAwI…

-----END CERTIFICATE-----

* En una estructura PKCS #7, sin datos y usada únicamente para distribuir certificados o CRL. La extensión suele ser p7b o p7c.
* En una estructura PKCS #12, incluyendo claves privadas y certificados, usualmente protegidos con una contraseña. La extensión suele ser p12.
* Usando PFX, el predecesor de PKCS #12, la extensión es pfx. Es también posible encontrar certificados PKCS #12 usando la extensión pfx. El caso habitual es aquellos certificados y claves privadas exportadas desde los sistemas operativos Windows usando su CryptoAPI.

## Cadenas de certificados

En el RFC 3280 se define el concepto de camino de certificados, equivalente al concepto de cadena de certificados. Cuando un usuario usa un servicio que usa un certificado digital, generalmente requiere comprobar la validez de dicho certificado. Para comprobar la validez de un certificado se debe construir la cadena de certificados desde el del servicio hasta el certificado raíz que firma el certificado del servicio.

Para construir la cadena de certificados se inspecciona el emisor del certificado y se compara con el valor del sujeto del propio certificado. Si el emisor es diferente del sujeto, entonces existe un certificado superior en la jerarquía que lo ha firmado. Se realiza esta misma comprobación por cada certificado superior que se descubra hasta que se llegue a un certificado autofirmado, es decir, aquel en el que el sujeto y el emisor sea el mismo. Cuando se llegue a un certificado así, habremos llegado al certificado raíz.

Para que un sistema confíe en un certificado final debe confiar en el certificado raíz que está al final de la cadena de certificados.

Una cadena de certificados tiene las siguientes propiedades:

* La cadena de certificados empieza con el certificado final, le siguen aquellos certificados de CA intermedias y termina con un certificado de una CA raíz. El certificado raíz generalmente será un certificado de una de las CA comerciales o gubernamentales si el certificado es usado públicamente, aunque también puede ser un certificado de una CA no pública como una CA interna de una organización.

Un ejemplo de cadena de certificados en formato pem.

-----BEGIN CERTIFICATE-----

Certificado final

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

Certificado intermedio

-----END CERTIFICATE-----

[...]

-----BEGIN CERTIFICATE-----

Certificado intermedio

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

Certificado raíz

-----END CERTIFICATE-----

* El campo Emisor de un certificado coincide con el campo Sujeto del siguiente certificado de la lista.
* Cada certificado está firmado por la clave privada del siguiente certificado. Evidentemente el último certificado estará firmado por sí mismo para que la cadena pueda terminar.

En las siguientes capturas podemos ver la cadena de certificados para el certificado de [www.itu.int](http://www.itu.int):

COMODO SECURE" 
COMODO RSA Extended Validation Secure Server CA 

Figura 6. Cadena de certificados para www.itu.int.

Machine generated alternative text:
Certificate Info rmation 
This certificate is intended for the following purpose(s): 
• Ensures the identty of a remote computer 
• Proves your identty to a remote computer 
•1.3.6.1.4.1.6449.1.2.1.5.1 
*Refer to the certficabon authority's statement for details. 
vvwvv.itu.nt 
Issued by: COMODO RSA Extended Valida bon Secure Server 
Valid from 28-Mar-17 to 29-Mar-19 
Issuer Statement 

Figura 7. El certificado final está firmado por un certificado intermedio COMODO RSA Extended Validation.

Machine generated alternative text:
Certificate Info rmation 
This certificate is intended for the following purpose(s): 
• Ensures the identty of a remote computer 
• Proves your identty to a remote computer 
• Protects e-mail messages 
• Ensures software came from softvvare publisher 
• Protects software from alteraton after publication 
• Allows data to be signed with the current tme 
*Refer to the certfcaton authority's statement for details. 
Issued to: COMODO RSA Extended Valida bon Secure Server 
Issued by: COMODO RSA Cert ficabon Authority 
Valid from 12-Feb-12 to 12-Feb-27 
Issuer Statement 

Figura 8. El certificado intermedio está firmado por COMODO RSA Certification Authority.

Machine generated alternative text:
Certificate Info rmation 
This certificate is intended for the following purpose(s): 
• Ensures the identty of a remote computer 
• Proves your identty to a remote computer 
• Protects e-mail messages 
• Ensures software came from softvvare publisher 
• Protects software from alteraton after publication 
• Allows data to be signed with the current tme 
Issued to: COMODO RSA Cert fica bon Authority 
Issued by: COMODO RSA Cert ficabon Authority 
Valid from 19-Jan-IO to 19-Jan-38 

Figura 9. COMODO RSA Certification Authority es un certificado raíz ya que es autofirmado.

### Comprobación de la validez de un certificado

Una vez se tiene la cadena de certificados es posible realizar una serie de comprobaciones para determinar si el certificado final que se presente es digno de confianza [9].

Una visión de alto nivel del algoritmo usado para comprobar la validez de la cadena es la siguiente:

1. Empezando por el certificado final, por cada certificado se comprueba:
   1. Que el momento en el que se hace la comprobación está dentro del periodo de validez del certificado.
   2. Se localiza el Emisor del certificado, bien sea porque es el siguiente en la cadena o porque está en la base de datos local de certificados.
   3. Se comprueba la firma del certificado actual con la clave pública del certificado del Emisor obtenido en el punto anterior.
   4. Si el certificado del Emisor venía de la base de datos local de certificados, entonces el proceso termina aquí. En otro caso, se vuelve a empezar el proceso comprobando ahora el certificado Emisor del actual.
2. Además, es posible comprobar la validez del certificado en base a las normas asociadas con cada certificado o con el proceso de validación.
3. Finalmente, también se suele comprobar el estado de la revocación del certificado.

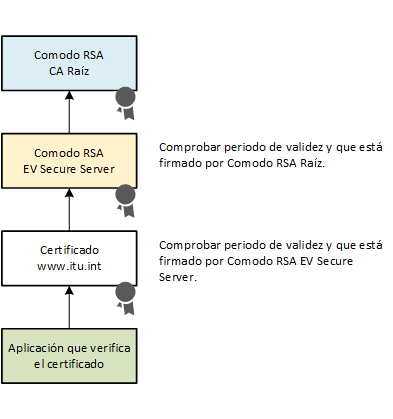


Figura 10. Ejemplo de validación de una cadena de certificados.

Como hemos visto, la comprobación termina cuando en la cadena nos encontramos con un certificado en el que ya confiemos. Es decir, no siempre es necesario llegar hasta el certificado raíz para comprobar la validez de un certificado final. En el ejemplo de la figura 10, si en nuestra base de datos de certificados de confianza nos encontramos con el certificado *Comodo RSA EV Secure Server*, entonces no hace falta que comprobemos este certificado ni el certificado raíz *Comodo RSA CA*.

Una restricción de la comprobación es que un certificado no puede aparecer más de una vez en la cadena. Es decir, no se permiten ciclos en la cadena.

### Certificados autofirmados

Un certificado autofirmado es aquél que está firmado con la clave privada asociada a la clave pública del certificado. Por tanto, en estos certificados el emisor es el mismo que el sujeto. Los certificados raíz, por tanto, deben ser certificados autofirmados, ya que no hay ninguna CA por encima que pueda firmar su certificado.

Debido a la rapidez y facilidad con la que pueden ser emitidos ya que no precisa de ningún proceso de validación por parte de un tercero, estos certificados se suelen usar en entornos de desarrollo de software donde se requiera del uso de un certificado.

## Validación cruzada de certificados

Hay que tener en cuenta que es técnicamente posible tener la misma clave pública firmada por dos CA diferentes y por tanto tener dos certificados para la misma clave pública. En cambio, no es posible tener un único certificado X.509 firmado por más de una CA.

La posibilidad de tener varios certificados por clave pública permite la creación de varias cadenas de certificados para un único certificado ya que se podría construir una cadena por cada certificado raíz.

Gracias a esto es posible, por ejemplo, que un certificado emitido dentro de una PKI pueda ser validado en otra PKI separada y viceversa. La importancia radica en que en cada PKI cada cliente pueda construir la cadena del certificado externo hasta una CA raíz propia de la PKI. En la siguiente figura[[10]](#footnote-11) cada estructura PKI tiene un certificado raíz que está autofirmado y también otro firmado por la CA de la otra estructura PKI. De esta manera es posible construir dos cadenas de certificados para cada certificado de usuario y así, un usuario de PKI 1 puede construir una cadena de certificados para el certificado de Usuario 2 que acabe en el certificado raíz de su estructura PKI.

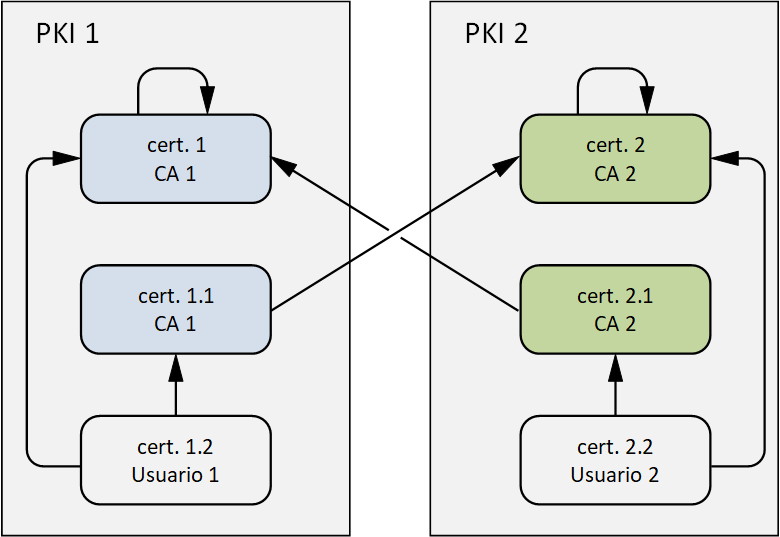


Figura 11. Validación cruzada entre PKI 1 y PKI 2.

La validación cruzada también permite a una CA crear un nuevo certificado raíz sin que los certificados emitidos con el certificado antiguo pierdan su validez. El proceso para generar los nuevos certificados raíz se explica en [10].

## Comprobación de la revocación de un certificado

Cuando se emite un certificado se espera que vaya a ser usado durante todo su periodo de validez. Sin embargo, es posible que un certificado deje de ser usado antes de que expire. Por ejemplo, se ha de cambiar el nombre asociado al certificado, se compromete la clave privada o se termina la relación entre el sujeto y la CA que emitió el certificado. En estos casos, la CA debe revocar el certificado para que no pueda continuar siendo válido.

### Listas de revocación de certificados – CRL

X.509 define un método para revocar certificados que consiste en que cada CA publique periódicamente una lista de revocación de certificados o CRL por sus siglas en inglés Certificate Revocation List. Esta lista consiste en una estructura firmada por la CA o por la entidad que publica la CRL y relaciona una fecha con un certificado que ha sido revocado mediante su número de serie. Por supuesto, estas listas deben ser de dominio público. Lo más normal es que además se distribuyan mediante canales no seguros para que no haya problemas a la hora de negociar la seguridad de la conexión y así poder estar seguros de que un cliente que la solicite pueda descargar la lista sin problemas.

Cuando se ha de comprobar la validez de un certificado, lo normal es obtener una copia más o menos reciente de la CRL de la CA que emitió el certificado y entonces comprobar si el certificado ha sido revocado.

Los certificados que incluyen la extensión CRL Distribution Point incluyen un URI que indica dónde conseguir dicha CRL. Esta URI puede ser una dirección URL a la que se accede por HTTP, una ruta LDAP o incluso una carpeta compartida. Qué tipo de URI se use dependerá del tipo de certificado y, sobre todo, del contexto del uso del certificado. Por ejemplo, un certificado que se use en un sitio web público incluirá una URL a un recurso HTTP. Un certificado que se use únicamente dentro de la red interna de una organización puede incluir una ruta LDAP que apunte a un directorio dentro de dicha organización.

Una limitación del uso de CRL es el tiempo que se tarda en publicar una nueva CRL ya que podría llegar a haber un tiempo considerable entre que un certificado es revocado y la actualización de la CRL de la CA correspondientes.

El formato de la estructura que describe una CRL es el siguiente:

* Algoritmo usado para firmar la CRL
* Firma de la CRL
* CRL
  + Versión (opcional, pero obligatorio con valor entero igual a 1 si se van a usar extensiones)
  + Algoritmo usado para firmar la CRL
  + Emisor (quién emite y firma la CRL)
  + Fecha de emisión
  + Fecha de la siguiente emisión
  + Lista de certificados revocados, por cada certificado
    - Número de serie del certificado
    - Fecha de revocación
    - Extensiones de la entrada
* Extensiones de la CRL

Algunas de las extensiones posibles son para una CRL son:

* Identificador de la clave de la autoridad
* Nombre alternativo del emisor
* Número de CRL

Algunas extensiones posibles de una entrada de certificado son:

* Código de razón de la revocación
* Fecha desde la que se sospecha que el certificado ha sido comprometido

|  |  |
| --- | --- |
|  |  |

Figura 12. CRL de Starfield Secure Certificate Authority - G2. Detalles y certificados revocados.

### Online Certificate Status Protocol – OCSP

Online Certificate Status Protocol es un protocolo usado para obtener el estado de la revocación de un certificado X.509. Se describe en el RFC 6960 [11]. Es una alternativa al uso de CRL y surgió para aliviar algunos de los problemas típicos de CRL. A un servidor OCSP se le conoce como "OCSP Responder".

La comunicación entre un cliente y un servidor OCSP se realiza mediante el protocolo HTTP usando mensajes codificados en ASN.1. Una diferencia respecto a las CRL es que la conexión con un servidor OCSP sí puede ser HTTPS y quizá en algunos casos pueda ser conveniente, como cuando se pregunta sobre el estado de certificados de usuario. De cualquier manera, el uso de HTTPS para comunicarse con el servidor OCSP no es obligatorio.

La principal diferencia entre OCSP y CRL es que cuando se solicita una CRL obtenemos una lista completa de todos los certificados revocados por una CA y esta lista puede llegar a ser de un tamaño considerable. En cambio, a un servidor OCSP se le pide el estado de un único certificado y por tanto la respuesta es generalmente más ligera y conlleva un ahorro de ancho de banda y tiempo de proceso por parte del cliente.

Sin embargo, el uso del protocolo OCSP implica que el propietario de la CA debe tener disponible permanentemente online el servicio para que un sistema pueda contactarlo. Además, este debe responder en el menor tiempo posible para que no provoque lentitud por los tiempos de respuesta en el sistema que comprueba un certificado.

Al igual que con CRL una CA incluye una extensión Authority Information Access en sus certificados donde indica la URL del servidor OCSP que conoce el estado de revocación de los certificados emitidos por la CA.

El protocolo OCSP tiene un funcionamiento en modo pregunta/respuesta:

1. El sistema que comprueba el estado de un certificado pregunta al servidor OCSP asociado al certificado por el estado de este. En la pregunta le envía el número de serie del certificado.
2. El servidor OCSP usa este número de serie para buscar el estado del certificado en la base de datos de la CA. Cuando conoce el estado del certificado, envía una respuesta firmada al usuario indicando el estado del certificado.
3. El usuario comprueba la firma de la respuesta usando la clave pública de la CA que firmó el certificado sobre el que se preguntaba.

Una respuesta consiste en:

* Campos como versión, algoritmo de firma y valor de la firma.
* Identificador del servidor que responde.
* Duración de la validez de la respuesta.
* Respuestas por cada uno de los certificados de la petición.
* Extensiones.

Los posibles valores de la respuesta son:

* El certificado es válido.
* El certificado está revocado.
* No se conoce el estado del certificado.
* Además, también se incluye la duración de la respuesta por cada certificado.

Finalmente, para que un cliente pueda considerar la respuesta como válida debe ser firmada por una de las tres siguientes entidades:

* La CA que emitió el certificado sobre el que se pregunta.
* Un Responder de confianza del usuario que pregunta.
* Un Responder autorizado por la CA. Este Responder tendrá un certificado firmado directamente por la CA emisora que le identificará como Responder autorizado para dar respuestas OCSP para esta CA.

### OCSP Stapling

Para mitigar el principal problema de rendimiento de OCSP, donde un Sistema debe contactar con el respondedor OCSP cada vez que se encuentra un certificado, se definió la extensión Certificate Stauts Request para el protocolo TLS en el RFC 6066 [12].

Esta extensión permite al cliente de una conexión TLS solicitar al servidor el estado de la revocación del certificado. Si el servidor entiende la petición, enviará una respuesta OCSP durante la negociación del protocolo TLS. Por supuesto, para que el cliente pueda aceptar dicha respuesta OCSP, esta tiene haber estado originada en el servidor OCSP original ya que debe estar firmada por la misma CA que firma el certificado del servidor. Por tanto, para que el servidor pueda devolver esta respuesta al cliente, debe haber contactado previamente con el servidor OCSP y haber almacenado dicha respuesta temporalmente. Es esta misma respuesta la que enviará al cliente cuando se la soliciten.

El tiempo que el servidor puede almacenar la respuesta OCSP original depende precisamente del tiempo que dicha respuesta sea válida y está determinado las mismas propiedades de la respuesta.

Cuando un cliente recibe una respuesta debe comprobar su validez y, si no es aceptada, entonces parar la conexión. Si no recibiera una respuesta entonces contactaría directamente con el servidor OCSP.

El problema de que el servidor mantenga en una caché las respuestas OCSP es que esto introduce un retraso en la revocación efectiva de un certificado ya que puede pasar una cantidad de tiempo indeterminada desde que un certificado es revocado hasta que la última respuesta OCSP cacheada expira.

Aun así, esta extensión al protocolo TLS optimiza el establecimiento de una conexión ya que evita que el cliente tenga que contactar al servidor OCSP y obtiene esa respuesta directamente desde el servidor con el que ya tiene una conexión. De esta manera, en el cliente se evitan varias conexiones (DNS para resolver el nombre del servidor OCSP y la petición HTTP hacia el mismo) y también se reduce el número de peticiones que se hacen al servidor OCSP en general.

La especificación de esta extensión solo permite que se envíe una única respuesta OCSP por conexión, por lo que en general es insuficiente para comprobar la validación de un certificado. Al solo tener una respuesta OCSP posible, que será para el certificado del servidor, esto impide poder comprobar por esta vía la validez del certificado raíz o intermedios que pudieran existir en la cadena de certificados. Para solucionar este problema se especifica la extensión TLS Multiple Certificate Status Request Extension [13] que ya permite solicitar y enviar respuestas OCSP para todos los certificados de la cadena.

## Autoridades de certificación

Una autoridad de certificación es la entidad que se encarga de generar certificados a partir de las claves públicas que recibe. De esta manera, es la autoridad certificadora la que garantiza que la clave pública que está firmando pertenece al sujeto del certificado.

Por tanto, una CA es una pieza clave en una estructura PKI jerárquica ya que, en el árbol que se podría representar en dicha estructura, es todos y cada uno de los nodos que no son hojas (certificados finales).

El certificado que usa una CA suele tener:

* Key Usage: Certificate Signing
* Basic Constraints: Subject Type=CA

Adicionalmente, podría tener más valores en las propiedades Key Usage, Key Enhanced Usage o Basic Constraints. Por ejemplo, para firmar una CRL el valor de Key Usage debe ser Off-line CRL Signing o para firmar una respuesta OCSP el valor de Key Enhanced Usage debe ser OCSP Signing.

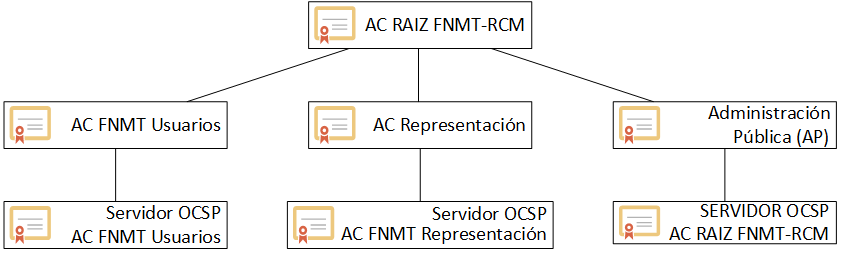


Figura 13. Esquema parcial de la jerarquía de certificados CA de la Fábrica Nacional de Moneda y Timbre (FNMT).

### Proceso de emisión de un certificado

Cuando una entidad quiere solicitar un certificado a una CA, la CA elegida recibe una solicitud de firma de certificado llamada CSR (Certificate Signing Request).

El procedimiento habitual para solicitar un certificado es:

1. Generar una pareja de claves por parte del sistema que solicitará el certificado.
2. Se recogen una serie de datos extras a enviar a la CA como pueden ser:
   * El DN o CN que se quiere relacionar con el certificado.
   * Información sobre la empresa, departamento, geográfica, etc.
   * Una dirección de correo electrónico.
   * Además, dependiendo del tipo de certificado que se esté solicitando las CA suelen requerir otro tipo de información, ya sea de manera telemática o física.
3. Se envía el CSR a la CA.
4. Por lo general, la CA hará algún tipo de validación de la información, sobre todo confirmará la identidad del solicitante con relación al certificado que se haya pedido, por ejemplo:
   * Para un certificado de usuario, por ejemplo, un certificado de usuario de la FNMT, el solicitante suele tener que ir a una de las oficinas de Hacienda o de otro organismo del gobierno con la solicitud y su DNI para que el funcionario de turno haga la comprobación de que el solicitante es el mismo que la persona que se nombre en el CSR[[11]](#footnote-12).
   * Para un certificado para un nombre DNS, por ejemplo, para poder conectarnos a una web usando HTTPS, la CA suele confirmar que el solicitante es administrador o propietario del dominio, con frecuencia enviando un correo electrónico a una lista limitada de direcciones reservadas habituales, por ejemplo: hostmaster, webmaster, etc. Otro método habitual es crear un registro TXT en el servidor DNS con un valor indicado por la CA. De esta manera, se
   * Existen también otros procesos más complejos. Las CA comerciales suelen tener clases de certificados de validación extendida que, en los navegadores modernos, hacen que la barra de direcciones cambia a otro color, para indicar de una manera más llamativa que se está usando una conexión cifrada. Por lo general, es necesario justificar la expedición de estos certificados demostrando el capital de la empresa solicitante, mediante justificantes de impuestos, contacto físico con el administrador/responsable de la empresa o procedimientos similares.

A la hora de solicitar certificados, existen otras autoridades que pueden entrar en juego:

* VA, Validation Authority. Este ente se encargaría de pedir y validar los datos identificativos de la persona que está solicitando el certificado.
* RA, Registration Authority. Este ente se encargaría de aprobar que la persona que solicita el certificado está autorizada para solicitar un certificado en nombre del Sujeto del certificado.

### Tipos de CSR

Independientemente del formato de una solicitud CSR, esta debe incluir una serie de datos mínimos para que una CA pueda generar un certificado a partir de ella. Obviamente, debe incluir la clave pública del sujeto, el sujeto y el uso que se quiere hacer de dicho certificado. Respecto a los formatos de CSR, los más habituales son PCKS #10 y SPKAC.

Cabe decir que una CA es libre de usar los atributos que se envían en la CSR o no. Es común que una CA necesite un CSR únicamente para recibir la clave pública en un formato determinado y que ignore el resto de los atributos en el CSR. Será ella misma la que determine el valor del resto de atributos del certificado.

#### PKCS #10

El formato PKCS #10 [14] es el más usado habitualmente. El formato de una CSR PCKS #10 se compone de tres partes:

* Información de la solicitud. Esta solicitud está formada por:
  + El DN (nombre distinguido) de la entidad que realiza la solicitud.
  + La clave pública.
  + Otros atributos sobre sobre la solicitud o sobre el sujeto. Aquí se pueden usar atributos definidos en PKCS #9 [15] para extender el certificado que se solicita.
* El algoritmo de firma usado para firmar la información de la solicitud.
* El valor de la firma de la solicitud.

#### SPKAC

Signed Public Key and Challenge o SPACK, también conocido como Netscape SPKI.

Este formato de CSR que se manipula con OpenSSL y que es el que implementan los navegadores web compatibles con Mozilla en el elemento keygen. Este elemento forma parte de la especificación HTML5, pero su uso está desaconsejado actualmente ya que los navegadores que lo soportan actualmente tienen planes de quitar su soporte y la última especificación de HTML 5.2 ha quitado el elemento[[12]](#footnote-13). No está soportado por Internet Explorer.

Esta implementación se usa dentro de formularios en páginas web para generar parejas de claves y enviar un CSR a un servidor web (almacenando el navegador la clave privada). El servidor que recibe la solicitud hará las comprobaciones pertinentes y creará un certificado que enviará al navegador. Estos certificados suelen ser por tanto certificados de usuario que después podrá usar el navegador para autenticarse con un sitio web mediante HTTPS.

Dado que es un formato que se envía mediante un formulario web, es un formato en texto plano con una serie de variables en formato Clave=Valor como, por ejemplo[[13]](#footnote-14):

|  |
| --- |
| SPKAC=MIIBXjCByDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA3L0IfUijj7+A8CPC8EmhcdNoe5fUAog7OrBdhn7EkxFButUp40P7+LiYiygYG1TmoI/a5EgsLU3s9twEz3hmgY9mYIqb/rb+SF8qlD/K6KVyUORC7Wlz1Df4L8O3DuRGzx6/+3jIW6cPBpfgH1sVuYS1vDBsP/gMMIxwTsKJ4P0CAwEAARYkYzBkZjFlYjctMTU0NC00MWVkLWFmN2EtZDRkYjBkNDc5ZjZmMA0GCSqGSIb3DQEBBAUAA4GBALEiapUjaIPs5uEdvCP0gFK2qofo+4GpeK1A43mu28lirYPAvCWsmYvKIZIT9TxvzmQIxAfxobf70aSNlSm6MJJKmvurAK+Bpn6ZUKQZ6A1m927LvctVSYJuUi+WVmr0fGE/OfdQ+BqSm/eQ3jnm3fBPVx1uwLPgjC5g4EvGMh8M  CN=Joe Snuffy  OU=Snake Oil, LLC  O=Department of pricing  L=City name  ST=State or province  C=United States  E=joe.snuffy@snakeoil.com |

Ejemplo de SPKAC que incluye la clave pública y el DN del solicitante

Siendo SPKAC la variable crítica dado que es ahí donde se incluye la clave pública. También incluye una frase de paso, contraseña o reto que es enviada por el servidor. Esta contraseña debería ser generada aleatoriamente por el servidor y enviada en cada petición para así impedir que un CSR SPKAC pueda ser reutilizado por un atacante[[14]](#footnote-15).

### Distribución del certificado generado

Dependiendo del tipo de certificado solicitado, la CA distribuirá el certificado de diferente manera:

* Para certificados de usuario o los usados en servidores web o de correo, se suelen descargar de la página web de la CA donde se ha comprado el certificado. También es posible que la CA muestre el certificado en texto plano, codificado en Base64 para que el solicitante lo copie y guarde localmente en su máquina.

Este procedimiento podría parecer inseguro, pero hay que tener en cuenta que para poder usar el certificado descargado es necesario tener la clave privada y esta no se ha transmitido en ningún momento por la red.

* También es posible que el certificado se transmita físicamente a la memoria de una smart card u otro dispositivo físico como un Trusted Platform Module (TPM). Parece lógico pensar entonces que cuando se emite un certificado de este tipo, el solicitante debe desplazarse físicamente a la entidad que lo haya expedido. Un ejemplo de esto es el DNIe. Cuando el certificado de usuario que lleva incorporado el DNIe expira, el ciudadano debe ir a una comisaría de Policía donde debe ir a una máquina que escribirá el certificado renovado en el chip de su DNIe.

### Seguridad relacionada con una CA

Como se puede suponer, si se compromete la clave privada de una CA, todos los certificados que hayan sido firmados por esa clave, deberían dejar de ser válidos, incluidos los certificados de otras CA que esta clave haya firmado. Esto es porque ya no se puede confiar en los certificados que se emitan desde el momento en el que se sepa que la CA ha sido comprometida. El poseedor de la clave privada comprometida podría generar tantos certificados como quisiera haciéndose pasar por la CA.

Por este motivo los certificados de una CA raíz se suelen mantener en secreto y offline y se suele confiar en una CA subordinada para generar certificados para entidades finales.

## Confianza en una PKI

¿Podemos realmente confiar en los certificados raíz o intermedios que vienen incluidos en los navegadores web o en los sistemas operativos?

El usuario normal "confía" ciegamente en las CA que su sistema operativo o navegador trae instaladas y no se preocupa por el proceso de comprobar la validez de los certificados. Evidentemente, la gran mayoría de los usuarios desconoce el sistema PKI o ni siquiera sabe que lo está usando por lo que este proceso de confianza se obvia en la mayoría de los casos.

Aunque aún queda mucha educación de cara a los usuarios, se les ha enseñado que están protegidos cuando en su navegador web se ve el icono de un candado y entonces el usuario ya se siente seguro sin sentir la necesidad de ninguna otra comprobación. Por su parte, los navegadores suelen mostrar páginas de error o mensajes bastante llamativos cuando el usuario quiere entrar en una página web mediante HTTPS y el certificado del servidor tiene algún problema de validación.

Los certificados que normalmente están instalados por el sistema operativo son certificados raíz y certificados intermedios, pero ¿qué impide a una CA firmada por uno de estos certificados raíz u otra CA subordinada de estas generar un certificado para un nombre que ya existe y realizar un ataque tipo man-in-the-middle? Obviamente hace falta comprometer más dispositivos que un certificado para poder realizar un ataque MITM, pero la posibilidad existe. Una organización podría hacer uso de un certificado de CA para realizar una monitorización de las conexiones cifradas dentro de la empresa si instalara un dispositivo de red a modo de proxy que intervenga las conexiones HTTPS y que pudiera generar al vuelo certificados para los nombres de servicios a los que sus empleados quieran entrar. Existen mecanismos para confirmar la identidad del servidor, pero un usuario medio no va a saber ni siquiera que el MITM está ocurriendo. Para el caso de una CA raíz pública, muchas de estas siguen las pautas establecidas en *Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates*[[15]](#footnote-16). Estas pautas indican qué tipo de políticas y normas son aceptables para las entidades que disponen de una CA pública.

Por otra parte, ¿podemos estar seguros de que nadie puede usar nuestra clave privada? Normalmente, un usuario guardará su clave privada en su máquina y puede que esté comprometiendo su clave al hacerlo así. Un virus o software malicioso podría estar usando la clave sin que el usuario lo sepa. Por otra parte, incluso si la clave está protegida por una contraseña, ¿cómo de segura es esta contraseña? Si el certificado se almacenara en una Smart-Card o TPM usando un código de acceso PIN estamos ante la misma pregunta. Si no podemos estar seguros de que nadie más está teniendo acceso a nuestra clave privada, ¿realmente se puede garantizar la seguridad en la autenticación que proporciona el sistema?

Confiamos en que el sujeto del certificado es el mismo que el que nos presenta el certificado, pero ¿podemos estar seguros de que son la misma persona? ¿Cómo podemos saber que no ha sido otra persona la que ha solicitado el certificado haciéndose pasar por el sujeto del certificado? Es cierto que muchas CA comerciales usan el sistema de Validación Extendida, pero un usuario no tiene por qué saber qué es eso ni cómo funciona y una organización puede no querer gastar ese dinero extra en obtener un certificado de este tipo. Por esto es tan importante el método de validación de la identidad del solicitante de cara a emitir un certificado.

Finalmente, ¿qué pasa si se nos presenta un certificado que no usa CRL u OCSP? Dependemos completamente de la implementación de PKI del software que estemos usando para que nos advierta de tal problema. Incluso si se usa CRL/OCSP, es posible desactivar esta comprobación en muchas aplicaciones por lo que estamos perdiendo un gran grado de seguridad.

Como vemos, el sistema PKI se basa en la confianza ciega en el propio sistema y en que las CA van a hacer bien su trabajo. Bien implementada y desplegada es un sistema seguro, pero seguimos siendo personas y se pueden cometer errores.

# ¿Resultados?

# ¿Conclusiones?

# Planificación

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Estimación inicial** | **Tiempo empleado** | **Error relativo** |
| Búsqueda y estudio de la documentación | 50 | 45 |  |
| Introducción y planificación | 60 | 76 |  |
| Requisitos | 40 | 20 |  |
| Instalación de la plataforma | 10 |  |  |
| Prototipo del diseño | 40 | 12 |  |
| Implementación | 250 |  |  |
| Revisión final de la documentación | 30 | 12 |  |
| **Total** | **480** | 165 |  |
| Error relativo medio |  |  |  |

# Bibliografía

|  |  |
| --- | --- |
| [1] | A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996. |
| [2] | RSA Laboratories, «Public-Key Cryptography Standards (PKCS),» [En línea]. Available: https://web.archive.org/web/20170417085550/https://www.emc.com/emc-plus/rsa-labs/standards-initiatives/public-key-cryptography-standards.htm. [Último acceso: Enero 2018]. |
| [3] | ITU-T, «Abstract Syntax Notation One (ASN.1) Recommendations,» [En línea]. Available: https://www.itu.int/ITU-T/studygroups/com17/languages/. [Último acceso: Enero 2018]. |
| [4] | «Características Chip DNIe,» [En línea]. Available: https://www.dnielectronico.es/PortalDNIe/PRF1\_Cons02.action?pag=REF\_240&id\_menu=[26\_%2030]. [Último acceso: December 2017]. |
| [5] | ITU-T, «RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,» [En línea]. Available: https://tools.ietf.org/html/rfc5280. [Último acceso: Enero 2018]. |
| [6] | ITU-T, «RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,» [En línea]. Available: https://tools.ietf.org/html/rfc3280. [Último acceso: Enero 2018]. |
| [7] | ITU-T, «RFC 6818: Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,» [En línea]. Available: https://tools.ietf.org/html/rfc6818. [Último acceso: Enero 2018]. |
| [8] | ITU-T, «Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER),» [En línea]. Available: http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf. [Último acceso: Enero 2018]. |
| [9] | Oracle, «Sun Directory Server Enterprise Edition 7.0 Reference: Verifying a Certificate Chain,» [En línea]. Available: https://docs.oracle.com/cd/E19424-01/820-4811/gdzea/index.html. [Último acceso: Enero 2018]. |
| [10] | PKI Forum, «Understanding Certification Path Construction,» [En línea]. Available: http://www.oasis-pki.org/pdfs/Understanding\_Path\_construction-DS2.pdf. |
| [11] | ITU-T, «RFC 6960: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP,» [En línea]. Available: https://tools.ietf.org/html/rfc6960. |
| [12] | ITU-T, «RFC 6066: Transport Layer Security (TLS) Extensions: Extension Definitions,» [En línea]. Available: https://tools.ietf.org/html/rfc6066. |
| [13] | ITU-T, «RFC 6961: The Transport Layer Security (TLS) Multiple Certificate Status Request Extension,» [En línea]. Available: https://tools.ietf.org/html/rfc6961. |
| [14] | RSA Laboratories, «RFC 2986: PKCS #10: Certification Request Syntax Specification Version 1.7,» [En línea]. Available: https://tools.ietf.org/html/rfc2986. [Último acceso: Marzo 2018]. |
| [15] | RSA Laboratories, «RFC 2985: PKCS #9: Selected Object Classes and Attribute Types Version 2.0,» [En línea]. Available: https://tools.ietf.org/html/rfc2985. [Último acceso: Marzo 2018]. |
| [16] | Certicom Corp, «Certicom Announces Elliptic Curve Cryptography Challenge Winner,» [En línea]. Available: https://web.archive.org/web/20110719233751/https://www.certicom.com/index.php/2004-press-releases/36-2004-press-releases/300-solution-required-team-of-mathematicians-2600-computers-and-17-months-. [Último acceso: Enero 2018]. |
| [17] | National Institute of Standards and Technology, «Recommendation for Key Management, Part 1: General,» [En línea]. Available: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final. |

1. CC o Common Criteria es una estándar ISO (ISO/IEC 15408) de certificación de seguridad informática. PP es el perfil de certificación Protection Profile y EAL significa Evaluation Assurance Level, la nota que se le otorga, que va desde 1 a 7, siendo 7 la mayor. [↑](#footnote-ref-2)
2. RFC 4086: Randomness Requirements for Security <http://tools.ietf.org/html/rfc4086> [↑](#footnote-ref-3)
3. PKCS #1: RSA Cryptography Specifications Version 2.2: <https://tools.ietf.org/html/rfc8017> [Accedido enero 2018] [↑](#footnote-ref-4)
4. Factorization of a 768-bit RSA modulus: Cryptology ePrint Archive: Report 2010/006: <https://ia.cr/2010/006> [Accedido enero 2018] [↑](#footnote-ref-5)
5. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms: IEEE Transactions on Information Theory, Vol 31, Julio 1985: <http://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf> [Accedido enero 2018] [↑](#footnote-ref-6)
6. What’s Elgamal?: <https://www.gnupg.org/faq/gnupg-faq.html#define_elgamal> [Accedido enero 2018] [↑](#footnote-ref-7)
7. FIPS 186-4: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf> [Accedido enero 2018] [↑](#footnote-ref-8)
8. Recommendation for Key Management, Part 1: General [17] [↑](#footnote-ref-9)
9. Certicom Announces Elliptic Curve Cryptography Challenge Winner [16] [↑](#footnote-ref-10)
10. Traducción de Cross-Validation Diagram. Wikipedia: <https://commons.wikimedia.org/wiki/File:Cross-certification_diagram.svg> [Accedido febrero 2018] [↑](#footnote-ref-11)
11. Manual Solicitud Certificado Persona Física. FNMT-Ceres. <https://www.cert.fnmt.es> [Accedido marzo 2018] [↑](#footnote-ref-12)
12. W3C. Especificación HTML 5.2: <https://www.w3.org/TR/html52/changes.html#features-removed> [Accedido marzo 2018] [↑](#footnote-ref-13)
13. SPKAC format for Certificate Signing Request (CSR): <https://github.com/jas-/node-spkac> [Accedido marzo 2018] [↑](#footnote-ref-14)
14. OpenSSL. Manual de spkac: <https://www.openssl.org/docs/manmaster/man1/spkac.html> [Accedido marzo 2018] [↑](#footnote-ref-15)
15. CA/Browser Forum. Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates: <https://cabforum.org/baseline-requirements-documents/> [Accedido marzo 2018] [↑](#footnote-ref-16)