UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Ampliación y mejora de servicios en la infraestructura de clave pública para e-ciencia de la UNLP (PKIGrid UNLP)

Autores: Guido José Celada, Juan Manuel Filandini

Director: Paula Venosa

Codirector: Nicolas Macia

Carrera: Licenciatura en Informática, Licenciatura en Sistemas.

Resumen

La Universidad Nacional de La Plata cuenta con una infraestructura de clave pública (PKIGrid UNLP) para emisión de certificados para Argentina acreditada por TAGPMA la cual soporta las actividades de e-ciencia de la comunidad académica Argentina. En esta tesina de grado se presenta una alternativa a la tecnología que se utiliza actualmente en PKIGrid UNLP para la infraestructura de clave pública, proponiendo mejoras a los servicios existentes y evaluando la posibilidad de incorporar nuevos servicios.

Palabras Claves

PKI, OpenCA, EJBCA, Certificados, CA, RA, Migración, Adaptación, Mantenibilidad, openssl, Seguridad, CeSPI, Arquitectura, TAGPMA, PKIGridUNLP.

Trabajos Realizados

Análisis de tecnologías utilizadas para la administración, generación, y manutención de certificados.
Propuesta de migración de tecnología de la PKIGridUNLP.
Configuracion y personalizacion de EJBCA en una máquina virtual con los cambios propuestos aplicados siguiendo las políticas de certificados que utiliza actualmente PKIGridUNLP.

Conclusiones

EJBCA tiene mucho potencial, las mejoras que este presenta sobre OpenCA superan en amplio margen cualquier tipo de contingencia generada por el proceso de migración. Queda demostrado que el cambio de tecnología es factible y representaría una gran mejora en el sistema utilizado actualmente.

Trabajos Futuros

Realizar mejoras para contribuir al código fuente open-source de EJBCA. Mejorar la arquitectura del código, las tecnologías usadas en front-end, utilizar Spring MVC en el backend, y cambiar las tecnologías de compilación.

Fecha de la presentación: NOVIEMBRE 2016

Universidad Nacional de La Plata

Facultad de Informática

Ampliación y mejora de servicios en la infraestructura de clave pública para e-ciencia de la UNLP (PKIGrid UNLP)

Guido José Celada

Juan Manuel Filandini

Directores:
Paula Venosa

Codirector:
Nicolás Macia

Noviembre 2016

Índice

1- Introducción

1.1- Motivación

La Universidad Nacional de La Plata cuenta con una infraestructura de clave pública (PKIGrid UNLP) para emisión de certificados para Argentina acreditada por TAGPMA la cual soporta las actividades de e-ciencia de la comunidad académica Argentina.

The Americas Grid Policy Management Authority (TAGPMA) es una federación de proveedores de autenticación y partes de confianzas encabezadas por una Autoridad de administracion de politicas de aquellos responsables de redes en américa del norte,central y del sur. El objetivo de TAGPMA es fomentar las relaciones de confianza entre dominios que son necesarios para desplegar redes en las Américas y en todo el mundo.

La federación está abierta para todo proveedor de autenticación relacionado con redes o partes de confianzas en las américas. A través de múltiples perfiles de autenticación reconocidos, trabajan con muchos tipos diferentes de autoridades implementen una infraestructura de clave pública clásica o no.

La infraestructura de clave pública de la UNLP se encuentra implementada en OpenCA el cual es un proyecto opensource que implementa una Autoridad de Certificación robusta, utilizando los servicios provistos por otras aplicaciones opensource como OpenSSL, Apache y PostgreSQL. A esta PKI se le realizaron diversas adaptaciones, no solo para tener en cuenta requerimientos iniciales, sino también para cumplimentar con recomendaciones de seguridad de TAGPMA como ser la migración del algoritmo de hash utilizado para la firma de los certificados digitales (de SHA1 a SHA2).

Es de interés para la UNLP mejorar los servicios brindados e incorporar nuevos servicios como así también evaluar otras alternativas para la implementación de la autoridad de certificación.

1.2- Objetivo

Presentar una alternativa a la tecnología que se utiliza actualmente en PKIGrid UNLP para la infraestructura de clave pública, proponiendo mejoras a los servicios existentes y evaluando la posibilidad de incorporar nuevos servicios.
Realizar la migración tecnológica de la infraestructura presente a la alternativa propuesta.

2-Marco teórico

2.1- Introducción a la seguridad en redes

La idea de seguridad en sistemas informáticos se basa en proteger la información, ya que está muchas veces constituye un activo muy importante con un rol fundamental. Se debe garantizar la seguridad y la privacidad de esta.
Esto significa que al garantizar la seguridad se debe proteger la información y los servicios del acceso, uso, divulgación, interrupción, modificación o destrucción no autorizados. Y al garantizar la privacidad se debe no revelar la información, o hacerlo selectivamente.
En otras palabras más formales, se debe garantizar sobre la información:

- Confidencialidad: Sólo es accesible por las personas autorizadas.
- Integridad: Sólo puede ser modificada por quien está autorizado a hacerlo.
- Disponibilidad: Los usuarios autorizados tienen acceso cuando lo necesiten.
- Autenticidad: La persona es quien dice ser.
- No repudio: La persona que envió o recibió el mensaje no puede negar haberlo hecho.

La seguridad en redes consiste en políticas utilizadas para prevenir, monitorear y controlar el acceso no autorizado, uso malintencionado, o denegación de una red, y recursos de la misma. La seguridad en redes empieza con la autorización del acceso a datos en la red sobre la cual se trabaja. Cubre una gran variedad de redes, sean públicas o privadas, que se usan cotidianamente tanto en ambientes hogareños como laborales, para

realizar transacciones, comunicación de negocios, etc.
Una manera de otorgar seguridad a los servicios es mediante la utilización de certificados digitales para la autenticación de usuarios, firma digital de documentos y cifrado de los mismos.

2.2- Conceptos básicos de criptografía

Para establecer una comunicación de datos entre dos entidades, ya sea personas o equipos informáticos, hacen falta al menos tres elementos básicos: el emisor del mensaje, el receptor del mismo y un medio o soporte físico por el cual se transfieren los datos. En una comunicación normal los datos se envían a través del medio tal como son, sin sufrir modificaciones de ningún tipo, de tal forma que el mensaje puede ser interceptado y leído por cualquier otra entidad que acceda a él durante su viaje por el medio.
Hay ocasiones en las que nos interesa que el mensaje sólamente pueda ser interpretado correctamente por el emisor del mismo y por el receptor al que va dirigido. En estas ocasiones es necesario implementar algún mecanismo de protección de la información sensible tal que el mensaje viaje seguro desde el emisor hasta el receptor, siendo imposible la interceptación por terceros del mensaje, o que si se produce ésta, el mensaje capturado sea incomprensible para quien tenga acceso al mismo.
La criptografía ha demostrado con el tiempo ser una de las mejores técnicas para resolver esta cuestión. Tanto es así que actualmente es el mecanismo más usado en los procesos de protección de datos, como las transacciones bancarias por Internet, el correo electrónico cifrado, etc.

2.2.1 - Objetivos de la criptografía

La criptografía actualmente se encarga del estudio de los algoritmos, protocolos y sistemas que se utilizan para dotar de seguridad a las comunicaciones, a la información y a las entidades que se comunican. El objetivo de la criptografía es diseñar, implementar, implantar, y hacer uso de sistemas criptográficos para dotar de alguna forma de seguridad.
Las bases de la criptografía se basan en distribuir un mensaje garantizando:

- Confidencialidad: Se garantiza que la información está accesible únicamente a personal autorizado.
- Integridad: Se garantiza la corrección y completitud de la información.

- Vinculación: Permite vincular un documento o transacción a una persona o un sistema de gestión criptográfico automatizado. Cuando se trata de una persona, se trata de asegurar su conformidad respecto a esta vinculación de forma que pueda entenderse que la vinculación gestionada incluye el entendimiento de sus implicaciones por la persona.
- Autenticación: Se proporcionan mecanismos que permiten verificar la identidad del comunicador.

2.2.2- Tipos de sistemas criptográficos

Existen dos tipos básicos de criptosistemas: simétricos y asimétricos.

Criptografía simétrica o de clave privada:

En este tipo se criptosistemas se encripta y desencripta el mensaje usando una clave compartida entre los que se comuniquen el mensaje.

El funcionamiento de la criptografía simétrica es el siguiente: el emisor quiere hacer llegar un documento al receptor. Toma ese documento y le aplica el algoritmo simétrico, usando la clave compartida. El resultado es un documento cifrado que se puede enviar por la red pública. Cuando el receptor recibe este documento cifrado, le aplica el mismo algoritmo con la misma clave, pero ahora en función de descifrar, obteniendo el documento original.

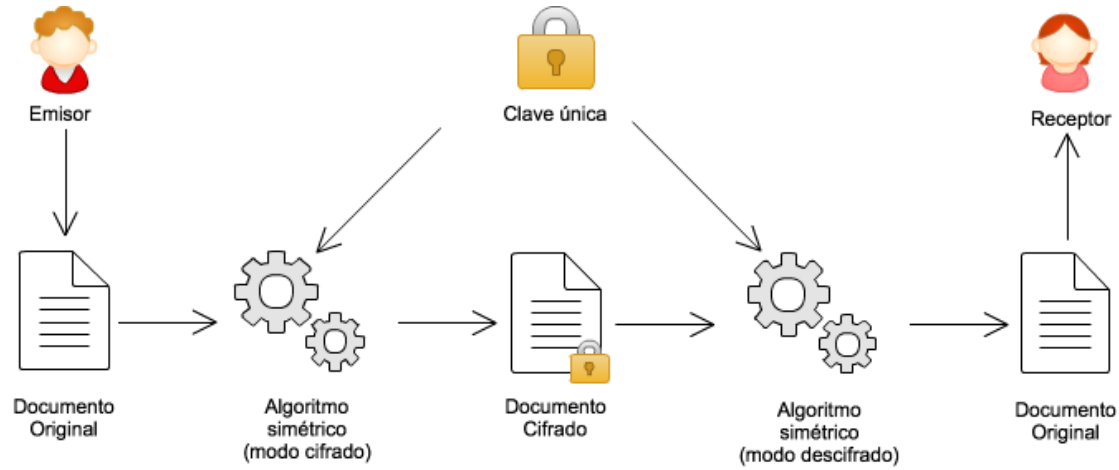


Figura 1. Criptografía simétrica

La criptografía simétrica es rápida y no aumenta el tamaño del mensaje, pero se debe tener en cuenta que la seguridad del sistema criptográfico depende del secreto compartido, por lo tanto la distribución de ésta debe hacerse a través de un medio seguro.

Criptografía asimétrica o de clave pública:

En este tipo de criptosistema se utilizan dos claves matemáticamente relacionadas entre sí, una privada, la cual sólo es conocida por el dueño de la clave, y una pública, conocida por todos. Ambas pueden usarse para encriptar y desencriptar, dependiendo del modo de operación utilizado. Los modos de operación son:

- *Modo encriptación*: Se encripta el mensaje usando la clave pública del receptor, y luego se desencripta usando la clave privada del receptor. De este modo el mensaje es encriptado y solo puede ser desencriptado por el receptor.

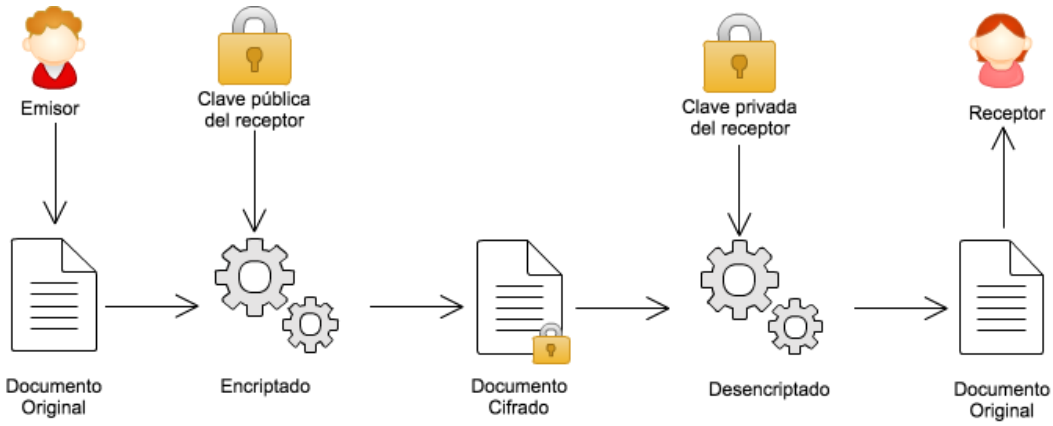


Figura 2. Criptografía asimétrica en modo encriptación

- *Modo autenticación*: Se encripta el mensaje usando la clave privada del emisor, luego se desencripta con la clave pública del emisor. El mensaje puede ser enviado a varios destinatarios. De este modo se puede confirmar que el emisor es quien dice ser.

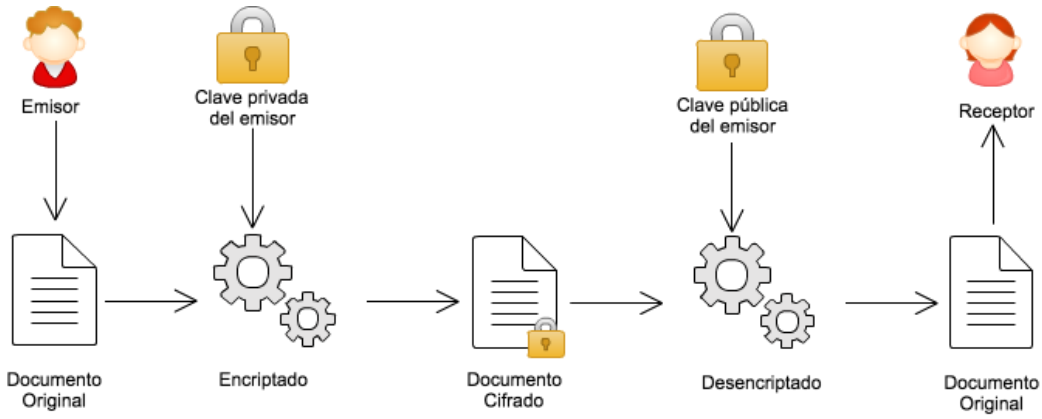


Figura 3. Criptografía asimétrica en modo autenticación

La criptografía asimétrica posee la ventaja de que no se precisa de un intercambio de claves. Este tipo de sistema criptográfico cubre gran parte de los requisitos de seguridad de la información.

2.3- Infraestructura de clave pública (PKI)

La Public Key Infrastructure (PKI) es la infraestructura encargada del manejo de los certificados digitales (crear, organizar, almacenar, distribuir y mantener claves públicas). Esta es básicamente una manera de relacionar las identidades de personas u organizaciones con sus respectivas claves públicas.

2.3.1- Componentes

Una PKI, como su nombre lo indica, es una infraestructura la cual posee diversos componentes, cada uno con una función específica. Estos son una serie de servidores, software, y/o personas que trabajan en conjunto.
Entrando más en detalle, los componentes de una PKI son los siguientes:

Certificados digitales (X.509):

Un certificado digital es un tipo de archivo conocido el cual confirma la identidad de alguien en internet. Este archivo asocia datos de identificación a una persona física, un organismo o empresa.
X.509 es el estándar que dictamina el formato universal para los certificados digitales. Un certificado digital consta de los siguientes datos:

- **Versión del certificado:** El número de versión de X.509 (1, 2, o 3).
- **Número de serie:** Identificador único dentro de la CA.
- **Identificador de algoritmo de firma:** Algoritmo utilizado (ej: SHA1)
- **Nombre del emisor:** Nombre de la CA.
- **Periodo de validez:** Fecha de creación y expiración del certificado.
- **Nombre del sujeto:** Nombre completo de la persona que requiere el certificado.
- **Clave pública:** La clave pública del sujeto.
- **Email del sujeto:** Su e-mail.
- **URI del sujeto:** Página web del sujeto.

Podemos observar un ejemplo de certificado (reducido) en la siguiente imagen:

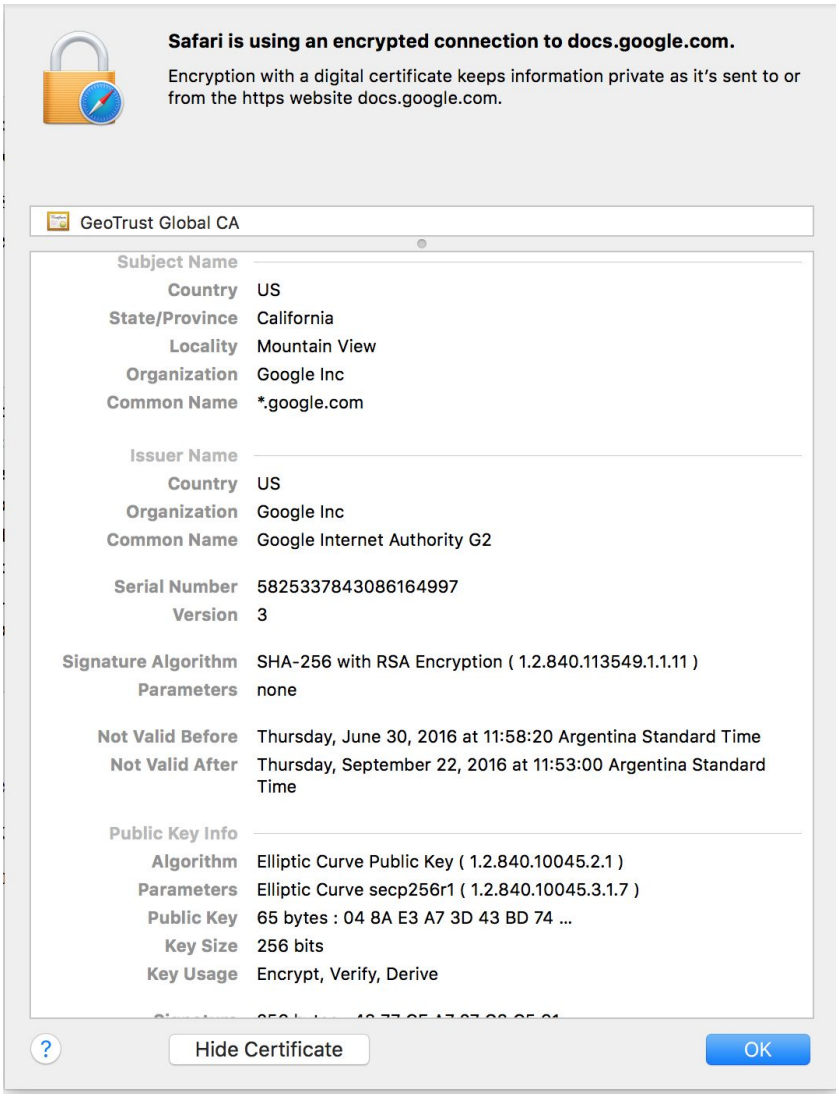


Figura 4. Ejemplo de certificado digital de Google (google.com)

Existen distintos tipos de certificados digitales, en función a la información que contienen cada uno y a nombre de quien son emitidos los certificados. Por ejemplo:

- Certificado personal, que acredite la identidad del titular.
- Certificado de pertenencia a empresa, que además de la identidad del titular acredita su vinculación con la entidad para la que trabaja.
- Certificado de servidor seguro, utilizado en los servidores web que quieren proteger ante terceros el intercambio de información con los usuarios.
- Certificado de firma de código, para garantizar la autoría y la no modificación del código de aplicaciones informáticas.

Autoridad Certificadora (CA):

Es una entidad de confianza la cual tiene la responsabilidad de emitir y revocar certificados digitales. Esta utiliza la criptografía de clave pública para su implementación.

La autoridad de certificación se encarga de verificar la identidad del solicitante de un certificado antes de su expedición. Los certificados recogen ciertos datos de su titular y su clave pública, estos son firmados electrónicamente por la autoridad de certificación utilizando su clave privada. La CA legitima ante los terceros que confían en sus certificados la relación entre la identidad de un usuario y su clave pública.

Autoridad de Registro (RA):

La función de la autoridad de registro es controlar la generación de certificados para los miembros de una entidad. La autoridad de registro se encarga de:

- Identificar a la entidad registrante, comprobando la veracidad y corrección de los datos que se aporten en la petición.
- Realizar la petición del certificado a una CA.
- Guardar los datos pertinentes.

Dependiendo del tamaño de la organización, la RA y CA pueden combinarse en un mismo componente.

Repositorio de certificados:

El repositorio de certificados es el componente encargado de hacer disponibles las claves públicas de las entidades registradas antes de que puedan utilizar sus certificados. Cuando el usuario necesita validar un certificado debe consultar el repositorio de

certificados para verificar la firma del firmante del certificado, garantizar la vigencia del certificado comprobando su periodo de validez y que no ha sido revocado por la CA y que además cumple con los requisitos para los que se expidió el certificado; por ejemplo, que el certificado sirve para firmar correo electrónico.

Listas de revocación de certificados (CRL):

Otro componente disponible en el repositorio de certificados son las CRL, también conocidas como listas de revocación de certificados. En estas listas se publican los certificados revocados.

Un certificado revocado es un certificado que no es válido aunque se emplee dentro de su período de vigencia. Un certificado revocado tiene la condición de suspendido si su vigencia puede restablecerse en determinadas condiciones.

Una CRL es generada y publicada periódicamente, a menudo en intervalos definidos. También puede ser publicada inmediatamente después de que un certificado fue revocado. La CRL es siempre emitida por la CA que emite los certificados correspondientes. Todas las CRL tienen un tiempo de vida durante el cual son consideradas válidas, generalmente dicho tiempo es de 24 horas.

Durante el período de validez de una CRL puede ser consultada por una aplicación para verificar un certificado previo a su uso.

Para prevenir spoofing o denial-of-service attack, la CRL usualmente acarrea una firma digital asociada con la CA que las publico. Para validar una URL específica antes de ser utilizada, el certificado de su CA correspondiente es necesitado, el cual puede ser encontrado generalmente en un repositorio público.

Políticas de certificados y Declaraciones de prácticas de certificación:

Estas son básicamente documentos estructurados que definen las operaciones, prácticas y procedimientos para el uso, administración y manejo de los certificados en una PKI, por ejemplo: Almacenamiento de los registros de la CA, backup, expiración y recuperación de claves, delegación de CA a RA, tipos de certificados, etc.

2.3.2- Flujo de vida de un certificado

El flujo de vida de un certificado consiste de los siguientes (aunque no necesariamente todos) eventos:

- **Generación de claves:** Se generan las claves pública y privada asociadas al certificado digital.
- **Presentación de la solicitud:** Se entregan a la CA las credenciales de la parte que solicita el certificado.
- **Registración:** Se registra en la CA la solicitud de nuevo certificado.
- **Certificación:** Se valida la identidad de la parte solicitante y se genera un certificado firmado digitalmente por la CA.
- **Distribución:** La CA pública el certificado generado.
- **Uso:** La parte solicitante utiliza el certificado para su uso autorizado.
- **Vencimiento:** A no ser que el certificado sea renovado o revocado, este expira en la fecha indicada en la sección de periodo de validez del mismo certificado.
- **Revocación:** Un certificado puede ser revocado en cualquier momento anterior a la fecha de expiración del mismo.
- **Renovación:** Un certificado puede ser renovado por la CA a pedido de su propietario. Este proceso requiere la generación de un nuevo par de claves.
- **Suspensión:** Un certificado puede ser suspendido temporalmente.
- **Recuperación:** Es el proceso mediante el cual se recupera el par de claves a partir de un backup luego de un evento de corrupción de archivos.
- **Destrucción:** Cuando un certificado expira y un lapso considerable de tiempo pasa, es esencial que todas las copias sean destruidas de todas las ubicaciones en donde puedan estar almacenadas.

2.3.3- Usos y aplicaciones de una PKI

Una PKI puede ser utilizada en infinitos escenarios, sin embargo, estos pueden ser abstraídos en dos categorías básicas combinables:

Autenticación:

Una PKI brinda el servicio de autenticar usuarios (y/o servidores) ya que puede utilizar el mecanismo de criptografía de clave pública en modo autenticación, el cual permite verificar fehacientemente la identidad de un usuario o servidor registrado en la PKI. A través de este mismo mecanismo también se permite vincular un documento o transacción a un usuario o servidor.

Un ejemplo en concreto de esta categoría y uno de los usos más populares de PKI es el de navegación segura en internet utilizando SSL (HTTPS). Este requiere de certificados digitales de identificación al servidor para poder confirmar su identidad de una manera confiable al cliente que ingresa a una página web de este servidor.
Otros ejemplos de esta categoría:

- *Firma digital de documentos*
- *Firma digital de emails*
- *Firma digital de software*

Encriptación:

Utilizando el mecanismo de criptografía de clave pública en modo encriptación se provee la funcionalidad de encriptación de todo tipo de datos a los usuarios registrados en la PKI.
Ejemplos en concreto de esta categoría:

- *Encriptación de e-mail*
- *Encriptación de documentos sensibles*
- *Navegación segura encriptada utilizando SSL*
- *Redes inalámbricas seguras utilizando PEAP & EAP-TLS*
- *Encriptación de filesystem*

2.3.4- Operatoria básica

La operatoria básica de una PKI consiste en la siguiente serie de pasos:

Generación del par de claves pública y privada:

Este es el primer paso para trabajar con una PKI. Aquí el usuario que quiere encriptar y enviar el mensaje primero genera el par de claves pública y privada. La clave privada es utilizada para firmar la información y la clave pública es utilizada para verificar esa firma.

Firmas digitales para identificar al emisor del mensaje:

Una firma digital adjunta con un mensaje encriptado identifica al emisor del mensaje. Esta firma es una función matemática que se deriva del par de claves generado. El proceso para adjuntar la firma al mensaje es:

1. Convertir el mensaje original en una palabra de longitud fija aplicando una función de hash al mensaje. Este proceso es conocido como *hashing* y la palabra obtenida *message digest*.
2. Encriptar el *message digest* con la clave privada del emisor. El resultado es en efecto la firma digital.
3. Adjuntar la firma digital al mensaje original.

Encriptar el mensaje:

Luego de aplicar la firma digital al mensaje original, se puede asegurar encriptando el mensaje. Para encriptar el mensaje y la firma digital se utiliza una clave simétrica.

Transmitir la clave simétrica:

Luego de encriptar el mensaje junto a la firma digital, la clave simétrica utilizada para encriptar el mensaje debe ser trasmitida al receptor del mensaje para que este pueda desencriptar el mismo. La clave simétrica debe ser únicamente conocida por el emisor y el receptor del mensaje ya que si esta es comprometida cualquiera podría desencriptar el mensaje y ver el contenido. Para proteger a la clave simétrica esta es encriptada utilizando la clave pública del receptor, logrando que sólo el receptor del mensaje pueda desencriptar la clave simétrica utilizando su clave privada.

Verificar la identidad del emisor del mensaje utilizando una CA:

Cuando el receptor recibe el mensaje encriptado, este puede pedir a la CA la verificación de la firma digital adjunta al mensaje. La CA en este caso entonces verifica y asegura que el emisor del mensaje es quien dice ser.

Desencriptar el mensaje y verificar sus contenidos:

Luego de que el mensaje encriptado es recibido es necesario desencriptarlo. El mensaje solo puede ser desencriptado utilizando la clave simétrica encriptada enviada con el mensaje. Por lo tanto, primero se desencripta la clave simétrica utilizando la clave privada del receptor, y luego se utiliza esta clave simétrica para desencriptar el mensaje. La firma digital adjunta con el mensaje es desencriptada utilizando la clave pública del emisor, obteniendo el message digest. Este message diggest se compara con un hash realizado nuevamente sobre el mensaje original para comprobar que el mensaje no fue alterado en el camino.

2.4- Software para administración de PKI

A la hora de querer implementar el uso de PKI existen múltiples alternativas. Una de ellas es contratar servicios pagos a través de internet con compañías como ser Entrust, Verisign, RSA Security, las cuales son entidades de confianza por defecto en los exploradores web. Otra de las maneras de implementar el uso de PKI es configurar una PKI manualmente, para esto existen distintos enfoques pero nos concentramos en tres de ellos: OpenSSL, OpenCA y EJBCA.

2.4.1- OpenSSL

OpenSSL es una librería de software open-source que implementa funciones y numerosas utilidades criptográficas como ser las de criptografía simétrica y asimétrica, firma digital, funciones de hash, etc.
Esta librería es la más utilizada en lo que respecta a implementación de PKI ya que provee maneras de manipular certificados, solicitudes, claves, etc. OpenSSL implementa los siguientes estándares:

- *RFC 1319: The MD2 Message-Digest Algorithm*
- *RFC 1320: The MD4 Message-Digest Algorithm*
- *RFC 1321: The MD5 Message-Digest Algorithm*
- *RFC 2246: The TLS Protocol Version 1*
- *RFC 2268: A Description of the RC2(r) Encryption Algorithm*
- *RFC 2315: PKCS 7: Cryptographic Message Syntax Version 1.5*
- *RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*
- *RFC 2898: PKCS #5: Password-Based Cryptography Specification Version 2.0*
- *RFC 2986: PKCS #10: Certification Request Syntax Specification Version 1.7*
- *RFC 3161: Internet X.509 Public Key Infrastructure, Time-Stamp Protocol (TSP)*
- *RFC 3174: US Secure Hash Algorithm 1 (SHA1)*
- *RFC 3268: Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)*
- *RFC 3279: Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*
- *RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

- *RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*
- *RFC 3713: A Description of the Camellia Encryption Algorithm*
- *RFC 3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*
- *RFC 4132: Addition of Camellia Cipher Suites to Transport Layer Security (TLS)*
- *RFC 4162: Addition of SEED Cipher Suites to Transport Layer Security (TLS)*
- *RFC 4269: The SEED Encryption Algorithm*
- *PKCS#11: Standards for Cryptographic Tokens*
- *RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1*
- *RFC 5208: PKCS#8: Private-Key Information Syntax Specification Version 1.2*
- *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*
- *RFC 6962: Certificate Transparency*
- *RFC 7292: PKCS #12: Personal Information Exchange Syntax v1.1*

2.4.1.1- Ejemplo de PKI simple utilizando OpenSSL

Para este ejemplo utilizaremos una organización ficticia llamada SimpleOrg con dominio www.simple.org, dicha organización utilizará PKI para securizar sus emails y su tráfico de intranet.

2.4.1.1.1- Creación, instalación y configuración de la infraestructura

Estructura de las CA:

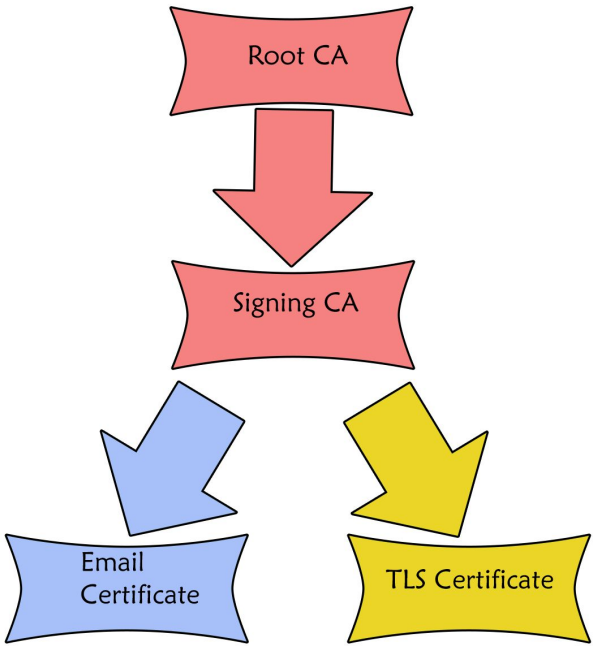


Figura 5. Diagrama de las CA para el ejemplo de PKI simple.

Para este ejemplo utilizaremos una jerarquía de CA (*Figura 4*) en la cual poseemos una CA raíz y una CA firmante la cual generará certificados de email y certificados TLS para el servidor web.

Estructura de directorios:

Para la creación de la PKI utilizaremos una estructura de directorios dividida de la siguiente manera:

```
.
├── ca
│   ├── root-ca
│   │   ├── 01.pem
│   │   ├── 02.pem
│   │   ├── db
│   │   │   ├── root-ca.crl.srl
│   │   │   ├── root-ca.crt.srl
│   │   │   ├── root-ca.db
│   │   │   └── root-ca.db.attr
│   │   └── private
│   │       └── root-ca.key
│   ├── root-ca.crt
│   ├── root-ca.csr
│   ├── signing-ca
│   │   ├── 01.pem
│   │   ├── 02.pem
│   │   ├── db
│   │   │   ├── signing-ca.crl.srl
│   │   │   ├── signing-ca.crt.srl
│   │   │   ├── signing-ca.db
│   │   │   └── signing-ca.db.attr
│   │   └── private
│   │       └── signing-ca.key
│   ├── signing-ca.crt
│   └── signing-ca.csr
├── certs
│   ├── pedro.crt
│   ├── pedro.csr
│   ├── simple.org.crt
│   └── simple.org.csr
├── crl
│   └── signing-ca.crl
└── etc
    ├── email.conf
    ├── root-ca.conf
    ├── server.conf
    └── signing-ca.conf
```

En esta estructura de directorios podremos encontrar cuatro carpetas raíces:

- Dentro de la carpeta `ca` podremos encontrar un directorio por cada CA, cada uno de estos posee:
    - La clave privada de la CA (`*.key`).
    - Una carpeta `db` con archivos que representan a la base de datos de la CA, la cual será explicada en las próximas secciones.
    - La solicitud de firma de certificado de la CA (`*.csr`).
    - El certificado de la CA (`*.crt`).
    - Dentro de la carpeta `private` encontraremos la clave privada de la CA (`*.key`).
    - Una copia de cada certificado firmado por la CA (`*.pem`).
- Dentro de la carpeta `certs` encontraremos todos los certificados firmados por las CA, con los request de cada una de estos.
- En la carpeta `crl` observaremos las listas de revocación de certificados (`*.crl`).
- Dentro de la carpeta `etc` estarán almacenados los archivos de configuración (`*.conf`).

Base de datos - Archivo index:

El archivo index consiste de cero o más líneas , cada una conteniendo los siguientes campos separados por tab.

1. Bandera de estado de certificado (V=Valido, R=Revocado, E=Expirado).
2. La fecha de expiración del certificado en formato YYMMDDHHMMSSZt.
3. La fecha de revocación del certificado en formato YYMMDDHHMMSSZ[,Razón]. Vacío si no está revocado.
4. Número serial del certificado en hexadecimal.
5. Nombre de archivo del certificado o el string literal “unknown”(Desconocido).
6. Nombre distinguido del certificado.

El comando openssl ca usa este archivo como base de datos de certificados.

Ejemplo:

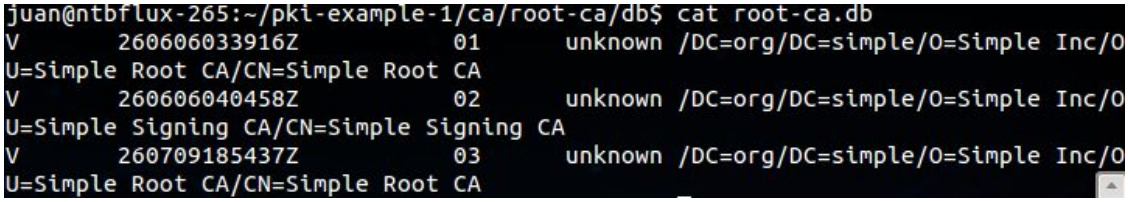


Figura 6. Certificados actualmente en la base de datos root-ca.db, correspondiente a la Root CA utilizada en el ejemplo.

Base de datos - Archivo de atributos:

El archivo de atributos contiene una sola línea: "unique_subject=no". Refleja los ajustes en la sección CA del archivo de configuración, en el momento en el que el primer registro es agregado a la base de datos.

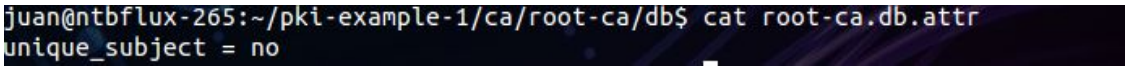


Figura 7. Archivo root-ca.db.attr correspondiente a la Root CA utilizada en el ejemplo.

Base de datos - Archivos de números de serie:

El comando openssl ca usa dos archivos de números de serie:

1. El archivo de numero de serie del certificado.
2. El archivo de numero de serie CRL.

Estos archivos contienen los siguientes números seriales disponibles en hexadecimal.

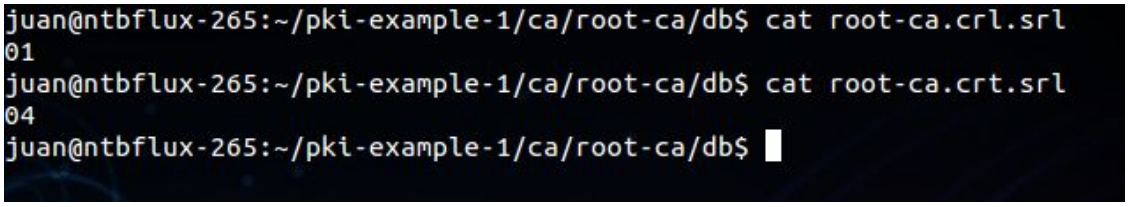


Figura 8. Contenido del archivo serial crl y crt respectivamente de la Root CA de este ejemplo.

Creación de la CA Raíz:

Para la creación de la CA raíz es necesario indicar un archivo de configuración para los comandos de openssl, crear la estructura de directorios, y crear una base de datos en donde se almacenará la información de la CA, por último se deberá generar el certificado de la CA.

El archivo de configuración es una parte importante de la creación de la CA ya que este será la referencia para todas las operaciones a realizar sobre la CA. El archivo utilizado para este ejemplo es el siguiente:

```
[ default ]
ca                      = root-ca               # Nombre de la CA
dir                     = .                     # Directorio raíz
[ req ]
default_bits            = 2048                  # Tamaño de la clave RSA
encrypt_key             = yes                   # Encripta la clave privada
default_md              = sha1                  # Algoritmo de encriptación utilizado
utf8                    = yes                   # Utilizar UTF-8
string_mask             = utf8only              # Emite palabras en UTF-8
prompt                  = no                    # No preguntar por DN (Distinguish Name)
distinguished_name      = ca_dn                 # Sección DN
req_extensions          = ca_reqext             # Extensiones requeridas
[ ca_dn ]                                       # Configuración del dominio
0.domainComponent       = "org"
1.domainComponent       = "simple"
organizationName        = "Simple Inc"
organizationalUnitName  = "Simple Root CA"
commonName              = "Simple Root CA"
[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
[ ca ]
default_ca              = root_ca                       # La CA por defecto
[ root_ca ]
certificate             = $dir/ca/$ca.crt               # Archivo del certificado de la CA
private_key             = $dir/ca/$ca/private/$ca.key # Clave privada de la CA
new_certs_dir           = $dir/ca/$ca                   # Directorio de certificados
serial                  = $dir/ca/$ca/db/$ca.crt.srl  # Archivo de número de serie
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl  # Número de archivo de CRL
database                = $dir/ca/$ca/db/$ca.db # Base de datos
unique_subject          = no                    # Requerir único sujeto
default_days            = 3652                  # Duración del certificado
default_md              = sha1                  # Algoritmo de encriptación utilizado
policy                  = match_pol             # Política de nombramiento
email_in_dn             = no                    # Agregar mail al DN (Distinguish Name)
preserve                = no
name_opt                = ca_default
cert_opt                = ca_default            # Opciones para el certificado
copy_extensions         = none                  # Copiar extensiones del CSR
x509_extensions         = signing_ca_ext        # Extensiones por defecto para el cert.
default_crl_days        = 365                   # Tiempo de vida de las CRL
```

```
crl_extensions          = crl_ext               # Extensiones CRL
# Las políticas de nombramiento controlan que partes del nombre de dominio terminan en el
certificado y bajo qué circunstancias la certificación deberia ser denegada.
[ match_pol ]
domainComponent         = match                 # El dominio debe ser 'simple.org'
organizationName        = match                 # El nombre debe ser 'Simple Inc'
organizationalUnitName  = optional
commonName              = supplied
[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional
# Las extensiones de certificado definen qué tipos de certificados la CA es capaz de crear.
[ root_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
[ signing_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
[ crl_ext ]
authorityKeyIdentifier  = keyid:always
```

El siguiente paso es crear la estructura de directorios, para esto se ejecuta el comando de bash para la creación de los directorios, y un segundo comando para la configuración de permisos:

```
mkdir -p ca/root-ca/private ca/root-ca/db crl certs
chmod 700 ca/root-ca/private
```

Luego es necesario configurar una base de datos para almacenar la información de los certificados. Para este ejemplo utilizaremos un simple archivo de texto como base de datos. Este archivo es el llamado “Index File”. Para esto ejecutamos:

```
cp /dev/null ca/root-ca/db/root-ca.db
cp /dev/null ca/root-ca/db/root-ca.db.attr
```

```
echo 01 > ca/root-ca/db/root-ca.crt.srl
echo 01 > ca/root-ca/db/root-ca.crl.srl
```

Por último es necesario crear un certificado para la CA. Para crear un certificado para la CA primero es necesario realizar el pedido de certificado el cual luego va a ser firmado por la misma CA (auto-firmado). Para esto ejecutamos:

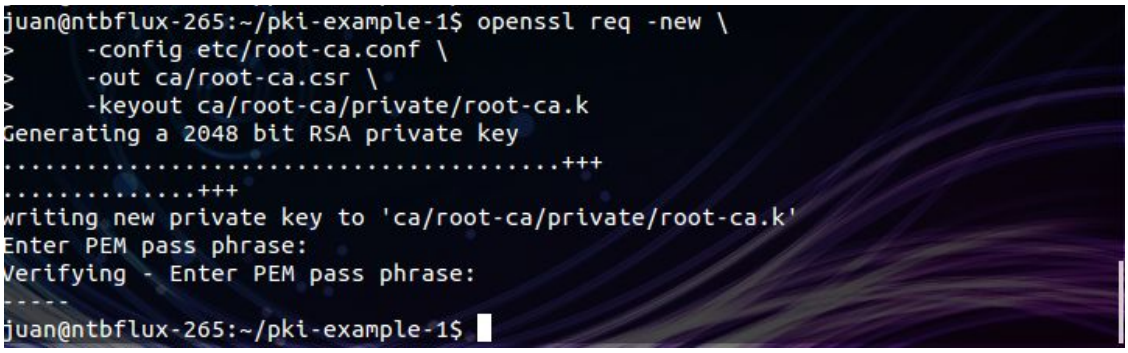


Figura 9. Creación de la solicitud de certificado de la CA

Este comando creará una clave privada y un requerimiento de firma de certificado (CSR) para el CA raíz en función a la configuración de la sección req del archivo de configuración de la CA. Al ejecutar este comando se nos pedirá una frase para proteger la clave privada.

Una vez listo el pedido de certificado, ahora es necesario auto firmarlo. Para esto ejecutamos:

```
openssl ca -selfsign -config etc/root-ca.conf  -in ca/root-ca.csr  -out
ca/root-ca.crt -extensions root_ca_ext
```

Así creamos un certificado para la CA raíz basada en el CSR anterior. El certificado raíz es autofirmado y sirve como punto de partida para todas las relaciones de confianza dentro de la PKI. El comando toma su configuración de la sección *[ca]* del archivo de configuración de la CA.

Creación de la CA firmante:

La creación de la CA firmante se realiza de una manera muy similar a la de la CA raíz.

Se ejecutan los siguientes comandos para crear directorios y cambiar permisos:

```
mkdir -p ca/signing-ca/private ca/signing-ca/db crl certs
chmod 700 ca/signing-ca/private
```

Luego se crea la base de datos:

```
cp /dev/null ca/signing-ca/db/signing-ca.db
cp /dev/null ca/signing-ca/db/signing-ca.db.attr
echo 01 > ca/signing-ca/db/signing-ca.crt.srl
echo 01 > ca/signing-ca/db/signing-ca.crl.srl
```

Se crea el requerimiento de certificado para la CA

```
openssl req -new -config etc/signing-ca.conf -out ca/signing-ca.csr -keyout
ca/signing-ca/private/signing-ca.key
```

Una vez listo el pedido de certificado, esta vez se lo firma con la CA raíz:

```
openssl ca  -config etc/root-ca.conf  -in ca/signing-ca.csr  -out ca/signing-ca.crt
-extensions signing_ca_ext
```

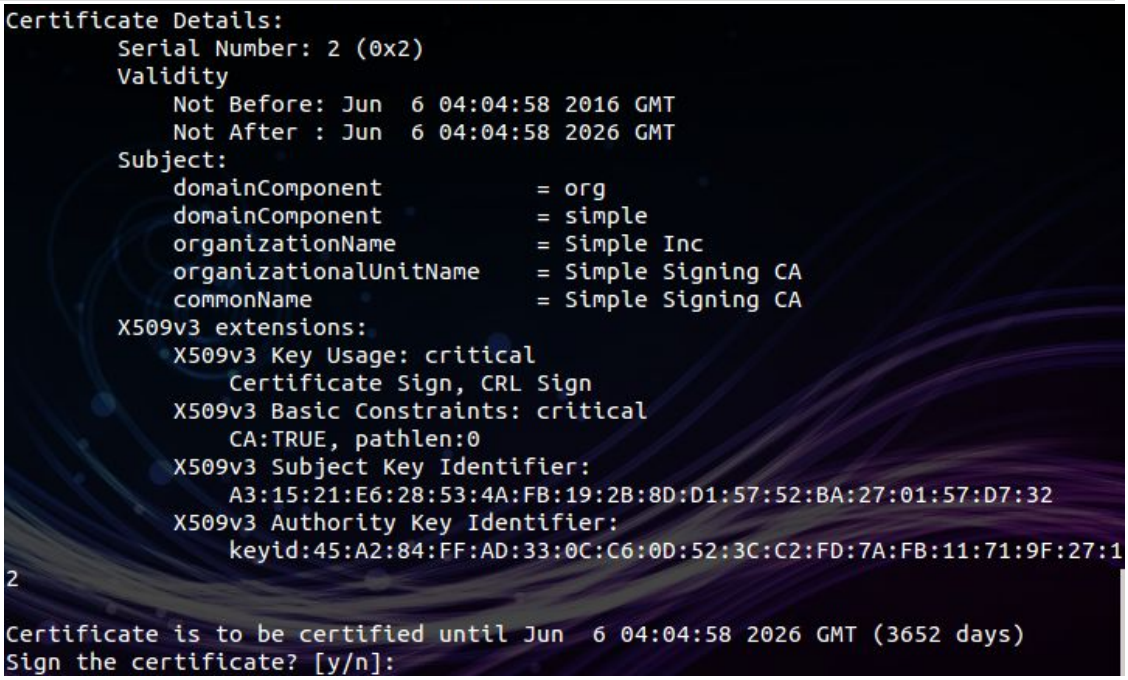


Figura 10. Resultado de la firma del certificado de la CA.

A continuación se indica el archivo de configuración de la CA firmante:

```
[ default ]
ca                      = signing-ca            # CA name
dir                     = .                     # Top dir
[ req ]
default_bits            = 2048                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha1                  # MD to use
utf8                    = yes                   # Input is UTF-8
```

```
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions
[ ca_dn ]
0.domainComponent       = "org"
1.domainComponent       = "simple"
organizationName        = "Simple Inc"
organizationalUnitName  = "Simple Signing CA"
commonName              = "Simple Signing CA"
[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash


[ ca ]
default_ca              = signing_ca            # The default CA section
[ signing_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 730                   # How long to certify for
default_md              = sha1                  # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = ca_default            # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = email_ext             # Default cert extensions
default_crl_days        = 7                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions
[ match_pol ]
domainComponent         = match                 # Must match 'simple.org'
organizationName        = match                 # Must match 'Simple Inc'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present
[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional
[ email_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
[ server_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
```

```
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
[ crl_ext ]
authorityKeyIdentifier  = keyid:always
```

2.4.1.1.2- Operativa

Crear un certificado para firma de email:

Como usuario primero debemos realizar el pedido de certificado con el siguiente archivo de configuración:

```
[ req ]
default_bits            = 2048                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha1                  # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = email_dn              # DN template
req_extensions          = email_reqext          # Desired extensions

[ email_dn ]
0.domainComponent       = "1. Domain Component         (eg, com)      "
1.domainComponent       = "2. Domain Component         (eg, company)  "
2.domainComponent       = "3. Domain Component         (eg, pki)      "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64
emailAddress            = "7. Email Address            (eg, name@fqdn)"
emailAddress_max        = 40

[ email_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = email:move
```

Con el comando openssl req -new creamos una clave privada y una CSR para un certificado de protección de email. Utilizamos los siguientes datos:

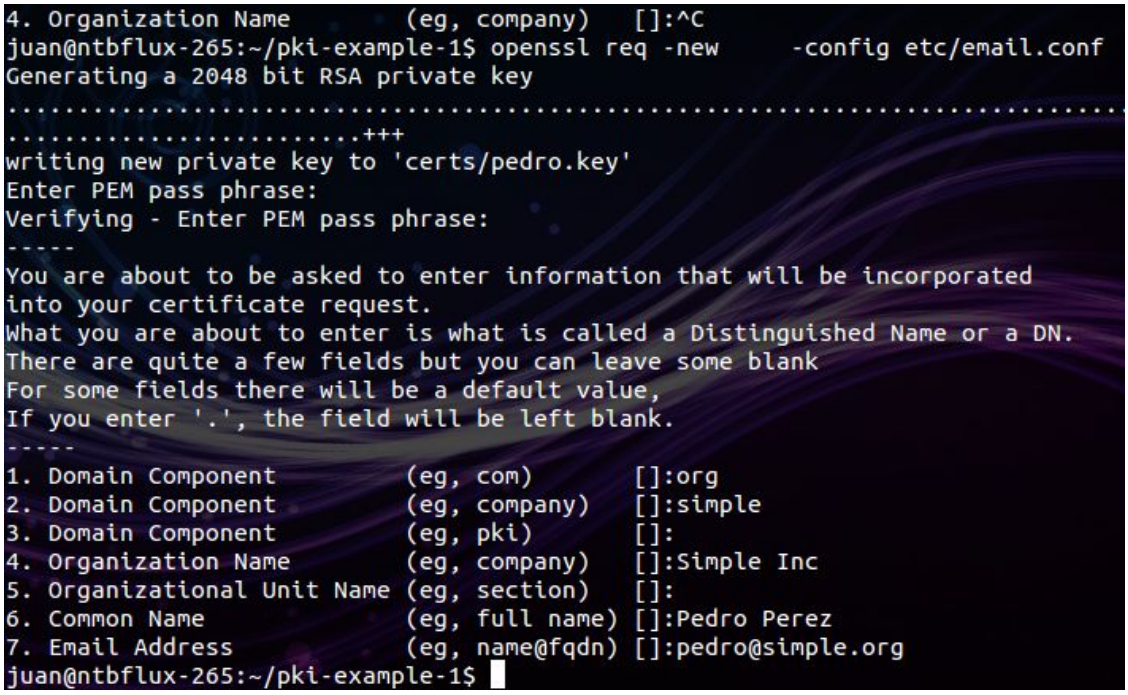


Figura 11. Creación del par de claves.

Luego es necesario generar el certificado a partir de este pedido. Para esto utilizamos la CA firmante para emitir un certificado de protección de email. El tipo de certificado es identificado por la extensión que adjuntamos. Una copia del certificado es guardada en el archivo de certificados bajo el nombre de ca/signing-ca/01.pem. (01 siendo el serial del certificado en hexadecimal). Debemos ejecutar:

```
openssl ca -config etc/signing-ca.conf -in certs/fred.csr -out certs/fred.crt
-extensions email_ext
```

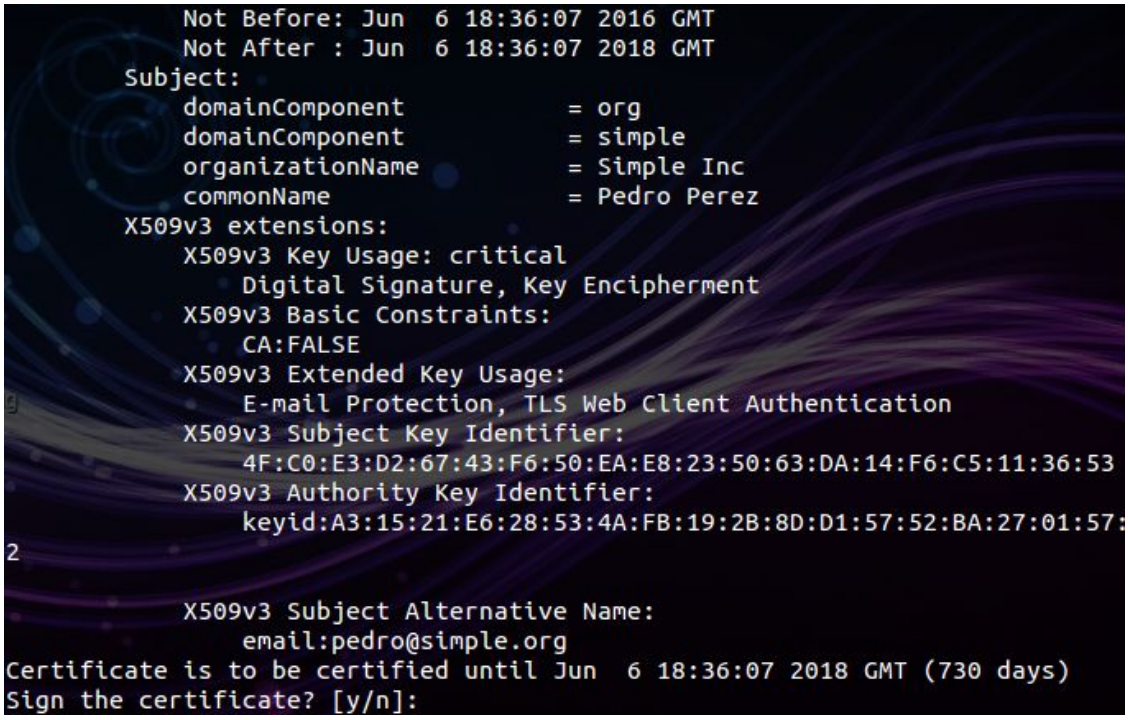


Figura 12. Firma del certificado por la CA.

Crear un certificado para un servidor TLS:

Primero creamos la solicitud con el siguiente archivo de configuración:

```
[ default ]
SAN                     = DNS:yourdomain.tld    # Default value

[ req ]
default_bits            = 2048                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha1                  # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = server_dn             # DN template
req_extensions          = server_reqext         # Desired extensions

[ server_dn ]
0.domainComponent       = "1. Domain Component         (eg, com)      "
1.domainComponent       = "2. Domain Component         (eg, company)  "
2.domainComponent       = "3. Domain Component         (eg, pki)      "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, FQDN)     "
commonName_max          = 64

[ server_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = serverAuth,clientAuth
```

```
subjectKeyIdentifier    = hash
subjectAltName          = $ENV::SAN
```

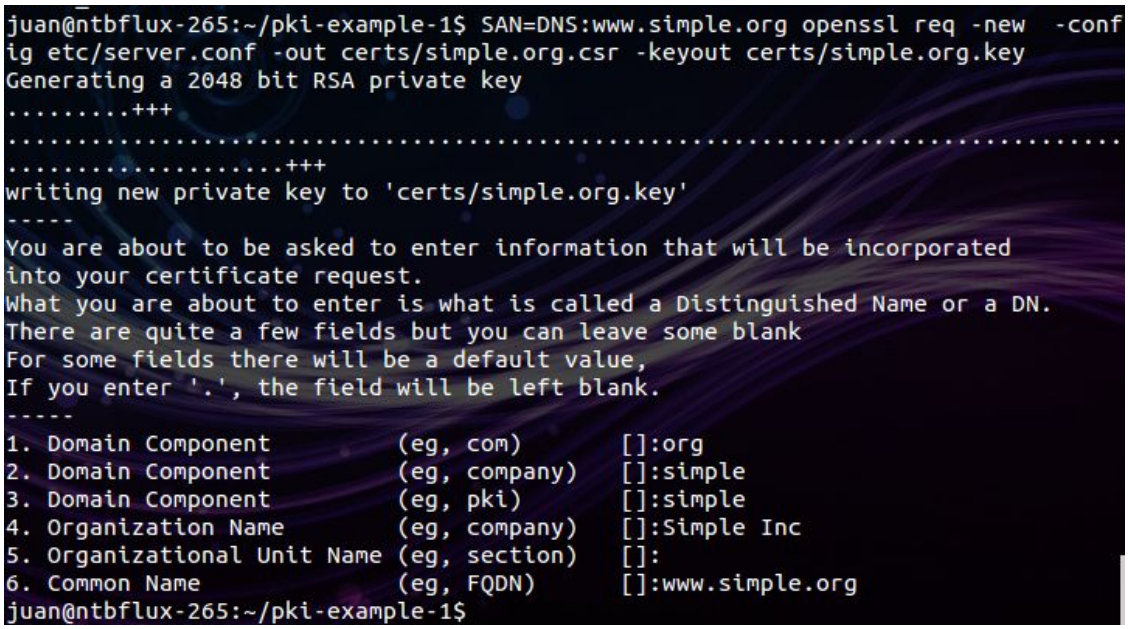


Figura 13. Creación de la solicitud.

Notar que se define SAN (subjectAltName) como variable de entorno. También que generalmente las claves de servidor no tienen frase protectora.

Luego generamos el certificado a partir del pedido:

```
openssl ca -config etc/signing-ca.conf -in certs/simple.org.csr -out
certs/simple.org.crt -extensions server_ext
```

Utilizamos la CA firmante para emitir el certificado del servidor. El tipo del certificado es definido por la extensión adjuntada. Una copia del certificado es almacenado en el directorios de certificados bajo el nombre ca/signing-ca/02.pem

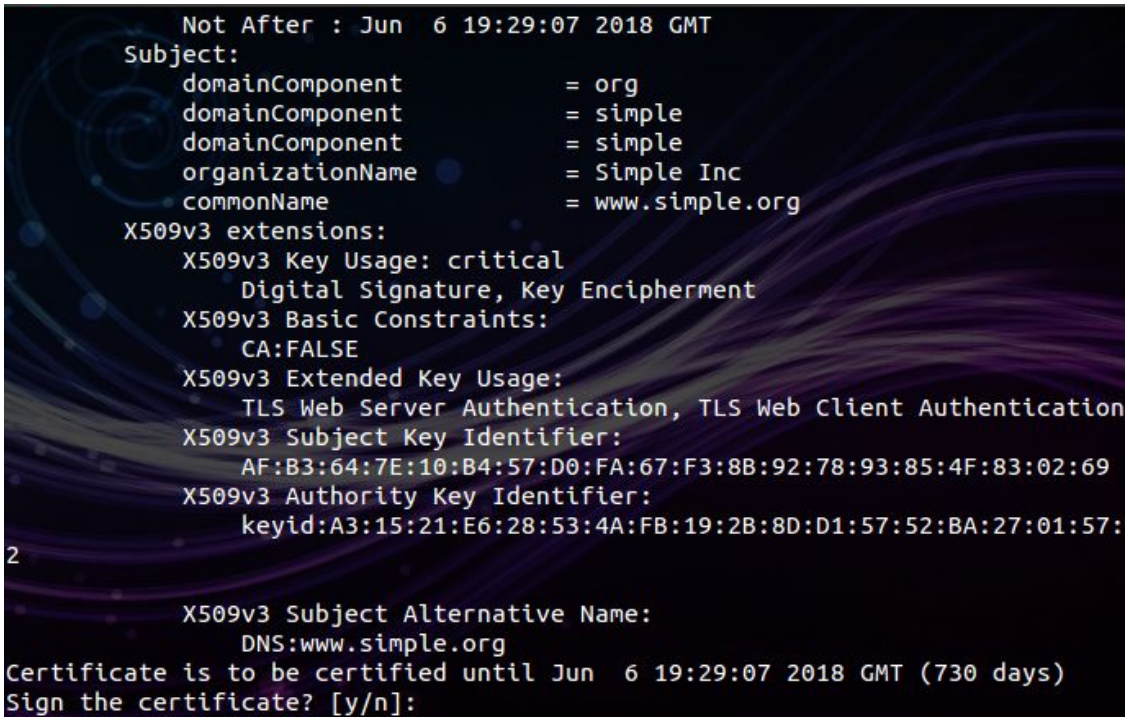


Figura 14. Firma del certificado

Revocar un certificado:

Ciertos eventos, como reemplazo de certificados o pérdida de la clave privada, requieren que un certificado sea revocado, antes de que expire. El comando openssl ca -revoke marca un certificado como revocado en la base de datos de la CA, será desde entonces incluido en la Certificate Revocation List(CRL) emitida por la CA. El comando ejecutado revoca el certificado con el número 01 en hexadecimal

Figura 15. Revocación de un certificado.

Crear una CRL (Certificate Revocation List):

```
juan@ntbflux-265:~/pki-example-1$ openssl ca -gencrl -config etc/signing-ca.conf
  -out crl/signing-ca.crl
Using configuration from etc/signing-ca.conf
Enter pass phrase for ./ca/signing-ca/private/signing-ca.key:
```

Figura 16. Creación de una CRL.

El comando openssl ca -gencrl crea una CRL. La CRL contiene todos los certificados revocados que no hayan expirado de la base de datos de la CA. Una CRL nueva debe ser emitida en intervalos regulares.

2.4.2- OpenCA

OpenCA es un proyecto open-source para implementar y administrar una PKI completa. El proyecto provee una interfaz web desarrollada con el lenguaje de programación *Perl* la cual permite realizar casi todas las operaciones de la PKI a través de esta. Por detrás utiliza OpenSSL y una base de datos en donde se almacena toda la información criptográfica como ser las solicitudes, certificados, CRL, etc.

2.4.2.1- Ejemplo de PKI simple utilizando OpenCA

2.4.2.1.1- Configuración de la infraestructura

La inicialización y configuración se realiza en 3 etapas desde el portal web del OpenCA instalado (asumimos tener OpenCA instalado):

1. Inicialización de la CA.
2. Creación del administrador inicial.
3. Creación del certificado inicial para la RA.

Para empezar ingresamos al portal de administrador y en la sección del menú '*PKI Init & Config*' seleccionamos '*Initialization*', aquí podremos ver las 3 etapas.

Figura 17. Inicialización de OpenCA.

La primer etapa consiste en una serie de cinco tareas para la inicialización de la CA:

1. Inicializar la BD.
2. Generar el par de claves para la CA.
3. Generar una solicitud de certificado para la CA.
4. Auto-firma del certificado.
5. Instalación del certificado.

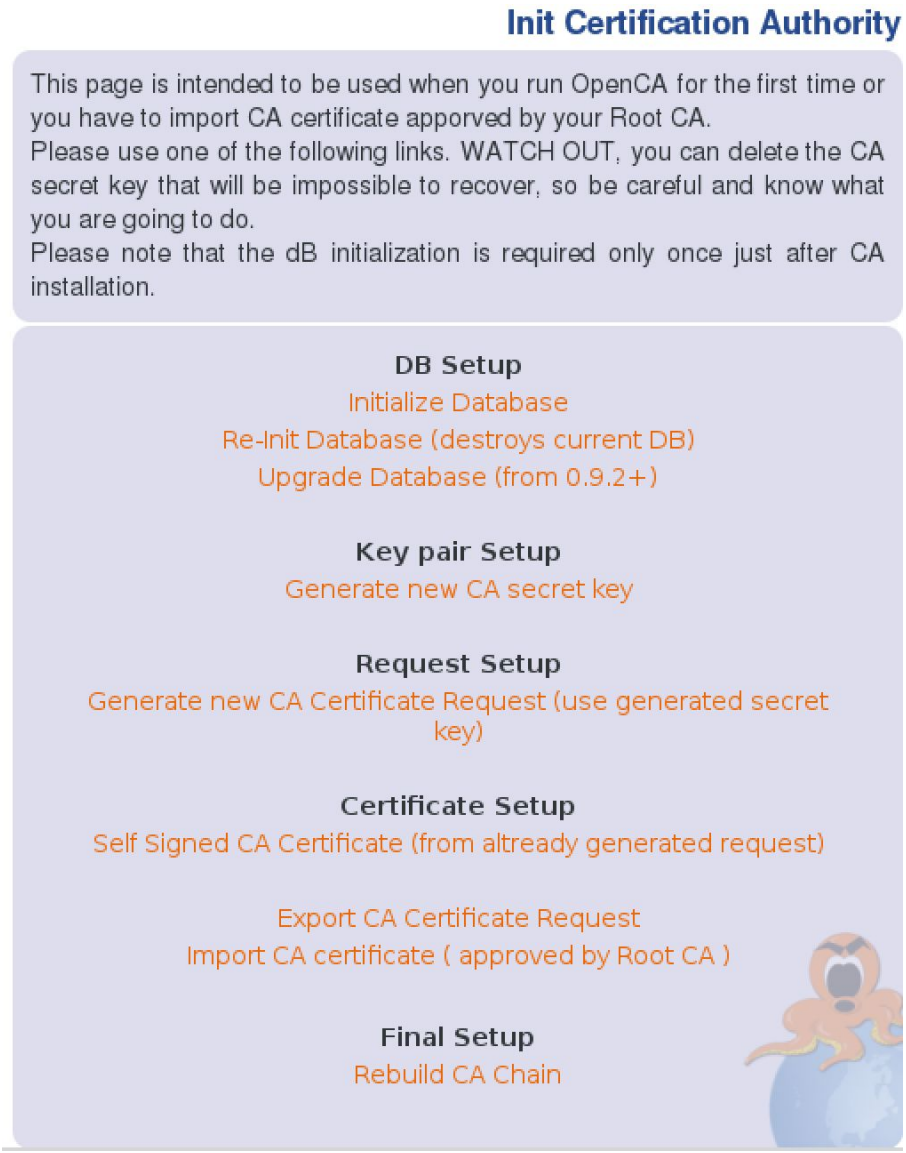


Figura 18. Inicialización de la CA.

La primera de ellas consiste en crear la estructura de las tablas de la base de datos. Esto se realiza automáticamente seleccionando la opción del menú '*Initialize Databas*e'.

El siguiente paso es generar el par de claves para la CA, para esto entramos a '*Generate new CA secret key'* y allí podremos configurar diversas opciones: el algoritmo simétrico de cifrado (AES256, AES128, DES3, IDEA), el algoritmo asimétrico de cifrado (RSA, ECDSA, DSA) y la longitud de la llave en bits. Presionamos 'OK' y obtendremos como resultado la clave privada de la CA.

Figura 19. Generación del par de claves de la CA.

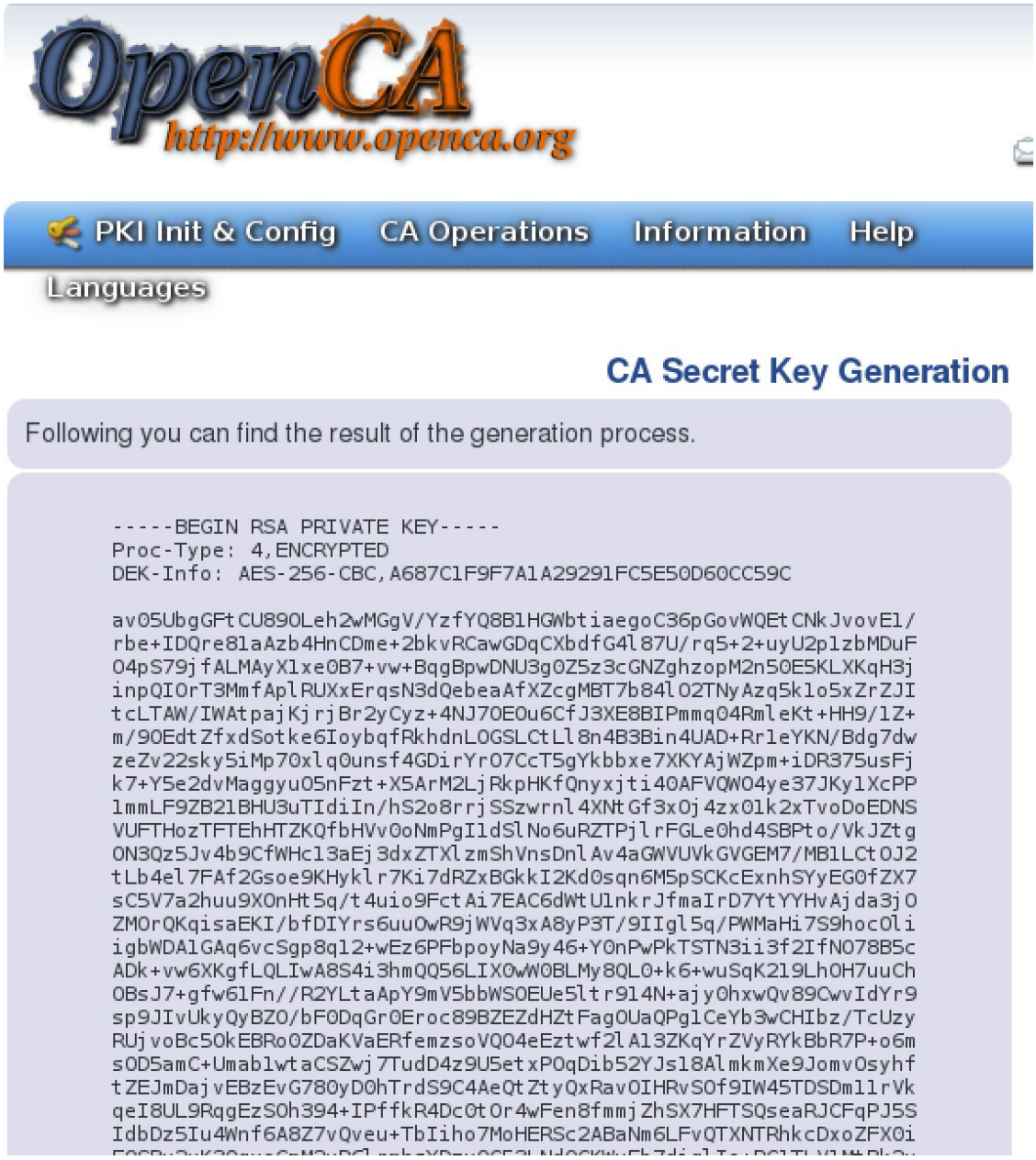


Figura 20. Resultado de la generación del par de claves de la CA.

Para el paso tres generamos una solicitud de certificado para la CA utilizando la clave generada en el paso anterior. Para esto ingresamos en '*Generate new CA Certificate Request (user generated secret key)*' y rellenamos un formulario con el nombre de la CA, del dominio, de la organización y su localidad. Una vez completado el formulario podremos enviar la solicitud.

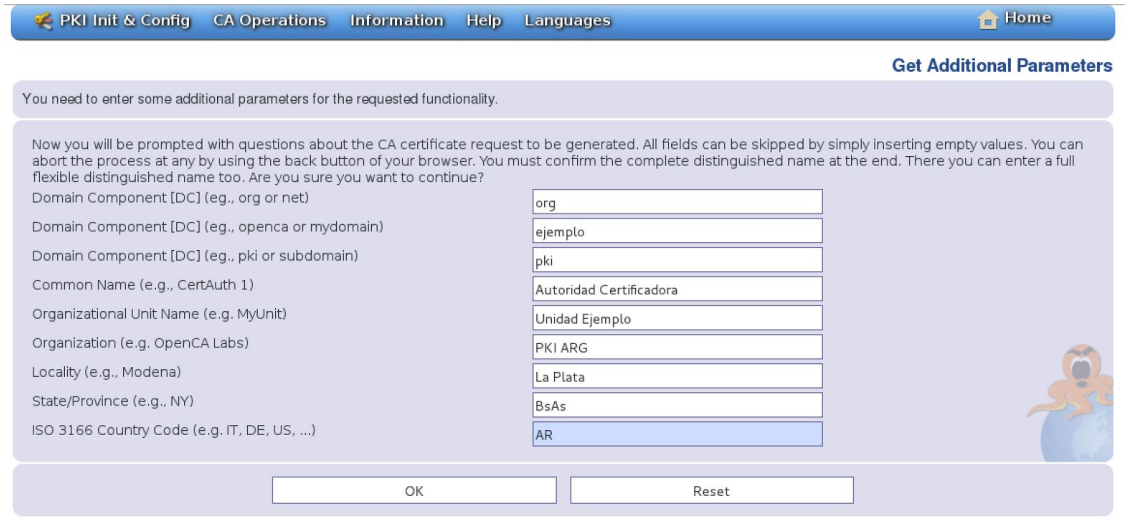


Figura 21. Generación de la solicitud del certificado de la CA.

El paso cuatro consiste en auto-firmar el certificado de la CA, para esto ingresamos en '*Self signed CA certificate (from already generated request)'* y completamos un formulario en donde indicaremos la validez del certificado. Al finalizar tendremos un certificado para la CA auto-firmado listo para utilizar.

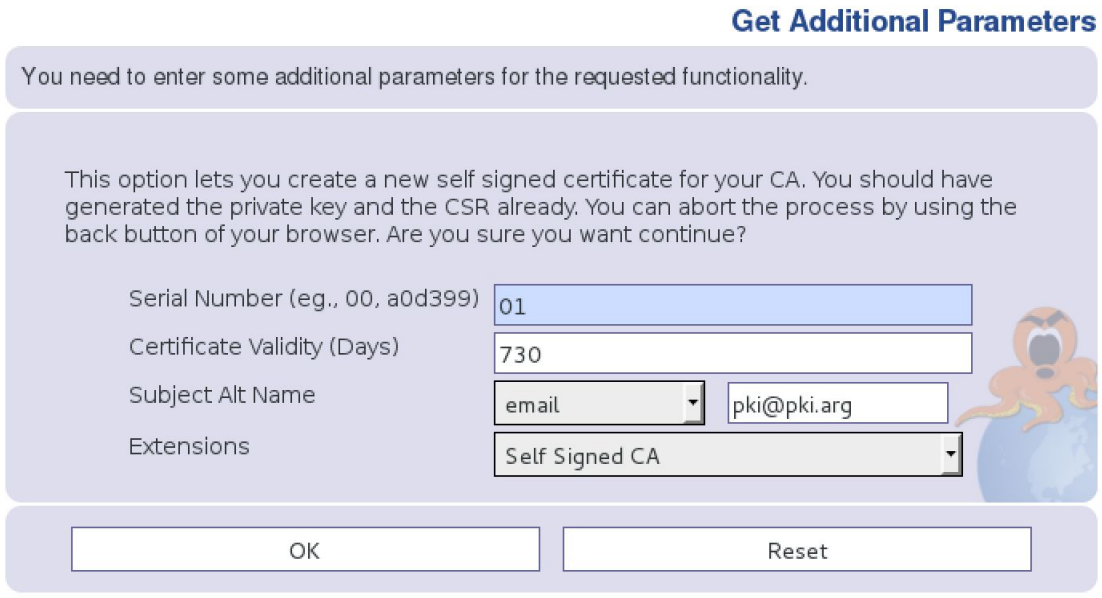


Figura 22. Firma del certificado de la CA.

Como último paso configuraremos el certificado de la CA en apache para poder autenticarnos con el mismo. Para esto copiamos los archivos generados de los certificados y las claves a las rutas correspondientes con los siguientes comandos:

```
cp /opt/openca/var/openca/crypto/keys/cakey.pem  /etc/pki/tls/private/pki.arg.key
cp /opt/openca/var/openca/crypto/cacerts/cacert.txt /etc/pki/tls/certs/pki.arg.crt
```

Luego editamos el archivo de configuración de SSL de apache (‘`/etc/httpd/conf.d/ssl.conf`’) indicando el certificado y la clave de la CA:

```
SSLCertificateFile /etc/pki/tls/certs/pki.arg.crt
SSLCertificateKeyFile /etc/pki/tls/private/pki.arg.key
SSLCACertificateFile /etc/pki/tls/certs/pki.arg.crt
```

Por último reiniciamos el servicio de apache para que tome los nuevos cambios y con esto finaliza la configuración para la primer etapa.

```
systemctls restart httpd.service
```

La segunda etapa consiste en una serie de tres tareas para la creación del administrador inicial de la CA:

1. Crear una nueva solicitud de certificado.
2. Emitir el certificado.
3. Instalar el certificado.

Figura 23. Inicialización del usuario administrador.

Como primer paso debemos crear una solicitud de certificado para el administrador, para esto ingresamos a '*Create a new request'* y completamos una serie de formularios en donde indicaremos el esquema con el cual se generarán el par de claves, un PIN para la generación de claves, una serie de datos personales, y podremos configurar que tipo de plantilla de certificado querremos usar, la RA, así como también la forma y nivel de seguridad para la generación de claves. Al finalizar podremos observar un resumen de todos los datos de la solicitud de certificado enviada.

Figura 24. Generación del par de claves.

Figura 25. CPS de OpenCA.

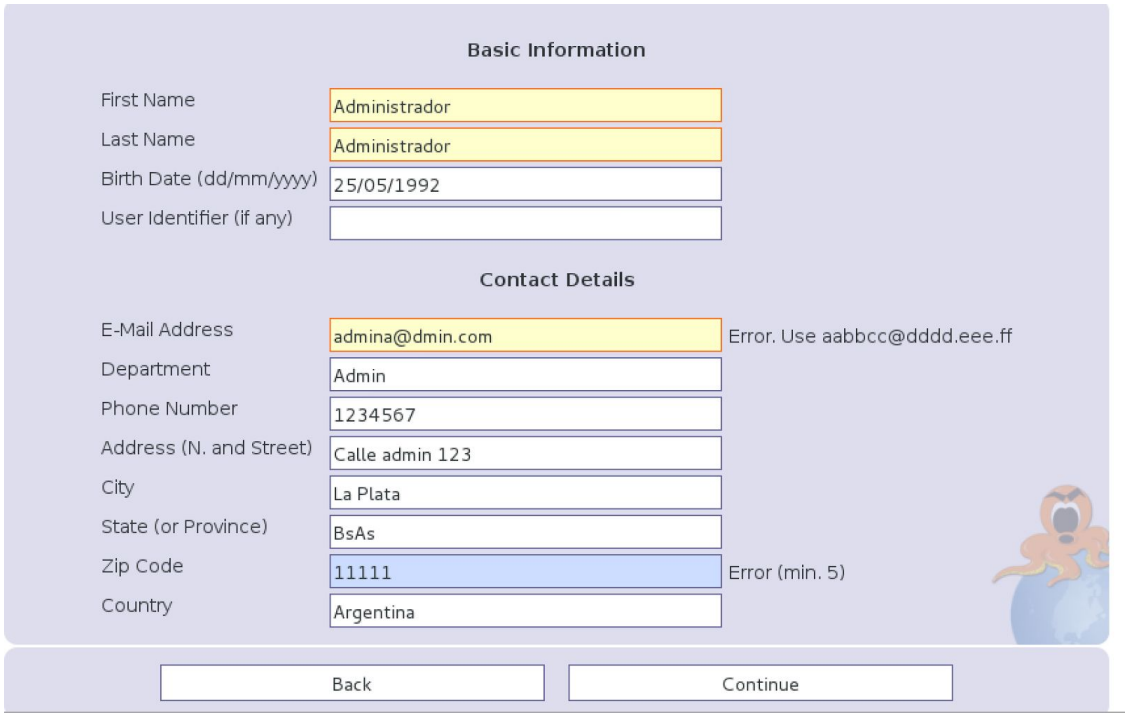


Figura 26. Formulario para la solicitud de certificado del administrador.

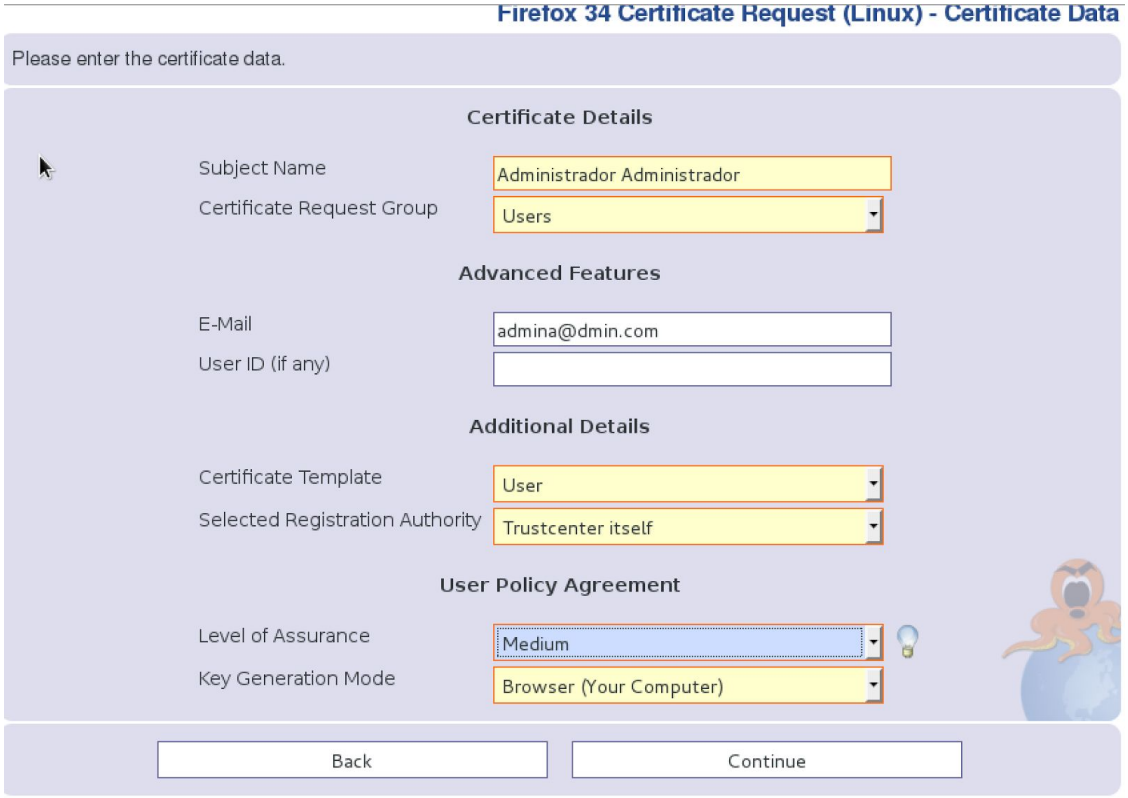


Figura 27. Formulario 2 para la solicitud de certificado del administrador.

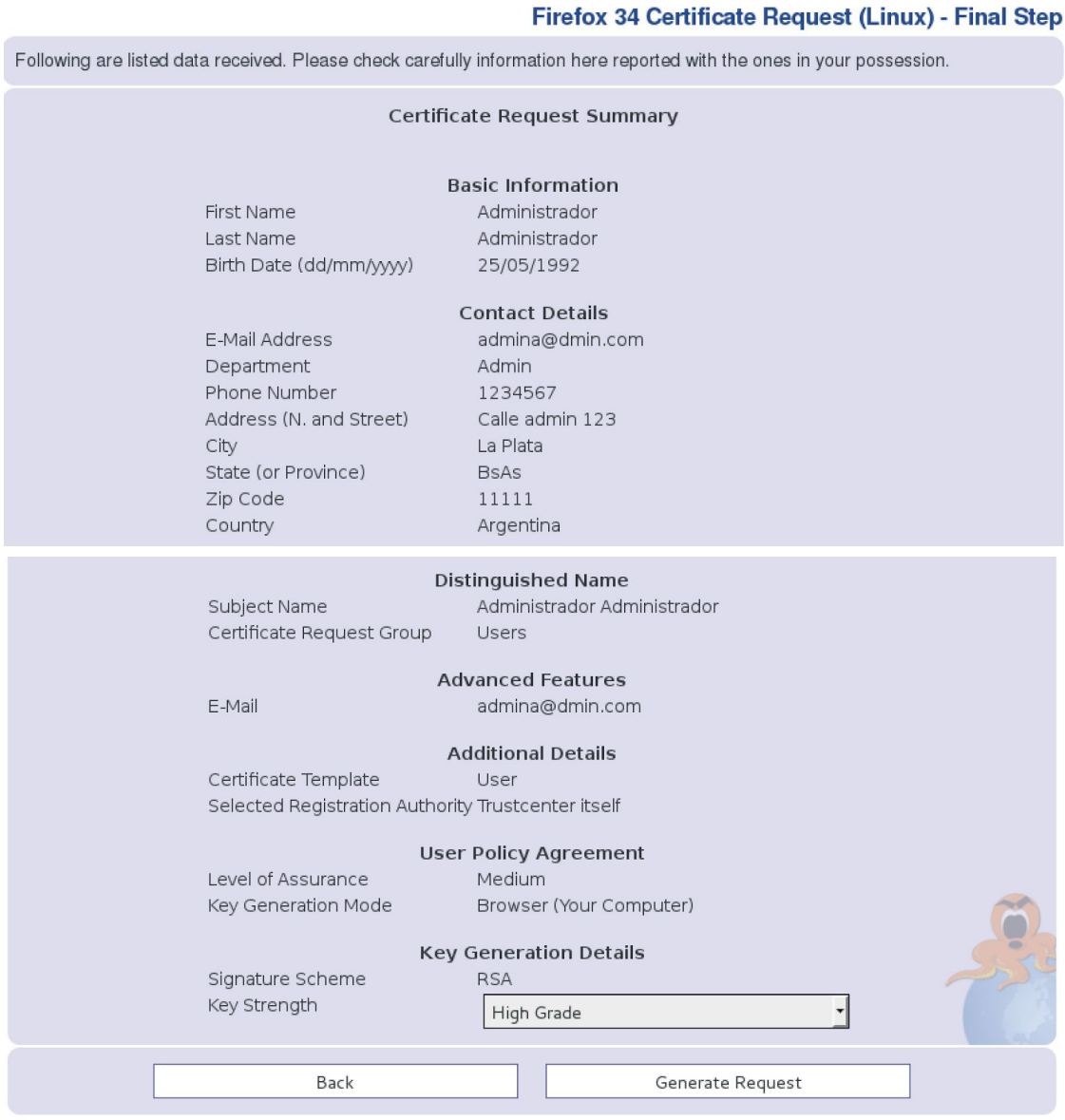


Figura 28. Confirmación de la solicitud de certificado del administrador.

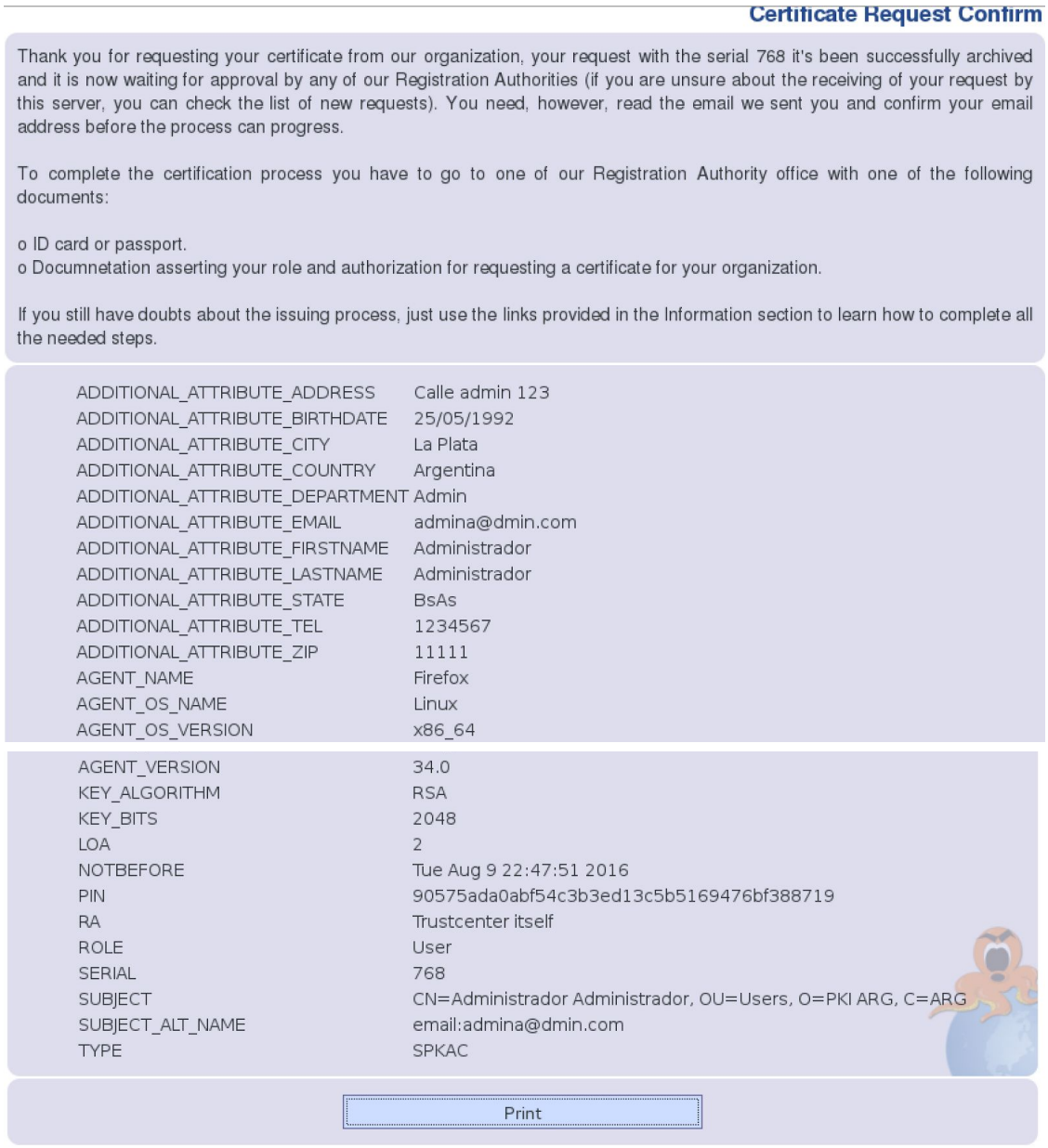


Figura 29. Datos de la solicitud de certificado del administrador.

Para el siguiente paso debemos emitir el certificado utilizando a la CA configurada previamente. Para esto simplemente ingresamos en '*Issue the certificate'* y presionamos el botón '*Issue the certificate'*.

Figura 30. Aceptación de la solicitud de certificado del administrador.

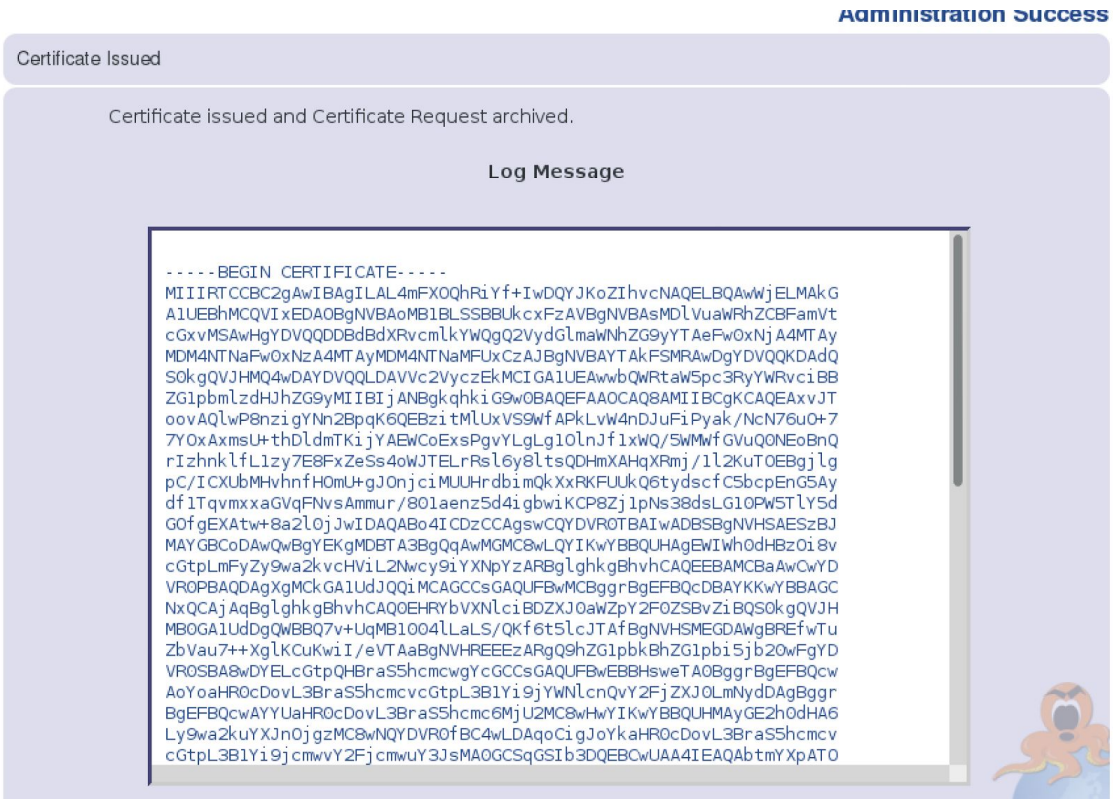


Figura 31. Certificado del administrador en formato texto.

Como último paso solo resta ver e instalar el certificado en nuestro explorador ingresando en la opción del menú '*Handle the certificate'* y luego presionando '*Install the certificate'*. Podemos comprobar la instalación del certificado ingresando en la configuración avanzada de nuestro explorador y buscar una sección de 'Certificados'.

Figura 32. Certificado del administrador en formato gráfico.

Wednesday 10 August 20:39:47 UTC

General Information	
Certificate Version	3 (0x2)
Serial Number	89795215375803198947325O (0xbe:26:15:73:90:85:18:98:7f:e2)
Common Name	Administrador Administrador
E-Mail	admin@admin.com
Distinguished Name	CN=Administrador Administrador OU=Users O=PKI ARG C=AR
Role	User
Fingerprint	SHA1:2D:AB:62:23:8A:EB:37:8A:8C:4A:B4:BB:F2:68:44:72:D0:B1:2D:2C
Issued by	CN=Autoridad Certificadora OU=Unidad Ejemplo O=PKI ARG C=AR
Valid From	Aug 10 20:38:53 2016 GMT
Expiration on	Aug 10 20:38:53 2017 GMT
Current Status	Valid
Extensions	
Authority Information Access	CA Issuers - URI:http://pki.arg/pki/pub/cacert/cacert.crt OCSP - URI:http://pki.arg:2560/ PRQP Service - URI:http://pki.arg:830/
Netscape Cert Type	SSL Client, S/MIME
Netscape Comment	User Certificate of PKI ARG
Authority Key Identifier	keyid:44:7F:04:EE:65:B5:5A:BB:BF:BE:5E:09:4A:0A:E2:B0:88:8F:DE:55
Basic Constraints	CA:FALSE
CRL Distribution Points	Full Name: URI:http://pki.arg/pki/pub/crl/cacrl.crl
Certificate Policies	Policy: 1.2.3.3.4 Policy: 1.2.3.3.5 Policy: 1.2.3.3.6 CPS: https://pki.arg/pki/pub/cps/basic
Extended Key Usage	TLS Web Client Authentication, E-mail Protection, Microsoft Smartcardlogin
Issuer Alternative Name	email:pki@pki.arg
Key Usage	Digital Signature, Non Repudiation, Key Encipherment
Subject Alternative Name	email:admin@admin.com

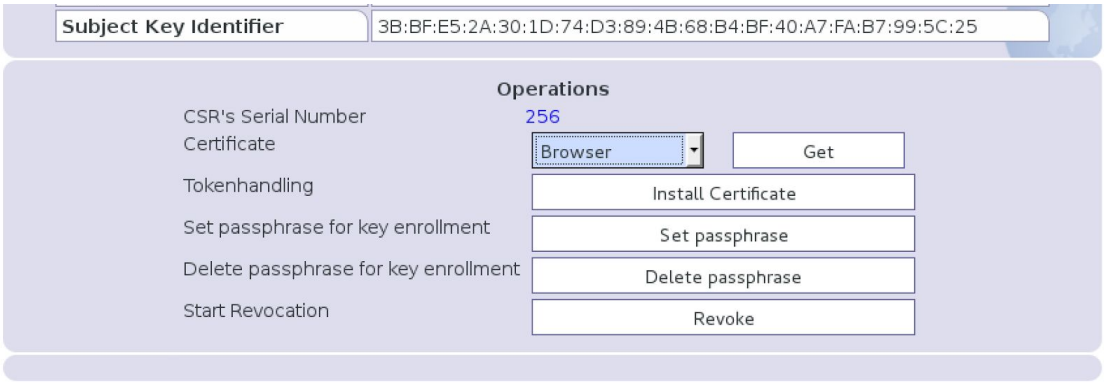


Figura 33. Instalación del certificado del administrador.

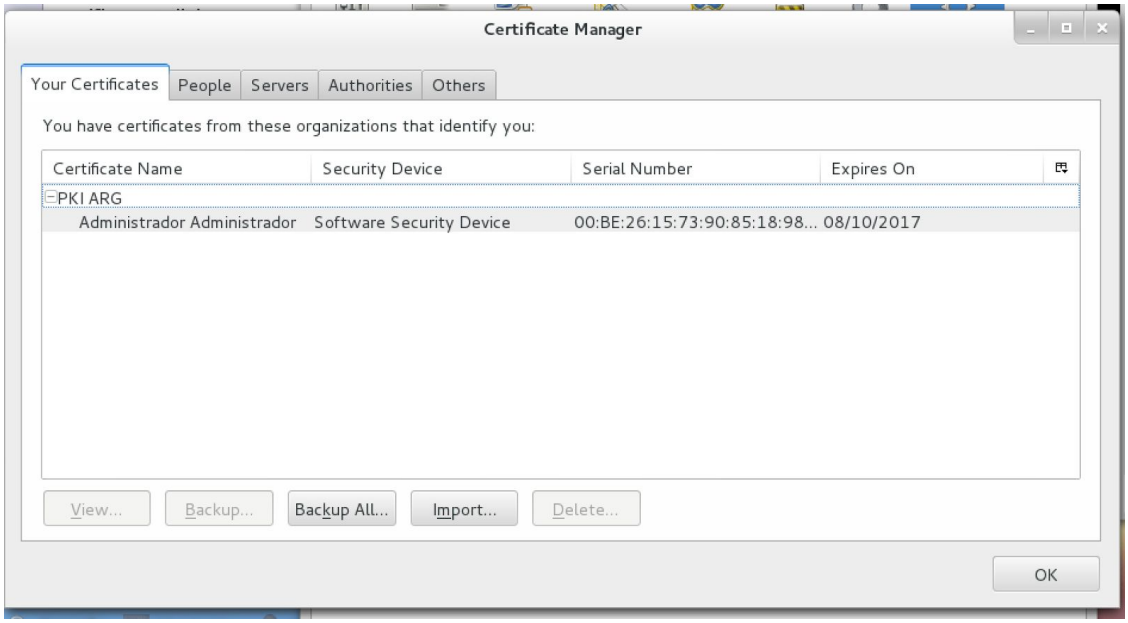


Figura 34. Certificado del administrador instalado en firefox.

2.4.2.1.2- Operativa

Solicitud de un certificado:

Para poder solicitar un certificado como un usuario, entramos en la interfaz web del módulo "*pub*".

Como primer paso debemos instalar el certificado de la CA en nuestro explorador, para esto hacemos click en el botón "*Get CA Certificate*" el cual descarga el certificado de la CA en nuestra computadora, paso siguiente debemos instalarlo en nuestro explorador de la misma manera en la que se expresó en la sección anterior.

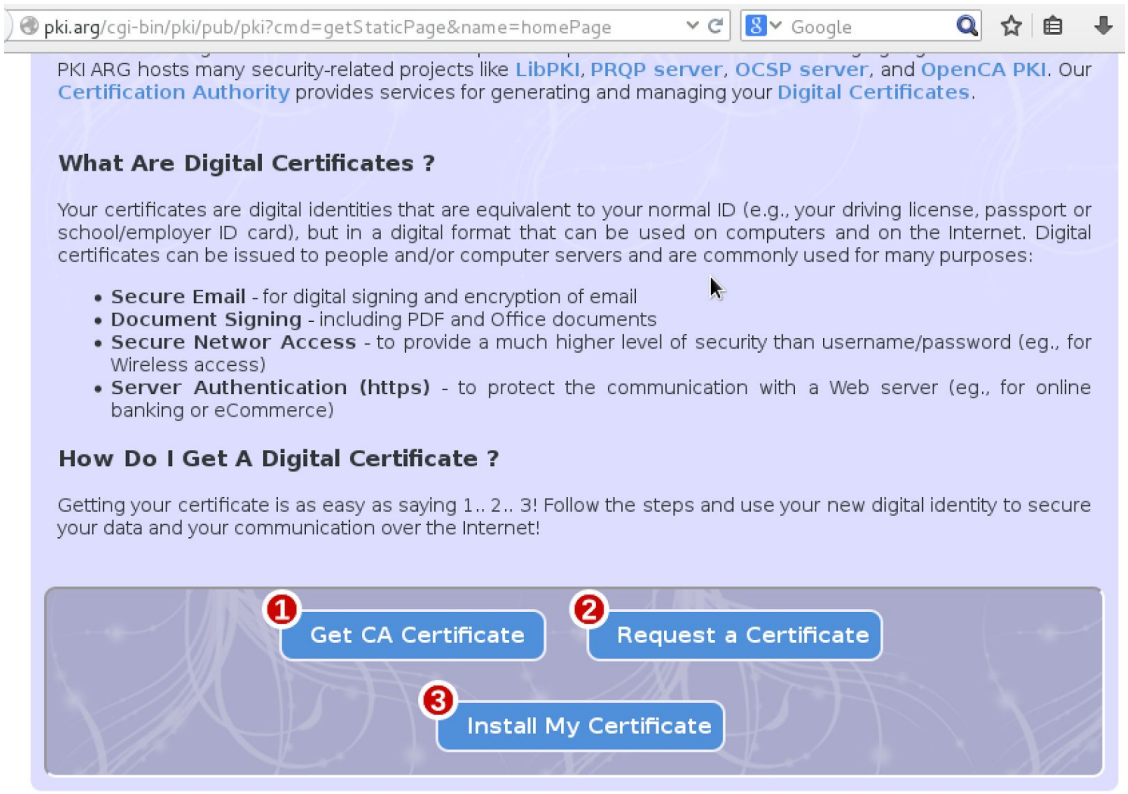


Figura 35. Web pública de OpenCA.

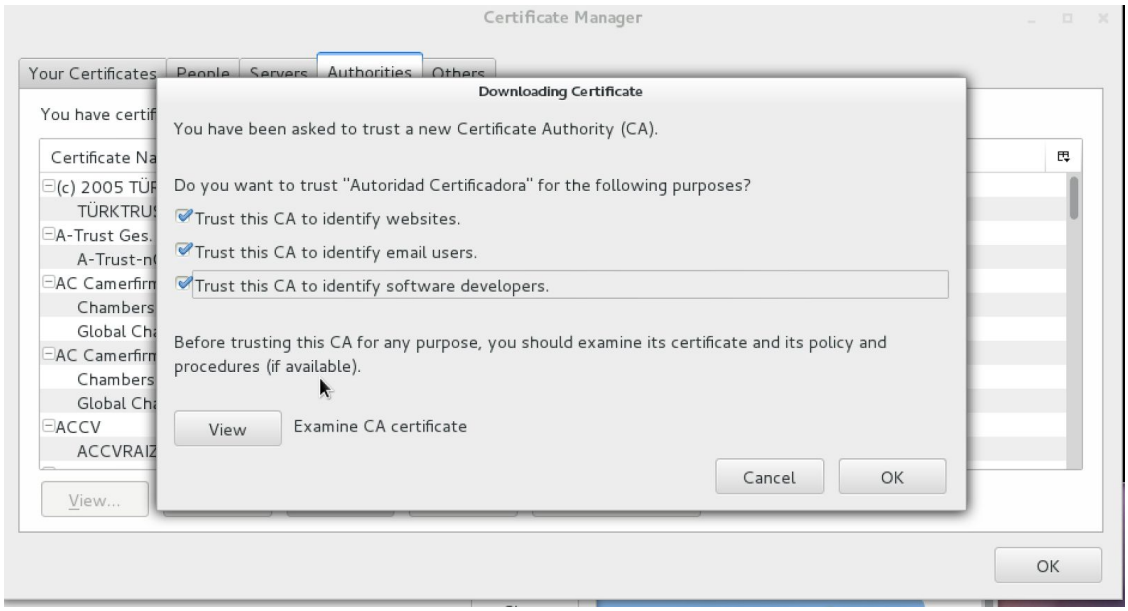


Figura 36. Instalación del certificado de la CA en Firefox.

Como segundo paso hacemos click en el botón "*Request a Certificate"* el cual nos lleva a una nueva pantalla en donde podremos seleccionar el tipo de certificado que queremos. Seleccionamos la primer opción "*Browser Certificate Request*". El proceso a partir de aquí es el mismo que el expresado en las solicitudes de certificados anteriores, se completan los formularios con los datos personales y esto genera una solicitud de certificado que debe ser aprobada por la CA.

Figura 37. Pantalla de requerir certificado.

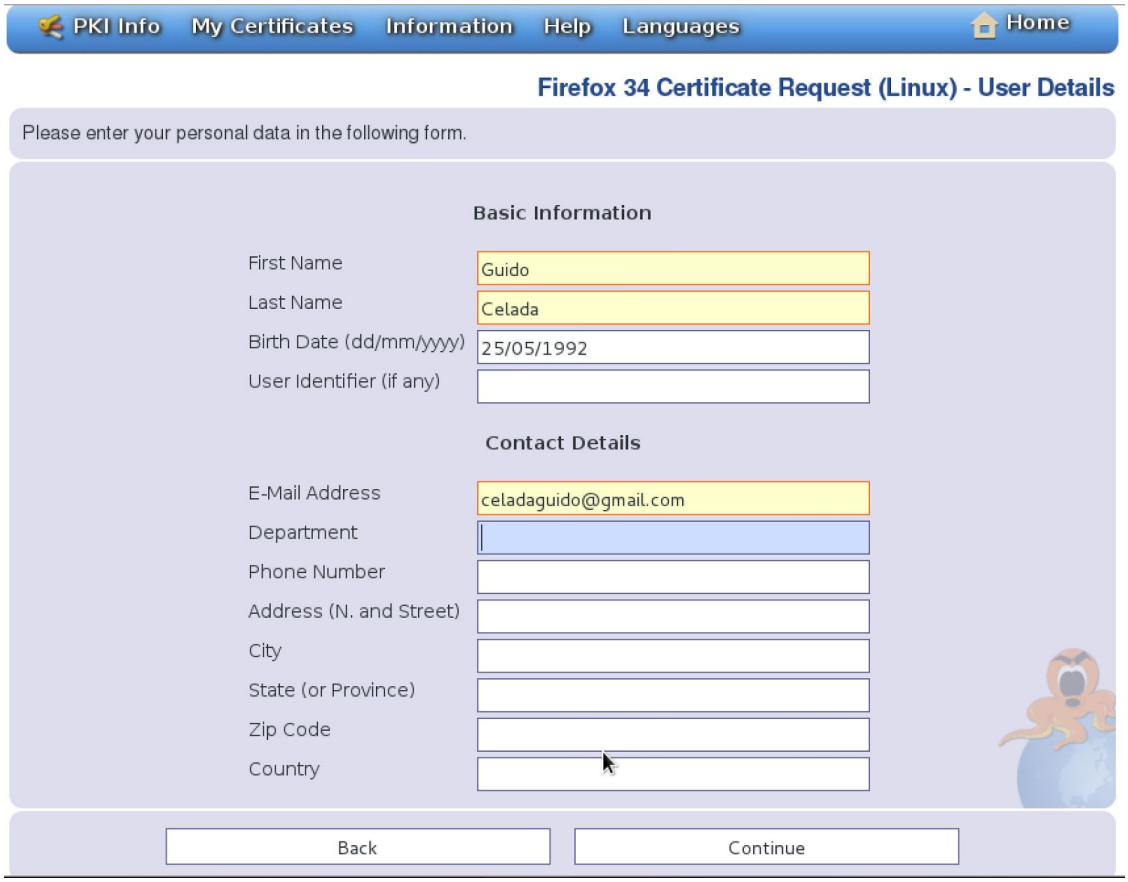


Figura 38. Formulario para requerir un certificado.

Figura 39. Formulario 2 para requerir un certificado.

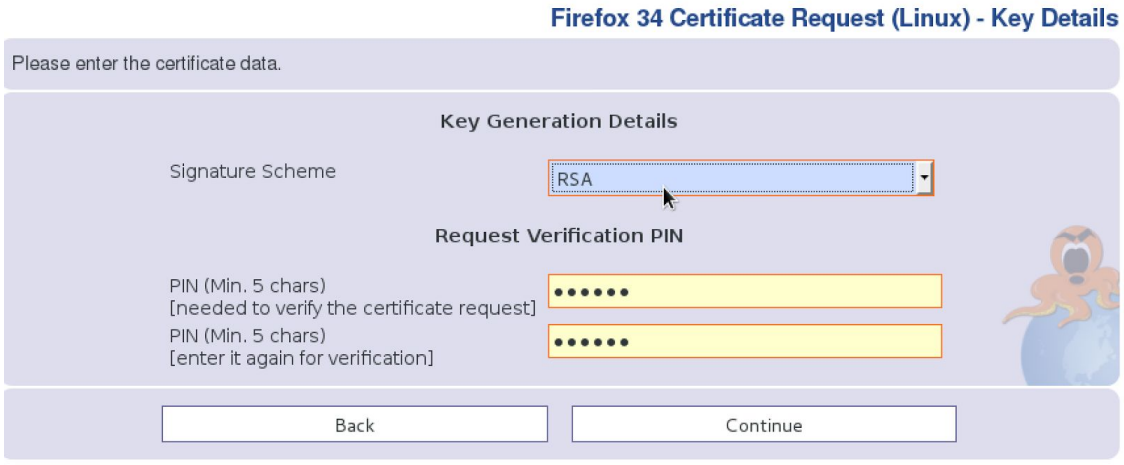


Figura 40. Formulario 3 para requerir certificado.

Figura 41. CPS para requerir un certificado.

Figura 42. Confirmación de la solicitud de certificado.

Figura 43. Datos de la solicitud del certificado realizada.

Una vez realizada la solicitud de certificado, es importante anotar el "*SERIAL*" de la solicitud ya que será utilizado más tarde como referencia para obtener el certificado.

Como tercer paso debemos ingresar a OpenCA como un usuario administrador de la CA al módulo web "*ca*". Una vez ingresado, debemos ingresar en la opción del menú "*CA Operations* > *Certificate Requests* > *New*". Se nos presenta una pantalla con todas las solicitudes de certificados pendientes de aprobación y emisión. Seleccionamos la solicitud del usuario que deseamos haciendo clic en su *Serial*, lo cual nos lleva a otra pantalla con los detalles de la solicitud del usuario, usualmente aquí se realiza un chequeo cara a cara con el usuario de sus datos para comprobar que estos sean verídicos. Una vez comprobado todo, clickeamos en "*Issue certificate*" finalizando el proceso de emisión del certificado.

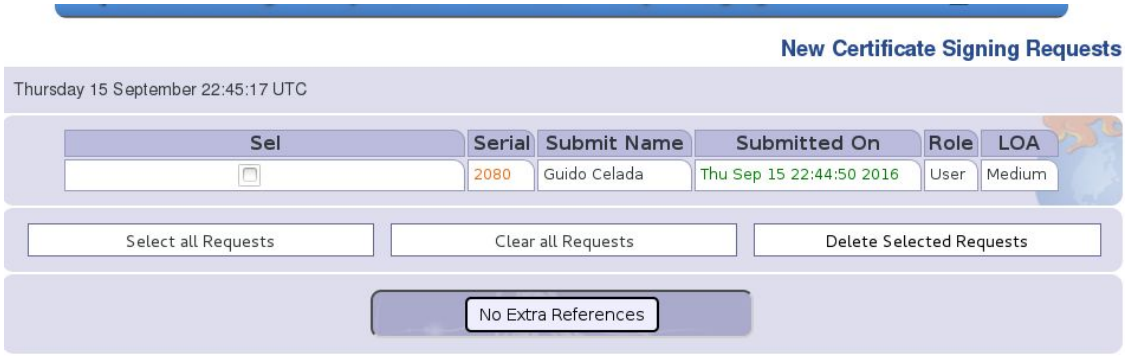


Figura 44. Certificados pendientes en pantalla de administrador.

New Request Waiting for Approval

Following you can find the CSR's details.

Thursday 15 September 22:48:08 UTC

Request Information

Request Version	1
Serial Number	2080
Common Name	Guido Celada
E-Mail	celadaguido@gmail.com [Not Verified]
Subject Alternative Name	email.0=celadaguido@gmail.com
Distinguished Name	CN=Guido Celada OU=Users O=PKI ARG C=AR
Requested Role	User
Requested LOA	Medium
Used Identification PIN	90575ada0abf54c3b3ed13c5b5169476bf388719
Submitted on	Thu Sep 15 22:44:50 2016

Request Status Information

Current Status	NEW
Signed By	Not Signed

Lifetime Information

Lifetime (days)	n/a
Not before (YYMMDDhhmmss)	n/a
Not after (YYMMDDhhmmss)	n/a
Lifetime check	Lifetime would be ok.

Request Technical Info

Request Type	SPKAC

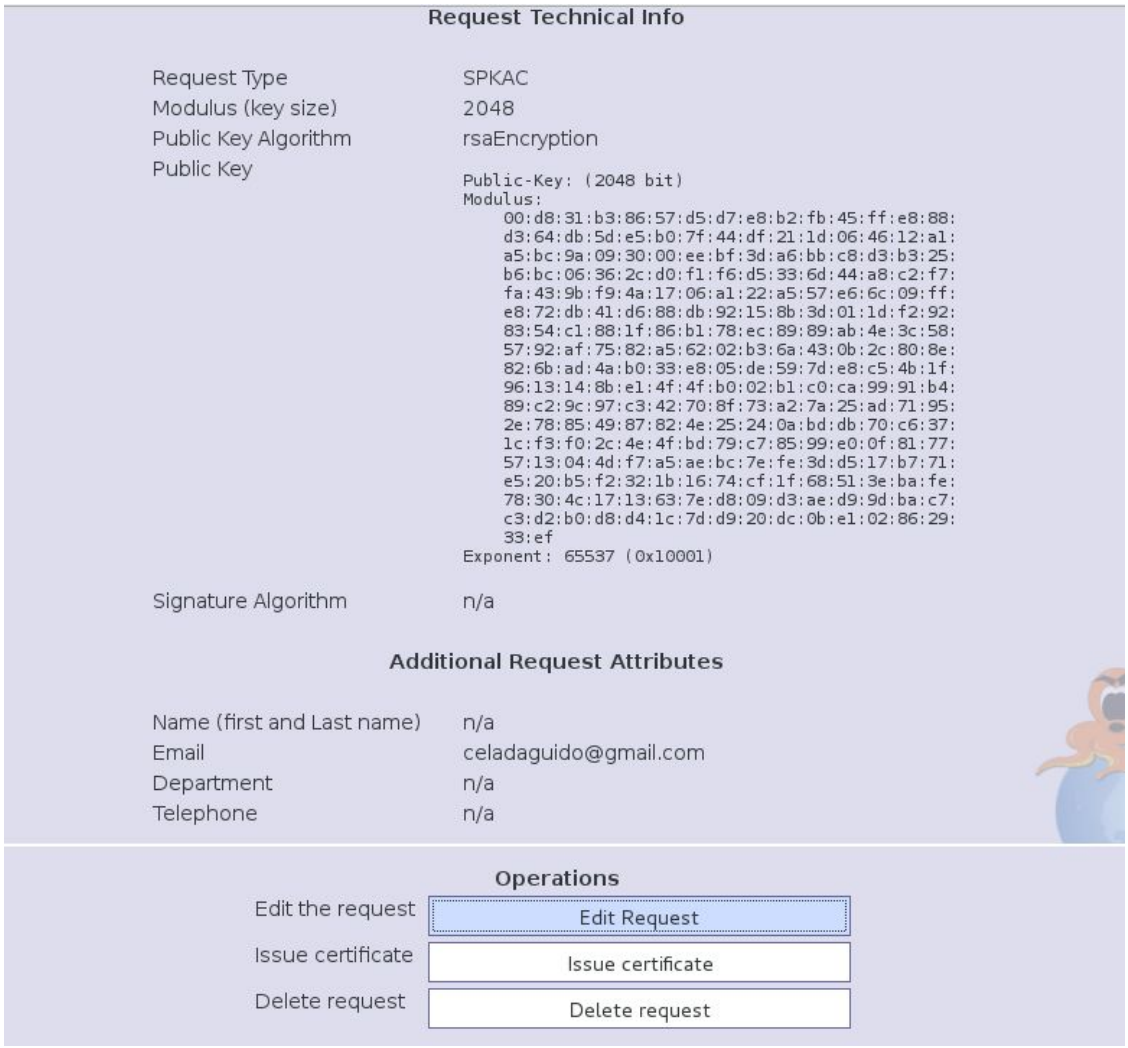


Figura 45. Datos de la solicitud de certificado.

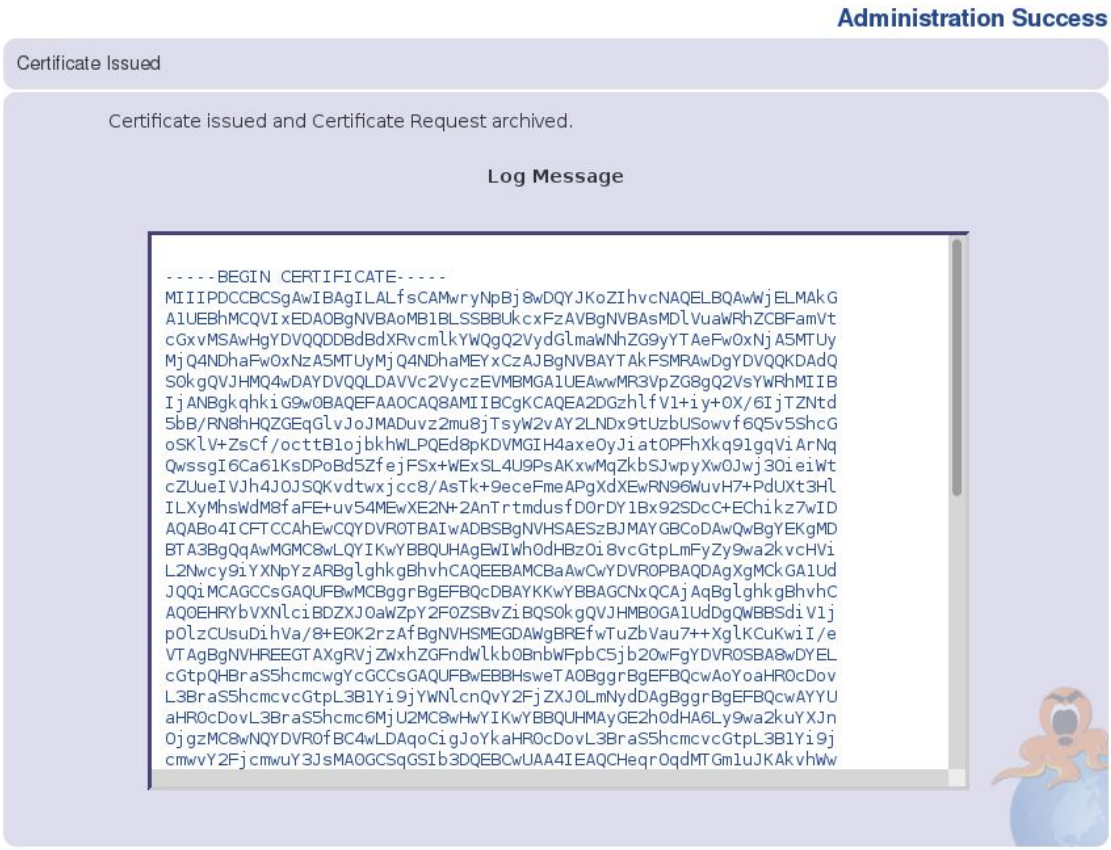


Figura 46. Certificado generado en formato texto.

Como último paso ingresamos nuevamente como un usuario a la sección pública del sitio de la PKI e ingresamos en "*My Certificates > Install My Certificate*", ingresamos el número serial de la solicitud y presionamos el botón "*OK*". Esto nos instala nuestro certificado personal en nuestro navegador y ya podremos empezar a utilizarlo. Podremos verificar esto en las opciones avanzadas del explorador, en la sección de "*Mis certificados*".

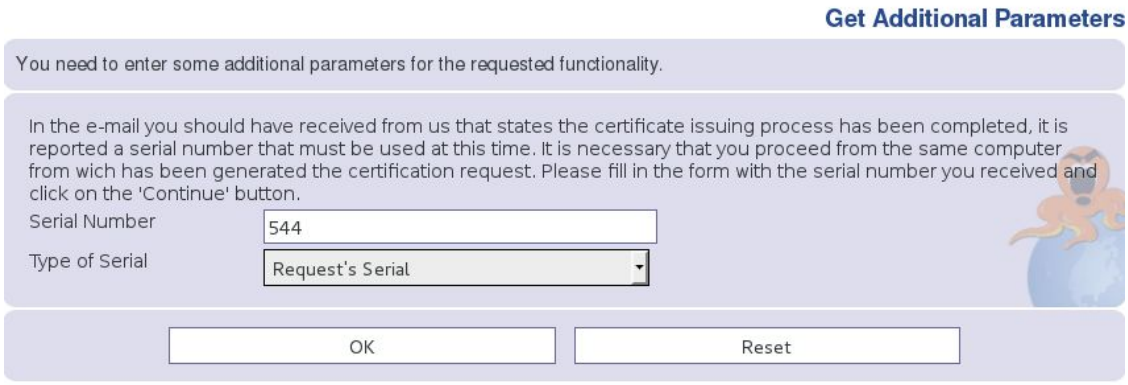


Figura 47. Búsqueda de certificado por serial number para ser instalado en el explorador.

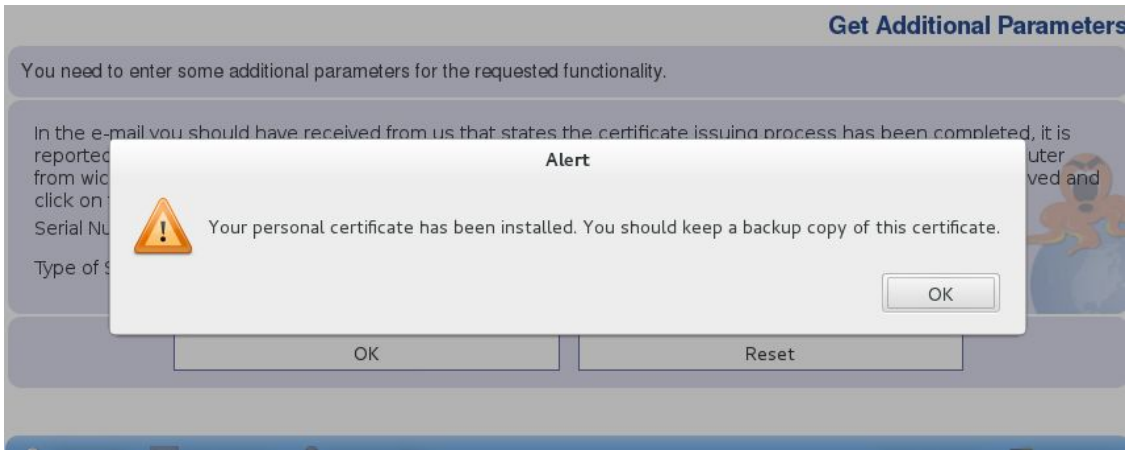


Figura 48. Instalación del certificado en Firefox.

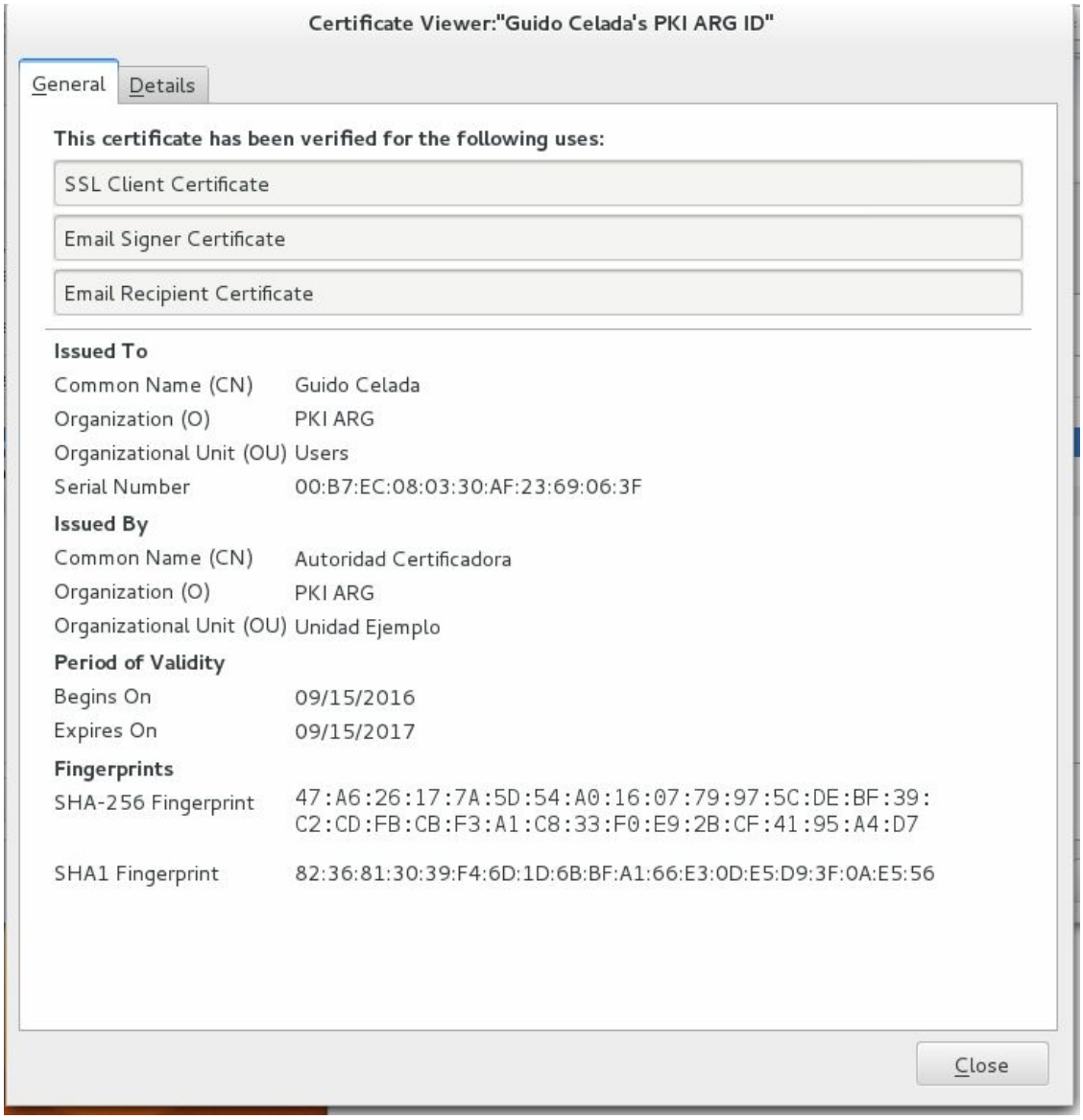


Figura 49. Detalles del certificado instalado en Firefox.

Revocar un certificado:

Para revocar un certificado válido debemos ingresar como administrador de la CA al sitio de la PKI e ingresar desde el menú a la sección "*Information>Issued Certificates>Valid*" en donde podremos observar un listado con los certificados válidos de la PKI.

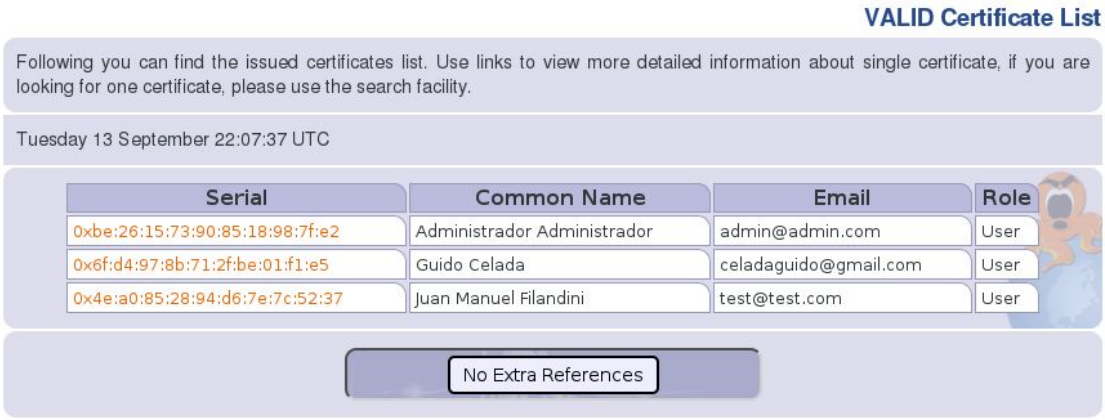


Figura 50. Listado de certificados válidos.

Como primer paso seleccionaremos el certificado que deseamos revocar haciendo click en su "Serial", esto nos lleva al detalle del certificado en donde podremos verificar el estado y todos los datos del mismo, así como también tendremos la opción de revocar el certificado haciendo click en el botón "Revoke".

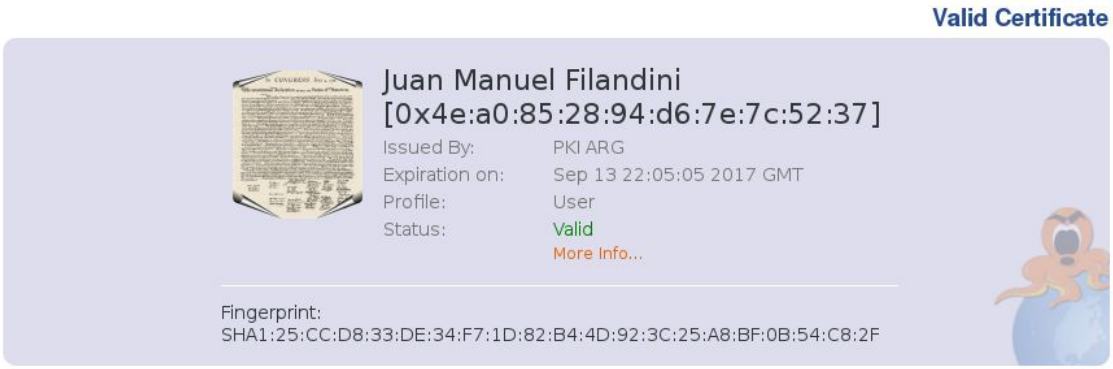


Figura 51. Detalles del certificado a revocar.

Valid Certificate

Following you can find the certificate details.

Tuesday 13 September 22:08:20 UTC

General Information	
Certificate Version	3 (0x2)
Serial Number	371305659798280933757495 (0x4e:a0:85:28:94:d6:7e:7c:52:37)
Common Name	Juan Manuel Filandini
E-Mail	test@test.com
Distinguished Name	CN=Juan Manuel Filandini OU=Users O=PKI ARG C=AR
Role	User
Fingerprint	SHA1:25:CC:D8:33:DE:34:F7:1D:82:B4:4D:92:3C:25:A8:BF:0B:54:C8:2F
Issued by	CN=Autoridad Certificadora OU=Unidad Ejemplo O=PKI ARG C=AR
Valid From	Sep 13 22:05:05 2016 GMT
Expiration on	Sep 13 22:05:05 2017 GMT
Current Status	Valid
Extensions	
Authority Information Access	CA Issuers - URI:http://pki.arg/pki/pub/cacert/cacert.crt OCSP - URI:http://pki.arg:2560/ PRQP Service - URI:http://pki.arg:830/

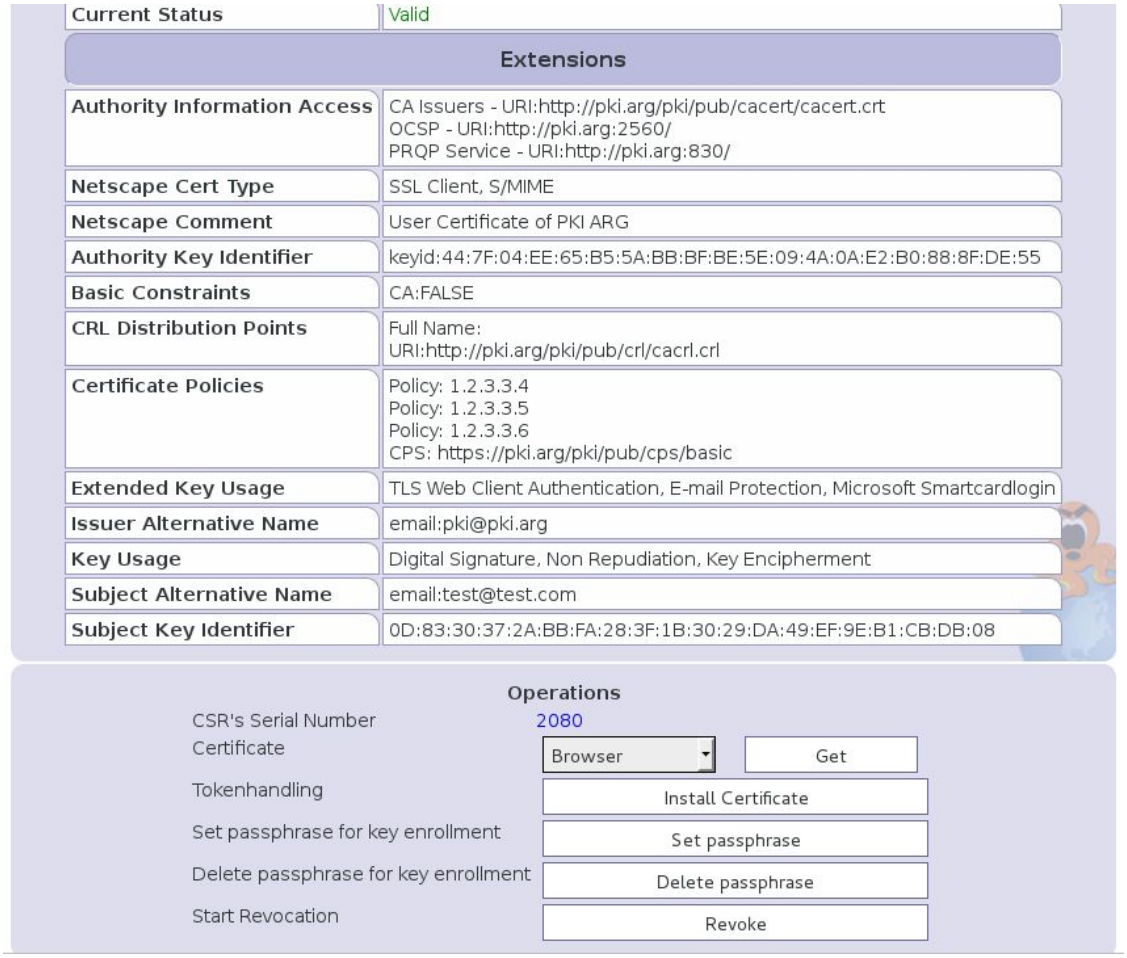


Figura 52. Detalles ampliados del certificado a revocar.

Presionamos el botón para revocar el certificado e indicamos la razón de la revocación. Esto genera una petición de revocación del certificado que en este caso será aceptada por el mismo administrador de la CA.

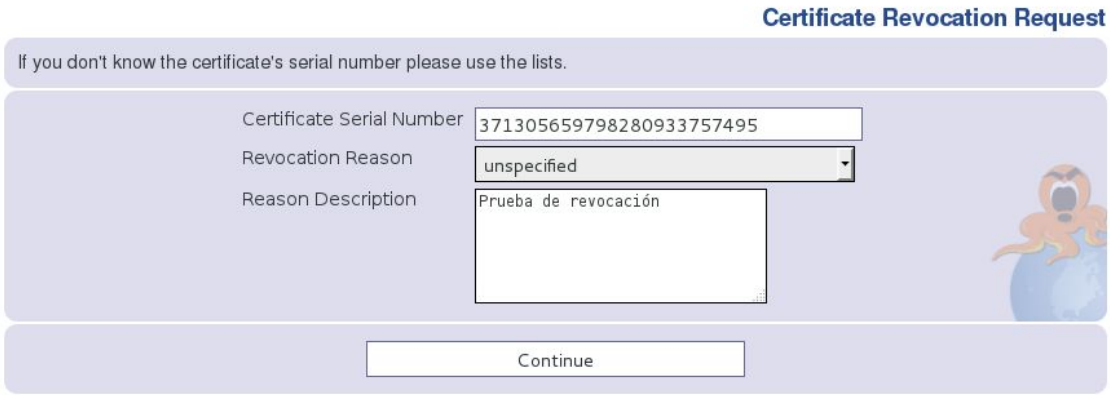


Figura 53. Pantalla de revocación de certificado.

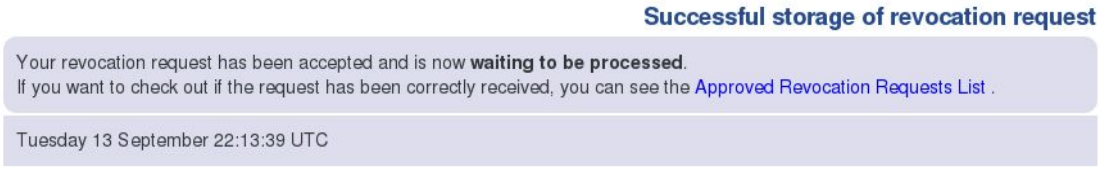


Figura 54. Confirmación de solicitud de revocación enviada.

Para poder aprobar la petición de revocación de certificado ingresamos a la sección del menú "*CA Operations>Revocation Requests*" en donde se nos mostrará otro listado con todas las peticiones de revocación. Aquí seleccionamos la petición y podremos observar una pantalla con los datos del pedido de revocación. En esta misma pantalla presionamos el botón para aceptar la solicitud y esto automáticamente genera que el certificado pase a estado de revocado y no pueda utilizarse más.

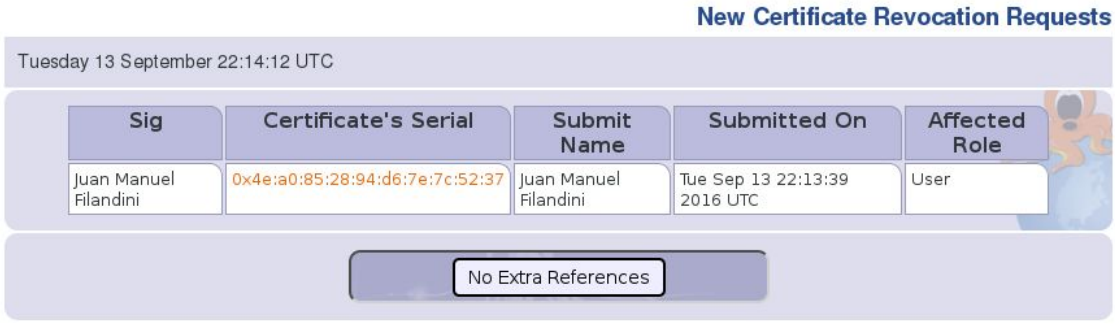


Figura 55. Solicitudes de revocación pendientes.

Figura 56. Solicitud de revocación de certificado.

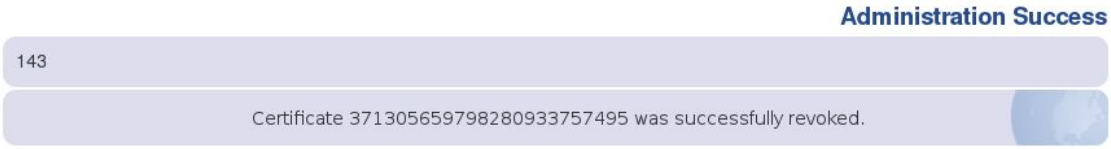


Figura 57. Confirmación de revocación.

Como último paso deberemos generar una nueva CRL en donde se indique al público todos los certificado revocados que ya no deben ser confiados. Para esto vamos a "*CA*

Operations > Issue new CRL" e indicamos el periodo de validez de la CRL.

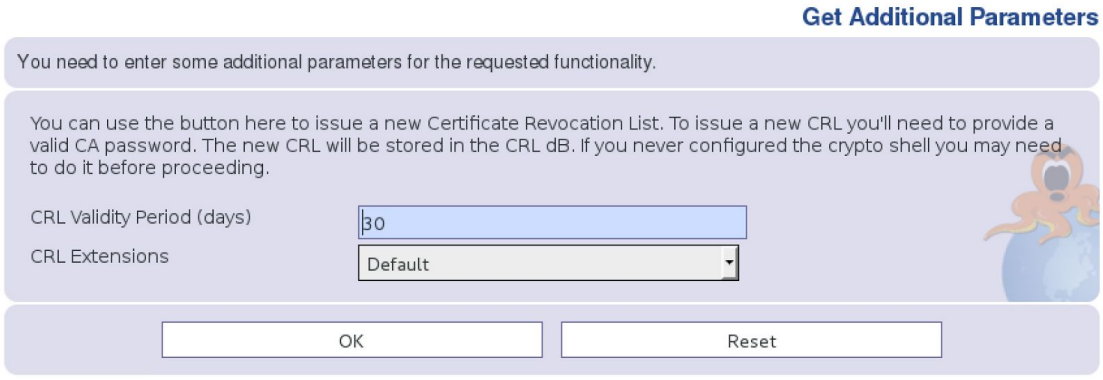


Figura 58. Creación de una CRL.

Podremos verificar nuestra nueva CRL ingresando a "*Information>CRLs>Valid*". Aquí se nos mostrará en pantalla la CRL y los certificados revocados incluidos en esta. Ingresamos al certificado de la CRL y comprobamos que se está incluyendo el certificado revocado anteriormente en la CRL.

Figura 59. CRL.

Figura 60. Certificado revocado.

2.4.3- EJBCA

Enterprise Java Beans Certificate Authority, o EJBCA, es un paquete de software libre y gratuito de PKI y Autoridad Certificadora mantenido y presentado por la compañía PrimeKey Solutions AB la cual es poseedora de los derechos de autor de la mayoría del código base. El código fuente está disponible bajo los términos de la licencia GNU (Lesser GNU General Public License).

El sistema está implementado en Java EE y fue diseñado para ser independiente de la plataforma además de poder ser utilizado en un cluster, para permitir un alto grado de escalabilidad a diferencia del que es típico en paquetes de software similares. Múltiples instancias de EJBCA son ejecutadas simultáneamente, compartiendo una base de datos que contiene las autoridades certificadoras(CA) actuales. Esto permite que cada instancia pueda acceder a cualquier CA. El software también soporta el uso de HSM(Hardware Security Mode), el cual provee seguridad adicional. Instalaciones a gran escala utilizan múltiples instancias de EJBCA ejecutándose en un cluster, una base de datos distribuida en un cluster separado y un tercer cluster con HSM, almacenando las diferentes claves de las CA.

EJBCA soporta varias arquitecturas PKI, como por ejemplo:

- Todo en un solo servidor.
- Autoridades de registro(RA) y validación de autoridad externa.

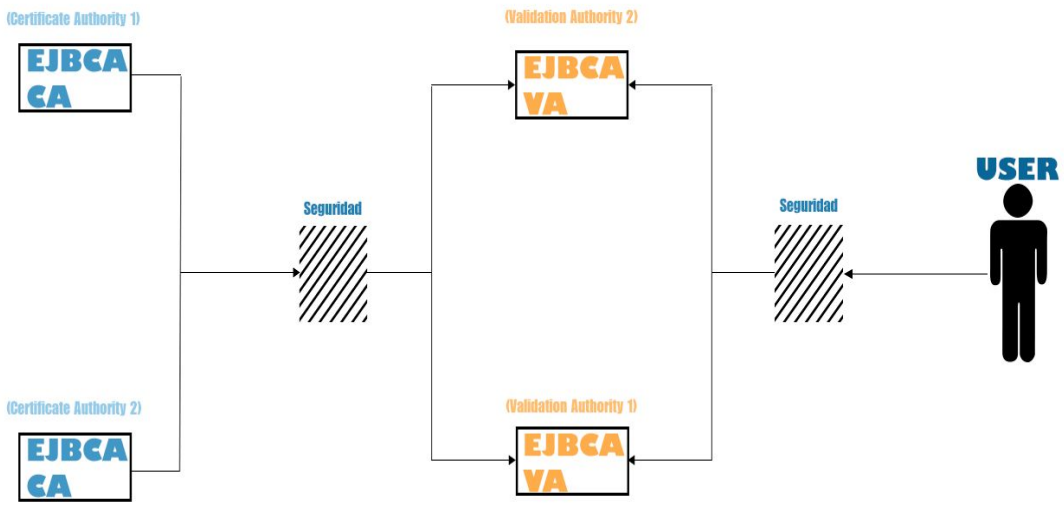


Figura 61. Arquitectura de EJBCA.

2.4.3.1- Ejemplo de PKI simple utilizando EJBCA

2.4.3.1.1- Creación, instalación y configuración de la infraestructura

EJBCA nos provee una maquina virtual con todo lo que necesitamos instalado configurado y funcionando. Sin embargo si se requiere una instalación desde cero se deben seguir los siguientes pasos:

Guía de instalación:

Para la siguiente guía de instalación utilizaremos:

- JDK 7 OpenJDK o Oracle JDK, si está disponible OpenJDK está recomendada.
- Si se utiliza OracleJDK se necesita Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files para la JDK.
- JBoss Application Server 7.1.x en adelante o JBoss EAP 6 en adelante. O WildFly 8 en adelante.
- Apache Ant 1.8 en adelante para construir. Nota: El soporte javascript puede ser requerido en ant para algunos componentes.

Archivos de configuración:

La configuración de EJBCA no puede hacerse desde la interfaz de Admin, en cambio se encuentra ubicada en archivos de propiedades dentro del directorio conf. Todas las propiedades están documentadas en un archivo de muestra(sample file) y para configurar una opción se lo copia. Se recomienda familiarizarse con los contenidos de los archivos:

- conf/install.properties,conf/ejbca.properties
- conf/cesecore.properties.

La mayoría de las opciones excepto aquellas dentro de install.properties pueden ser cambiadas después de la instalación.

Configuración de EJBCA:

- Copiar conf/install.properties.sample a conf/install.properties.
- Copiar conf/ejbca.properties.sample a conf/ejbca.properties.
- Copiar conf/cesecore.properties.sample a conf/cesecore.properties.

En caso de tener que hacer alguna configuración personal, este es el paso donde se realiza. Por defecto con copiar estos archivos sirven para una instalación de prueba.

Se debe configurar appserver.home' dentro de ejbca.properties para apuntar al directorio del servidor de la aplicación. Ejemplos de como hacer eso se encuentran dentro de ejbca.properties.sample.

Esto hace que las librerías del servidor de la aplicación se encuentren disponibles para EJBCA durante la construcción.

Si solo se va a probar EJBCA, y no se está armando un ambiente de producción, se puede saltear el resto de estas indicaciones. Ya que con las configuraciones por defecto debería funcionar para todo.

-Personalizar las propiedades de las CA en conf/ejbca.properties si se necesita. Para un ambiente en producción este paso es esencial, se debe editar las contraseñas para que sean seguras y secretas. Mantener el archivo conf/ejbca.properties lo más oculto y secreto posible.

-Para utilizar un hard ca token desde el principio se debe cambiar ca.tokentype, ca.tokenpassword y ca.tokenproperties en el archivo install.properties.

También es necesario agregar los valores apropiados al archivo ca.tokenproperties para el HSM(Hardware Security Module). Los valores adecuados para realizar dicho cambio se encuentran en la documentación de HSM.

-Para poner el certificado de superadmin inicial en una tarjeta inteligente, se debe configurar la propiedad superadmin.batch en false, en web.properties. Inscribirse desde la web pública después de que finalice la instalación, el nombre de usuario es "superadmin" y la contraseña es la que se encuentra en superadmin.password en web.properties.

-Si se está implementando en JBoss EAP se debe mirar la propiedad jboss.config, ya que 'produción' puede ser el servidor por defecto para inicar en JBOSS Eap.

-Para personalizar la base de datos se deben modificar las propiedades en conf/database.properties si es necesario. Pero es más fácil dejar la configuración por defecto como esta. De esta manera EJBCA utiliza JBoss con HSQLDB embebido. Para producción utilizar una base de datos real en vez de la embebida.

Extender la implementación Sun PKCS#11:

Para cambiar el atributo CKA_MODIFIABLE de una clave privada a falso directamente después de que ha sido generada la implementación Sun PKCS#11 debe ser extendida. Esta extensión debe hacerse mediante el agregado de clases al classpath "Installed Extensions".

Se logra poniendo '$EJBCA_HOME/dist/ext/cesecore-p11,jar' en uno de los directorios que es definido por la propiedad de sistema 'java.ext.dirs'. El jar puede ser colocado en '$JAVA_HOME/jre/lib/ext'.También se puede cambiar la propiedad para incluir el directorio del jar de la siguiente manera:

JAVA_OPTS="-Djava.ext.dirs=/usr/lib/jvm/java-7-openjdk-amd64/jre/lib/ext:$EJBCA_HOME/dist/ext" $JBOSS_HOME/bin/standalone.sh

Se debe mantener '$JAVA_HOME/jre/lib/ext' en el classpath.

También es necesario configurar 'pkcs11.makeKeyUnmodifiableAfterGeneration=true' en $EJBCA_HOME/conf/cesecore.properties

Si se utiliza '$EJBCA_HOME/dist/ejbcaClientToolBox.sh' los pasos de arriba no son necesarios ya que el script se encarga de ejecutarlos.

Si no se realizan estos cambios CKA_MODIFIABLE estaría configurada en TRUE para toda clave que es generada.

La razón de poner dicha variable en false es que no debería ser posible configurar la variable CKA_DERIVE en verdadera, ya que si se encuentra en ese valor es posible extraer la clave privada de algunos HSMs.
Algunos HSMs se niegan a cambiar CKA_MODIFIABLE(Según el estándar este comportamiento es aceptable).
Se puede verificar ejecutando el siguiente comando:
$EJBCA_HOME/dist/ejbcaClientToolBox.sh PKCS11HSMKeyTool test

Configurar el servidor de la aplicación:

Debido a algunos bugs y problemas de con ciertos servidores, para lograr un correcto funcionamiento se recomienda utilizar la guia de instalacion en distintos tipos de servidores encontrada en la página oficial de EJBCA.

Consideraciones:

Una vez está todo preparado, hay que configurar unas pequeñas cosas antes de iniciar las aplicaciones y tener todo ejecutándose en el ambiente de producción.
En un ambiente de producción se recomienda utilizar algo similar a la siguiente estructura:

1. Crear una AdminCA. Utilizar un DN simple. Esta CA solo debe ser utilizada para emitir certificados de administrador. No debe ser publicada en LDAP(Lightweight Directory Access Protocol). Si se desea utilizar HSM para dicha CA, se recomienda utilizar la documentación ubicada en el archivo conf/ejbca.properties.sample.
2. Una vez instalado, crear todas las CA reales utilizando la admin-Gui.Se pueden utilizar lo perfiles de certificado que se deseen. Estos certificados pueden ser publicados en la LDAP(Lightweight Directory Access Protocol).

En un ambiente de producción se debería utilizar una base de datos diferente a la que viene con JBoss(Hypersonic) por las siguientes razones:

1. La base de datos Hypersonic es en-memoria, lo que significa que con el tiempo consumirá más memoria y más memoria, si un número grande de certificados es emitido,podría volverse un problema.

2. La base de datos Hypersonic no soporta SQL por completo, en particular las consultas con ALTER. Esto significa que si se libera una nueva versión de EJBCA no se puede escribir un script que pueda actualizar la base de datos.

Instalación en JBOSS:

La instalación debe hacerse con un usuario que tenga los privilegios para escribir APPSRV_HOME y sus subdirectorios. Ejecutar los comandos de implementación e instalaciòn como descriptos a continuación también configura automáticamente el servidor de la aplicación con los datasources y la configuración web.

1) Configurar la propiedad 'appserver.home' en conf/ejbca.properties hacia donde está instalado el JBoss, por ejemplo:

appserver.home=/opt/jboss-7.1.1.Final

Asegurarse que las correctas herramientas java están disponibles en el system PATH, ejemplo:

./usr/local/jdk1.7.0_25/bin.

2) Abrir una terminal e iniciar el JBoss, se puede utilizar el comando normal 'standalone.sh' de APPSRV_HOME/bin. Jboss debería encontrar y cargar las implementaciones de los war sin problemas ni errores.

3) Abrir una consola y ejecutar el siguiente comando:

ant deploy

El comando va a compilar y construir EJBCA e implementarlo en el JBoss como también crear datasources y el servicio de mail automáticamente. Se nos preguntará el valor para database.password si no fue configurado previamente en database.properties.

4) Ejecutar:

ant install

El comando generará los certificados claves y todo lo que sea necesario para correr EJBCA con la CA inicial. Además va configurar TLS en el contenedor de servlet para

usar archivos keystore y truststore(Los cuales también son copiados en el proceso).Las claves de admin se encontraran en ${ejbca.home}/p12.
El comando 'ant install' solamente se ejecuta una vez, cuando la CA es instalada por primera vez. Crea muchas cosas en la base de datos y no puede ser ejecutado nuevamente.
superadmin.p12 debería ser importado en el navegador, es el certificado de administración.

5)Reiniciar el servidor JBoss

6)Importar el certificado de EJBCA_HOME/p12/superadmin.p12 en el navegador web. Este es el certificado de superadmin utilizado para ingresar a la Admin-GUI Otros administradores con privilegios específicos pueden ser creados en esta página. La contraseña por defecto para superadmin12 es 'ejbca' y se encuentra configurada en web.properties.

7)Poner la siguiente url en el navegador para acceder a la admin Gui https://localhost:8443/ejbca/, o http://localhost:8080/ejbca para las páginas públicas.
Si se crean otras CA que se quieren agregar como CAs aceptables en la configuración SSL del servidor, o renovar el certificado de la CA, se puede instalar cualquier certificado de CA en la configuración SSL del servidor con el siguiente comando:

ant -Dca.name="Nombre de mi CA" javatruststore

Lo que esto hace es agregar el certificado CA a p12/truststore.jks y copia este archivo en JBOSS_HOME/server/default/conf/keystore, donde los keystore SSL están ubicados, se debe reiniciar el JBoss después de ejecutar este comando.

2.4.3.1.2- Operativa

Realizar solicitud de certificado de usuario:

Para realizar la solicitud de un certificado de usuario se realizarán los siguientes pasos:

Primero se debe hacer click en la opción *"request registration"*, la cual nos mostrará la siguiente selección, donde tendremos la elección *"User Certificate"* para realizar la creación de un certificado de usuario.

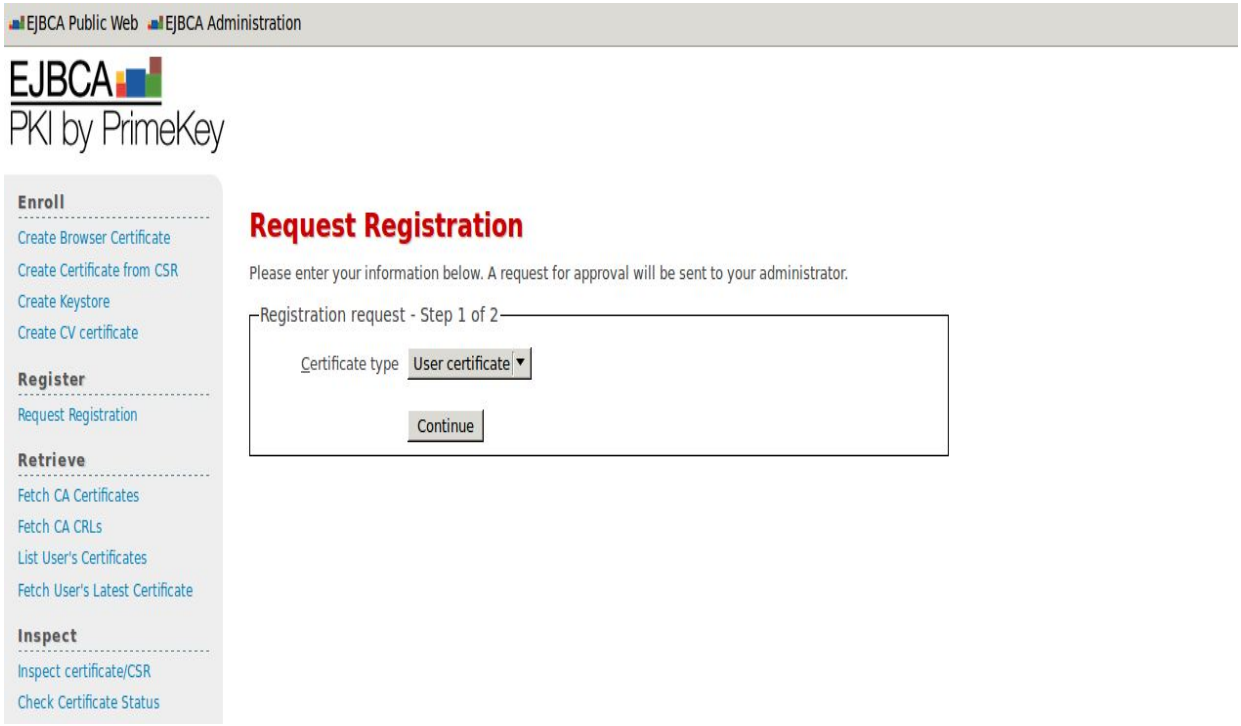


Figura 62. Creación de solicitud de certificado desde la web pública.

A continuación se nos presenta el siguiente formulario en el cual se deben completar los campos correspondientes a:

-Nombre: Nombre del usuario en cuestión

-E-mail: Email del usuario al cual va a pertenecer el certificado, esto es para luego enviar un mail con la clave para ingresar al sistema.

-Last character in E-mail: Este campo es para validar que no se realicen solicitud de certificados de manera automática, nos solicita los últimos caracteres del email para continuar con el registro.

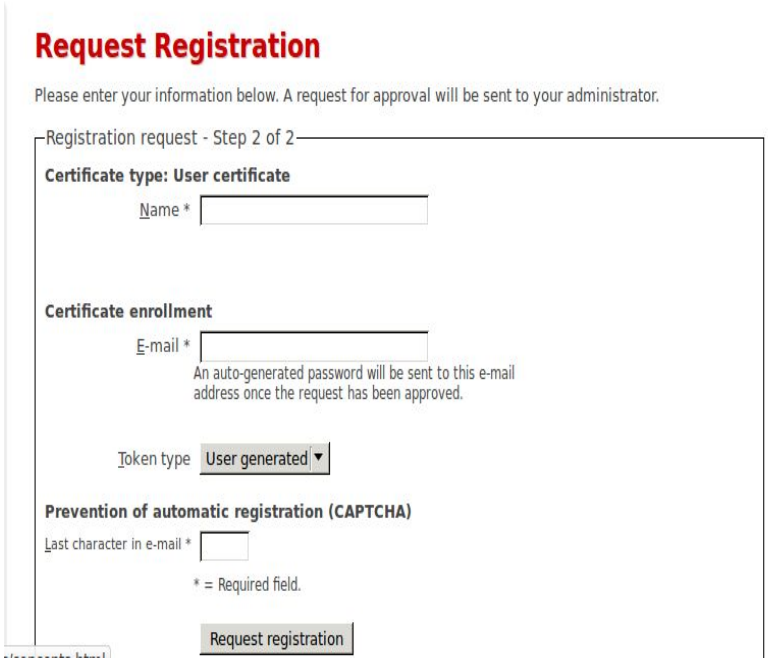


Figura 63. Formulario para la solicitud de certificado.

Una vez completados los campos mencionados, se debe hacer click en el botón *"Request registration"*, el cual enviará una solicitud de creación de certificado al administrador del sistema.

Desde la vista pública a la cual puede acceder cualquier persona que entre a la página web de EJBCA, se pueden visualizar distintas opciones públicas para interactuar con las CA.

La primera es para obtener las CRLs de la CA, al hacer clic en *"Fetch CA CRLs"* se nos da la posibilidad de por cada CA obtener las CRL.

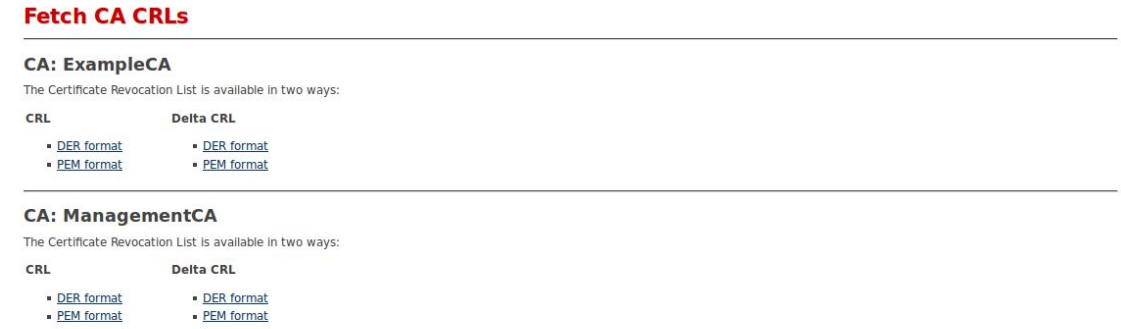


Figura 64. Obtención de CRL.

La segunda opción que vemos al hacer clic en *"Fetch Ca Certificates"* nos permite obtener los certificados de la CA correspondiente a la que elijamos para observar

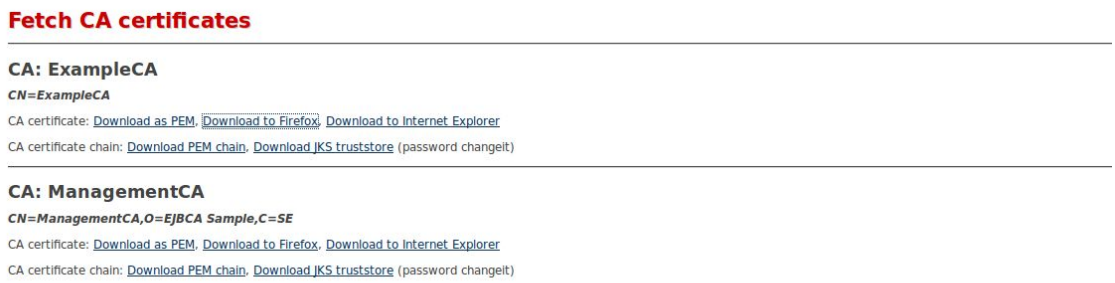


Figura 65. Obtención de los certificados de la CA.

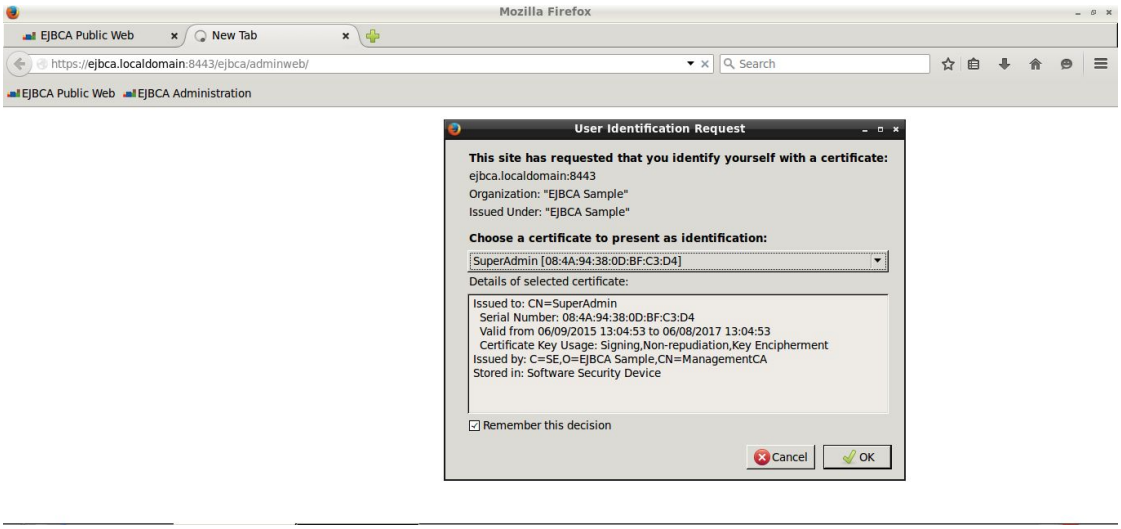


Figura 66. Ingreso a la web de administración

Del lado de la administración de EJBCA con lo primero que nos encontramos es que nos solicita el certificado de admin, este mismo es requerido para interactuar con la parte de administración de EJBCA.

Una vez verificada nuestra identidad mediante el uso de certificados accedemos a las operaciones que se pueden realizar en la vista de admin.

Al hacer click en la opción *"Certification Authorities"* nos habilita el siguiente formulario para realizar la creación de una Certification Authority.

En este formulario podemos personalizar la CA para que responda a nuestras necesidades o emita los certificados según nos convenga.

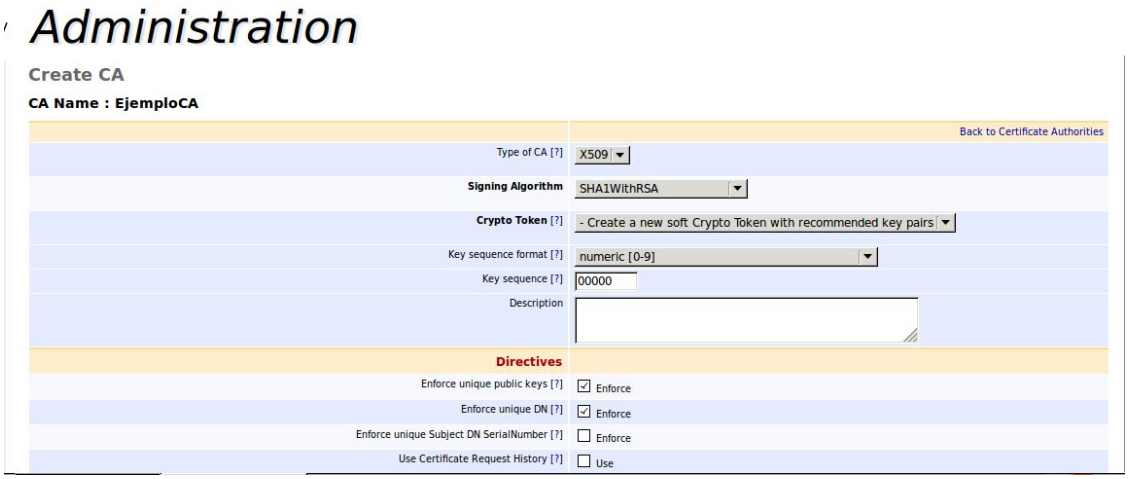


Figura 67. Creación de una CA.

En esta vista también tenemos la posibilidad de crear una CRL actualizada para cada CA creada y funcionando.

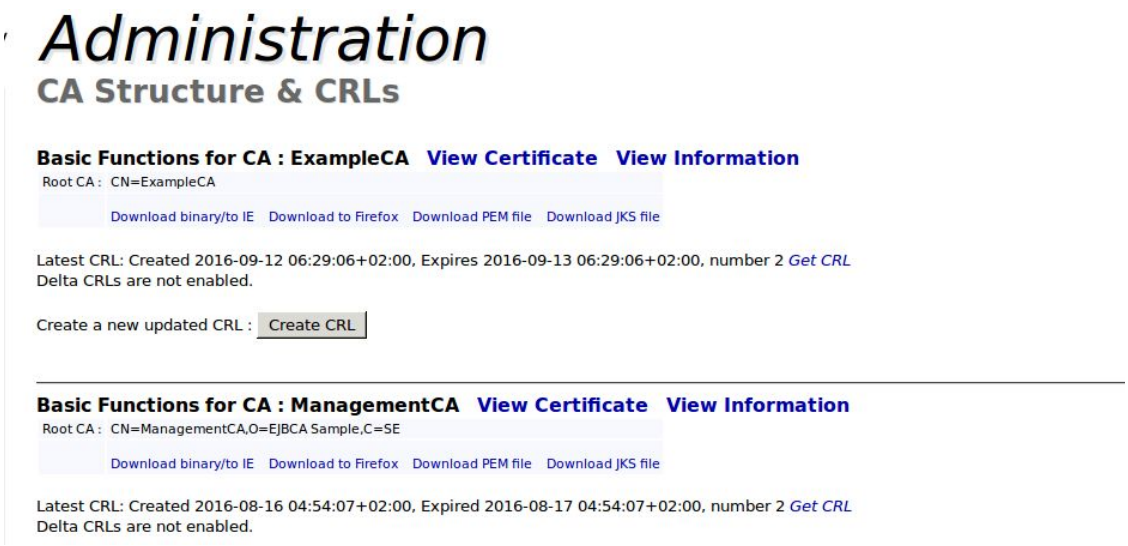


Figura 68. Generación de CRL.

Luego tenemos disponible la pantalla de administración de requerimientos los cuales abarcan desde agregar entidades hasta la creación de certificados de usuarios entre otras cosas. En dicha pantalla nos aparecerá la solicitud de creación de certificado de usuario

que realizamos en la pantalla pública, debe notarse que podemos acceder a esta pantalla ya que se trata de un ejemplo, en un caso real la persona que solicita un certificado de usuario podría no ser la misma que la que administra el sistema.

Seleccionamos la solicitud de creación de certificado que realizamos al principio de este ejemplo,y se mostrará la siguiente información, en esta pantalla tenemos la opción de decidir si aprobar o no el certificado en cuestión.

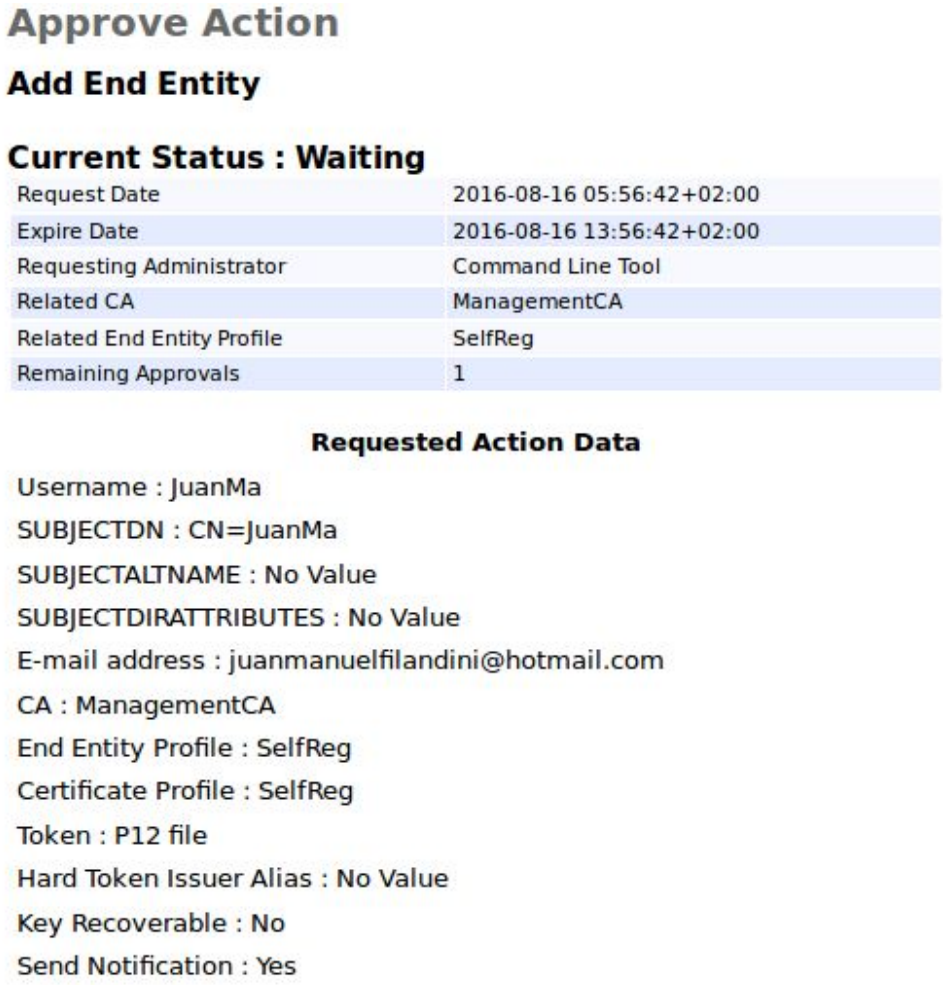


Figura 69. Aceptación de una nueva entidad.

Una vez realizada una acción en esta pantalla, dependiendo del a acción realizada pueden ocurrir dos cosas:

-Rechazo de certificado: Si se rechaza el certificado quedará registro de la acción y la persona que lo haya solicitado deberá comunicarse con el administrador del sistema o hacer otra solicitud en caso de ser necesario.
-Aceptación de certificado: Si el administrador del sistema acepta la solicitud, se enviará un mail al usuario en cuestión, con la contraseña generada para ingresar al sistema.

Realizar una revocación de certificado desde la vista de administración:

Para el siguiente ejemplo vamos a trabajar con el certificado del usuario Juan Manuel Filandini que fue creado anteriormente.
Al momento de revocar un certificado desde la vista de administración se deben seguir los siguientes pasos:

Buscamos el certificado mediante el username que fue utilizado al momento de crear la solicitud de certificado de usuario. En este caso Juan Manuel Filandini.

Figura 70. Búsqueda de entidades finales.

Luego hacemos click en el botón *"Search"*,nos mostrará en una lista todos los certificados cuyo CN coincida con el nombre que ingresamos previamente.

Figura 71. Listado de entidades finales.

Una vez encontrado el certificado, elegimos una razón de revocación, seleccionamos el certificado que queremos revocar, y hacemos click en el botón *"Revoke selected"*, nos preguntará si estamos seguros de revocar el certificado seleccionado, clickeamos en *"OK"* y el certificado será revocado.

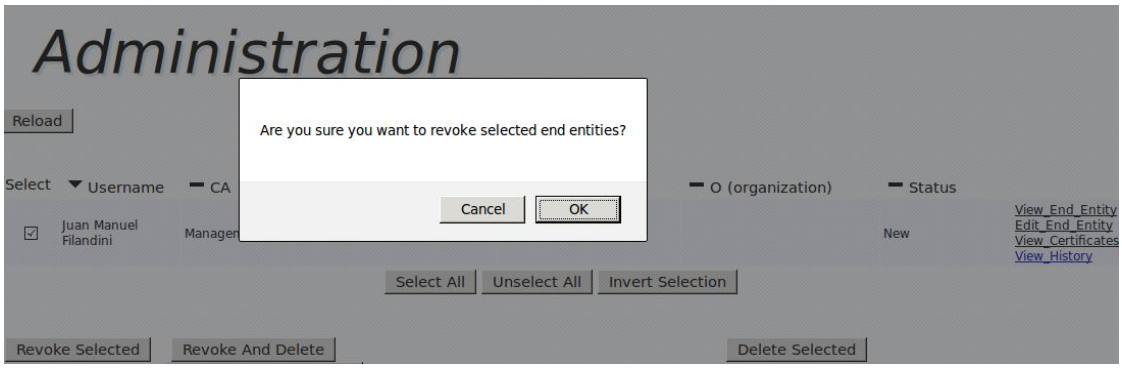


Figura 72. Revocación de un certificado.

Al revocarse el certificado quedará marcado como *"Revoked"* en la columna status de la lista esto nos indica que el estado del certificado es efectivamente "*Revocado"*.

Figura 73. Listado de entidades finales mostrando el estado revocado.

Envío de información mediante email de EJBCA:

Mediante el uso de EJBCA el administrador del sistema puede definir una casilla de mail, sobre la cual le llegan mails con notificaciones pertinentes al sistema.

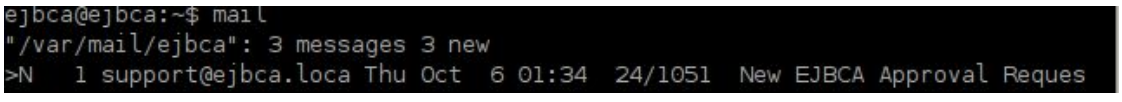


Figura 74. Mails del sistema.

En el momento en el que el usuario realiza el registro mediante la solicitud de un certificado de usuario, al administrador del sistema se le envía un mail con la siguiente información

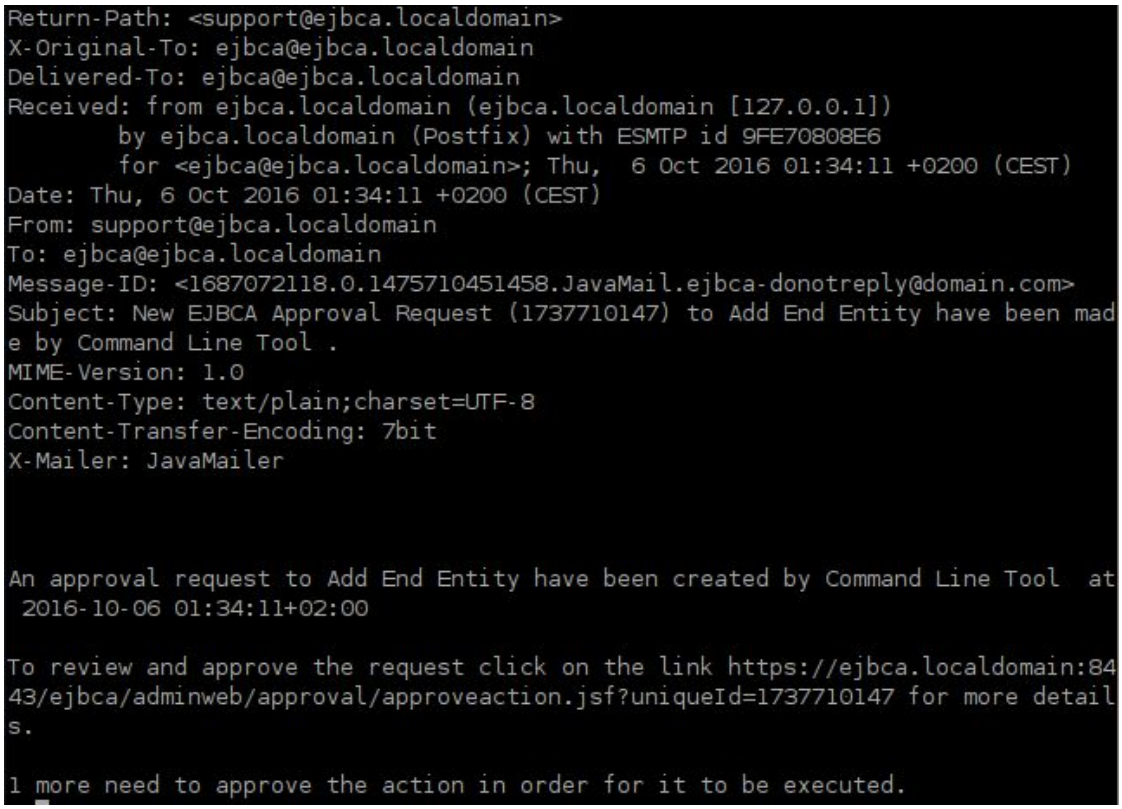


Figura 75. Mail de solicitud de certificado.

Lo cual nos indica que, existe una acción pendiente de aprobación, para agregar una *"End entity"*. Es decir, el usuario.

Una vez tomada una decisión sobre la acción pendiente,recibiremos un mail que contiene información sobre quien decidió sobre dicha acción y que decisión tomo.

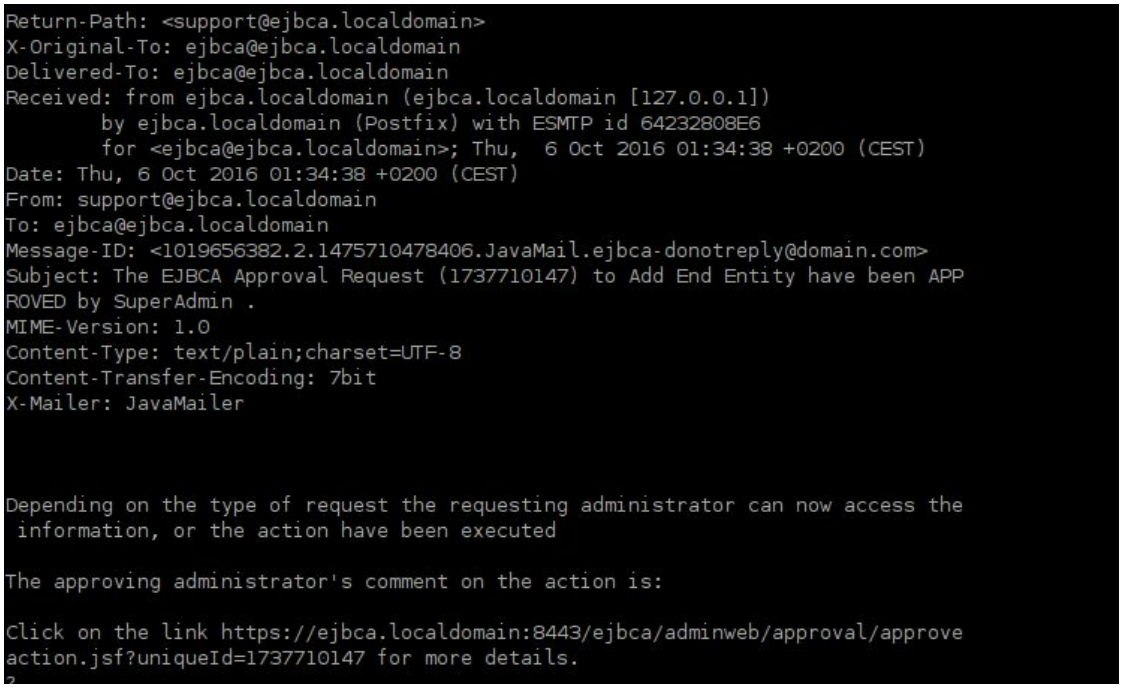


Figura 76. Mail de aprobación de la solicitud.

A su vez, el usuario que solicitó el registro, en este caso Juan Manuel Filandini, que es el usuario utilizado por defecto para el ejemplo, recibirá el siguiente mail en su casilla de correo.

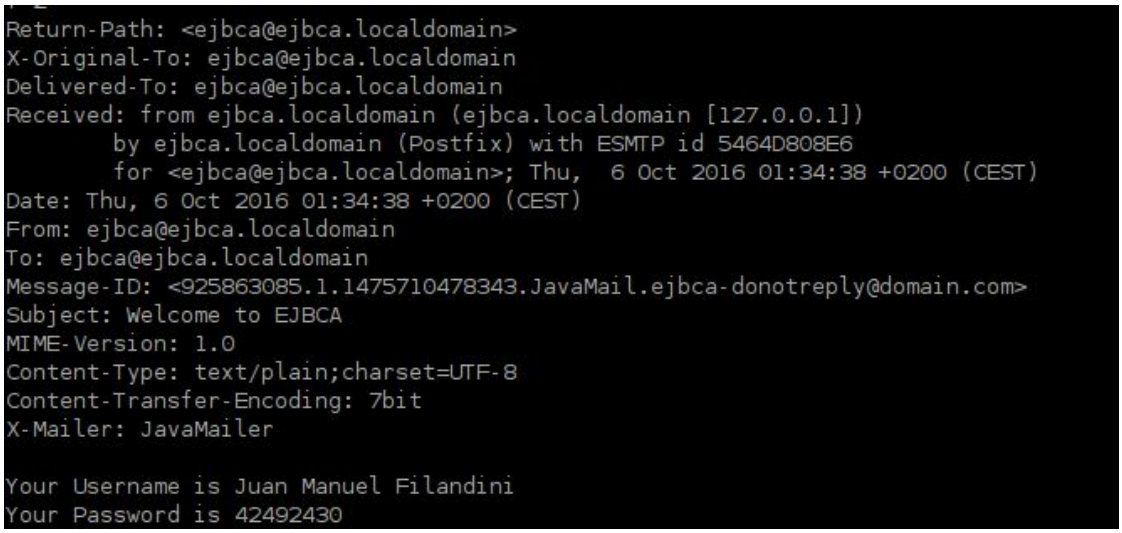


Figura 77. Mail enviado al usuario que requirió el certificado.

El mismo contiene nombre de usuario y contraseña o enrollment code.

3- PKIGrid CA U.N.L.P

3.1- Descripción

Hoy en día existen 69 iniciativas de PKI que ofrecen acceso a diferentes aplicaciones GRID para E-science, cumpliendo con las políticas establecidas por el International Grid Trust Federation (IGTF), el cual se encarga de establecer politicas y guias en comun para establecer relaciones de confianza globales interoperatebles entre proveedores de PKI. Las aplicaciones GRID para E-science promueven adelantos en investigación a partir de la posibilidad de compartir recursos globalmente. En estos casos es importante fortalecer los mecanismos de seguridad subyacentes a fin de garantizar que los mismos sean usados únicamente por grupos autorizados y de una manera adecuada. En este sentido, son muchos los avances que la tecnología ha ido realizando, entre ellos el uso de los certificados digitales X.509 y las infraestructuras PKI, que soportan su creación y uso.
El sitio se encuentra disponible en http://www.pkiunlprid.unlp.edu.ar.

Figura 78. Web pública de PKIGrid UNLP.

3.2- Adaptación con OpenCA

Para poner en marcha la CA raíz de una Infraestructura de clave pública en el CeSPI-UNLP para proyectos Grid de E-science se decidió utilizar OpenCA. Este producto se ha adaptado para cumplir con las reglas definidas en las políticas de certificados y declaraciones de prácticas de certificación de la PKI a implementar, las cuales cumplen con las pautas mínimas definidas por TAGPMA.

Para su implementación se decidió utilizar un modelo de confianza jerárquico con una única CA raíz y una única RA hija.

A fin de proveer un alto nivel de seguridad, la CA fue instalada en una máquina off-line, completamente aislada de la red, cumpliendo con lo establecido en las políticas de certificados y declaraciones de prácticas de certificación definidas para nuestra PKI. Por otra parte, la RA y el Sitio público residen en otra máquina conectada a la red, que solamente brinda esos servicios (se han instalado y habilitado únicamente los paquetes necesarios para que la RA y el sitio público funcionen) y que se encuentra conectada a una DMZ4 altamente monitoreada.

A fin de lograr seguridad y modularización de las tareas de administración se definen roles, los cuales se describen en el manual de procedimiento de la PKI. OpenCA permite identificar roles, módulos y funciones, de manera tal que cada rol tenga claramente establecidos los permisos de acceso que determinan en qué módulo puede actuar un operador y cuáles son las tareas que puede realizar en el mismo. Al emitir un certificado, OpenCA permite seleccionar un rol asociado al mismo. Esa información no forma parte del certificado pero si se incluye como información interna para OpenCA. El control de acceso consiste en:

- Verificación del canal: se encarga de chequear los parámetros de una conexión entrante para detectar clientes obsoletos o mal configurados.
- Login: establece el modo de loguearse a la interfaz de OpenCA, que puede ser sin mecanismo alguno, a través del mecanismo de usuario/clave o a través del uso de certificados.
- Manejo de sesión: establece algunas opciones relacionadas con el manejo de sesiones CGI.
- ACLs: permite definir las reglas de control de acceso a la interfaz de cada módulo,especificando si se usará una lista de control de acceso, cómo se utilizará

el certificado de la CA con el que se verificarán los certificados usados para autenticarse, si se habilitará el mapeo de certificados con sus correspondientes roles, si se habilitará el mapeo de nombres de scripts con operaciones.

- Uso de certificados: Se ha definido que sólo un usuario que presenta un certificado con rol de CA Operator puede acceder a la interfaz de administración de la CA y que sólo un usuario que presenta un certificado con rol de RA Operator puede acceder a la interfaz de administración de la RA.

A fin de resguardar las claves de los operadores, se decidió almacenar las mismas en dispositivos criptográficos que las protejan de usos indebidos. Esto es posible puesto que tales dispositivos, luego de validar al poseedor del mismo, permiten realizar operaciones de firma/verificación/encriptación a través de una API. La seguridad está dada porque la clave privada nunca sale del dispositivo, y ante un intento de fuerza bruta sobre el mismo, éste se bloquea sin posibilidad de recuperar los datos allí almacenados.Como dispositivos criptográficos se utilizaron los provistos por la empresa Aladdin, llamados etokens pro de 32K. Los mismos permiten operaciones de 1024 bits por lo cual es posible almacenar claves como máximo de 1024 bits. Se los probó con éxito tanto en Windows como en Linux, utilizando dirvers abiertos y propietarios, y testeando la posibilidad de usar un etoken en Windows y en Linux indistintamente. En particular, la utilización de estos dispositivos brinda movilidad al operador de la RA, que posee el etoken, ya que el mismo puede administrar la RA desde cualquier host, siempre y cuando las políticas del firewall de la red lo permitan.
Para realizar la definición de perfiles de certificados en OpenCA se trabajó sobre:

- Adaptar el contenido del directorio ${OpenCADIR}/etc/openssl/openssl, de manera tal que para cada perfil de certificados a emitir exista un archivo “perfil”.conf, en el cual se definan los valores de los campos obligatorios del certificado X509. Estos archivos se usan para armar los requerimientos en el módulo de la RA.
- Adaptar el contenido del directorio ${OpenCADIR}/etc/openssl/extfiles, de manera tal que para cada perfil de certificados a emitir exista un archivo “perfil”.ext, en el cual se definan los campos que se incluirán como extensiones y sus valores. Estos archivos se usan para construir los certificados en el módulo de la CA.

- El hecho de poder agregar nuevos perfiles de certificados y de poder manejar sus campos es una facilidad presente en la versión usada de OpenCA, la cual no estaba incluída en versiones anteriores.
- Adaptar la configuración de OpenCA para que el formato del certificado de la CA estuviera acorde a lo definido en las políticas de certificados y declaraciones de prácticas de certificación.
- En lo que se refiere al módulo PUB, que representa la interfaz WEB de la PKI, resultó necesario adaptar la interfaz del mismo así como la funcionalidad para proveer los servicios descriptos en las políticas de certificados y declaraciones de prácticas de certificación. Para ello se adaptó la configuración y se modificaron scripts escritos en PERL a través de los cuales se implementan las operaciones de la PKI.

Todos aquellos puntos de las políticas de certificados y declaraciones de prácticas de certificación a implementar que no se cubran desde la funcionalidad del producto pueden implementarse a través de procedimientos.

Es necesario que la CA y sus RAs se comuniquen a través de algún mecanismo para poder llevar a cabo la emisión de certificados: la RA debe aprobar los requerimientos, posteriormente a la verificación de identidad, y luego enviar dichos requerimientos aprobados a la CA para que la misma emita los certificados, luego de lo cual la CA debe enviar los certificados al sitio público de la PKI para que están disponibles, tanto para el usuario que los requirió como para la comunidad entera. Al estar la CA offline por razones de seguridad, resulta de vital importancia establecer la forma en que va a comunicarse con las RAs subordinadas. OpenCA permite definir los puntos de intercambio de datos entre la CA y las RAs. Ello se realiza a través de los nodos. El nodo es la interfaz de OpenCA que se comunica con la Base de Datos y que se encarga de implementar las operaciones de importanción y exportación. En la PKI se configuraron diferentes reglas para la sincronización de los nodos de los diferentes niveles de la jerarquía (nodo de la CA y nodo de la RA e interfaz pública), de manera tal de posibilitar el intercambio de datos entre los componentes a través de un medio off-line.

3.3- Política de certificados definida

La política de certificados y declaración de práctica de la certificación definida para PKIGrid UNLP puede verse adjunta a este documento en la sección de Referencias Bibliográficas. Los puntos más importantes que nos servirán de referencia en este trabajo son:

1. Todos los certificados emitidos por la CA entran en el perfil Internet PKI (PKIX) para certificados X.509 como se encuentra definido por la RFC 3280. La CA solo emite certificados X.509 versión 3.
2. La CA de la UNLP PKIGrid emite certificados a miembros de la comunidad académica argentina para personas naturales, hosts administrados por la organización solicitante y servicios provistos en un host administrado por una organización competente.
3. Los certificados emitidos deben cumplir con las siguientes extensiones:
   Certificados para persona natural:

▪ Basic Constraints:	critical, ca: false
▪ Subject Key Identifier:	hash
▪ Authority Key Identifier:	keyid
▪ Subject Alternative Name:	Email
▪ Key Usage:	critical, digitalSignature, nonRepudiation, KeyEncipherment, dataEncipherment
▪ Extended Key Usage	clientAuth, emailProtection, codeSigning timeStamping
▪ Netscape Cert Type:	SSL Client, S/MIME, Object Signing
▪ Netscape Comment:	STRING
▪ CRL Distribution Points:	URI
▪ Certificate Policies:	OID
▪ Issuer alternative Name:	Email
▪ nsRevocationUrl	URI
▪ nsCaPolicyUrl	URI

Figura 79. CP definida para los certificados de Persona natural.

   Certificados para servidor/servicios:

▪ Basic Constraints:	critical, ca: false
▪ Subject Key Identifier:	Hash
▪ Authority Key Identifier:	Keyid
▪ Subject Alternative Name:	DNS, Email
▪ Key Usage:	critical, digitalSignature, KeyEncipherment, dataEncipherment
▪ Extended Key Usage	serverAuth, clientAuth, timeStamping
▪ Netscape Cert Type:	SSL Server, SSL Client
▪ Netscape Comment:	STRING
▪ CRL Distribution Points:	URI (CRL)
▪ Certificate Policies:	OID
▪ Issuer alternative Name:	Email
▪ nsRevocationUrl	URI
▪ NsCaPolicyUrl	URI

Figura 80. CP definida para los certificados de Servidor/Servicios.

Certificados para la CA:

▪ Basic Constraints:	critical, ca: true
▪ Subject Key Identifier:	hash
▪ Authority Key Identifier:	keyid
▪ Key Usage:	Critical, KeyCertSign, CRLSign
▪ Netscape Comment:	STRING
▪ CRL Distribution Points:	URI
▪ nsRevocationUrl	URI
▪ NsCaPolicyUrl	URI

Figura 81. CP definida para el certificado de la CA.

4. La CA de la UNLP PKIGrid operará un repositorio online seguro que contiene:

- El certificado de CA de la UNLP PKIGrid (disponible en formatos PEM, CRT y DER), y todos los anteriores necesarios para chequear certificados aún válidos,
- Los certificados emitidos por la CA,
- Una lista de Revocación de Certificados (disponible en formatos PEM o DER),
- Una copia de la versión más reciente de este CP/CPS y todas las versiones anteriores,
- Otra información juzgada relevante para el servicio de la CA de la UNLP PKIGrid.
- Un link al repositorio trust anchor de TAGPMA (TACAR, www.tacar.org), donde la raíz de confianza de la CA ha sido previamente publicado.

5. La lista de revocación de certificados (CRL) deberá tener una vida útil máxima de 30 días.
6. La CA de la UNLP PKIGrid DEBE emitir una nueva CRL por lo menos 7 días antes de su vencimiento o inmediatamente luego de haber procesado una revocación, lo que ocurra primero. La CRL DEBE ser publicada inmediatamente después de su emisión.
7. Los siguientes eventos deberán ser registrados:
   - Login / logout / reinicio.
   - Creación y firma de certificados.
   - Revocación de certificados.
   - Temas relacionados con la CRL.
   - Recepción de solicitud de revocación de certificado.
   - Validación de solicitud de certificado de la RA.
   - Exportación de CSR desde RA.
   - Emisión e importación de certificado a LDAP.
   - Revocación de certificado.
   - Temas relacionados con la CRL.
8. Claves de longitud menor a 1024 bits no serán aceptadas. La clave de la CA UNLP PKIGrid es de 2048 bits de longitud.
9. Los OID para algoritmos usados para firma de certificados emitidos por la CA UNLP PKIGrid son:
   - Función hash: id-sha1 1.3.14.3.2.26.
   - Encriptación: rsaEncryption 1.2.840.113612.1.1.1.
   - Firma: sha1WithRSAEncryption 1.2.840.113612.1.1.5.
10. La CA UNLP PKIGrid crea y publica CRLs X.509 v2 completas para todos los certificados emitidos por ella independientemente de la razón de la revocación. La razón de la revocación no deberá estar incluida en la CRL. La CRL deberá incluir la fecha para la cual la próxima CRL deberá ser emitida. Una nueva CRL deberá ser emitida antes de esa fecha si nuevas revocaciones son emitidas. Las extensiones de CRL que deberán estar incluidas son:
    - El Identificador de Clave de Autoridad
    - El número de CRL
    - No se usarán extensiones de campos de CRL.

3.3- Problemas actuales

Actualmente PKIGrid U.N.L.P posee problemas sustanciales en su operatoria así como también precisa de nuevas funcionalidades importantes:

1. **Soporte:** OpenCA tiene una baja mantenibilidad, no provee soporte, la comunidad online es casi nula y la documentación oficial es sumamente escasa e incompleta.
2. **Actualizaciones del software:** Las actualizaciones de OpenCA poseen una instalación compleja, se requiere que las actualizaciones se puedan instalar de una manera sencilla.
3. **Problemas de encoding de acentos:** Las palabras con acentos y caracteres especiales no se muestran de una manera correcta en la web.
4. **Soporte de exploradores:** OpenCA solo posee soporte para Mozilla Firefox para solicitar un certificado. Se requiere que PKIGrid U.N.L.P pueda ser utilizado en cualquier explorador moderno.
5. **Falta de emails de vencimiento de certificados:** Se requiere que se envíen alertas por email cuando se aproxime la fecha de vencimiento del certificado.

3.4- Solución propuesta

Para poder solucionar los problemas expuestos anteriormente e incorporar nuevas funcionalidades, planteamos las siguientes propuestas.

3.4.1- Migrar la PKI a EJBCA

Todos los problemas enunciados anteriormente se pueden atacar directamente realizando una migración de herramienta. Lo cierto es que OpenCA es una herramienta que no tuvo más actualizaciones desde el año 2013 por lo que se la declara como una herramienta deprecada. Es necesario entonces migrar la PKI a una herramienta actual como lo es EJBCA. EJBCA ataca todos los puntos débiles de OpenCA:

1. **Soporte:** EJBCA posee una mantenibilidad constante en su proyecto, con mejor soporte online y presenta una documentación completa y actualizada.
2. **Actualizaciones de software:** Cuando se realiza una actualización no es necesario re-instalar ni re-configurar EJBCA, simplemente se ejecuta un comando y la actualización es automática.

3. **Problemas de encoding de acentos:** EJBCA no presenta este problema.
4. **Soporte de exploradores:** EJBCA posee soporte para todos los exploradores modernos.
5. **Falta de emails de vencimiento de certificados:** EJBCA presenta una interfaz intuitiva en donde se pueden configurar diversos eventos de emails.

En caso de precisar alguna configuración extra o extender alguna funcionalidad es posible desarrollar plugins para la herramienta EJBCA. Por último, EJBCA presenta una separación de la vista pública y la vista de administración de la CA.

3.4.2 - Incorporar Online Certificate Status Protocol(OCSP)

Las mejores prácticas requieren que sin importar la forma en la que el estado del certificado es mantenido, se debe verificar dicho estado cada vez que alguien lo quiere utilizar. Fallar esa verificación, puede causar que un certificado revocado sea aceptado como válido. Esto significa que para usar una PKI efectivamente, uno debe tener acceso a las CRLs actuales.

La necesidad de consultar una CRL (u otro servicio de estado de certificados) antes de aceptar certificados genera un posible ataque de denial-of-service contra la PKI. Si la aceptación de un certificado falla en la ausencia de una CRL válida disponible, entonces ninguna operación que dependa de la aceptación de certificados puede realizarse.

Una solución alternativa a usar CRLS es el protocolo de validación de certificados conocido como OCSP. OCSP tiene el beneficio primario de requerir menos ancho de banda, y permitir verificación en tiempo real de grandes volúmenes de operaciones así como de operaciones de alto valor.

Otra de las grandes razones para incorporar OCSP es que Mozilla Firefox anunció en el año 2014 que invalidó el uso de las CRL en favor de OCSP.

El Online Certificate Status Protocol (OCSP) es un protocolo de internet utilizado para obtener el estado de revocación de un certificado digital x.509. Está descrito en RFC 6960 y está en los estándares de internet. Fue creado como una alternativa a la CRL (Certificate revocation list). Específicamente para solucionar ciertos problemas asociados con la CRL cuando era utilizada en una infraestructura de clave pública. Los mensajes transmitidos utilizando este protocolo son codificados en ASN.1 y usualmente comunicados mediante HTTP. La naturaleza de “solicitud/respuesta” de estos mensajes derivan a denominar los servidores OCSP como OCSP responders(respondedores).

Comparación entre OCSP y CRL:

Una de las diferencias más importantes entre OCSP y CRL, es que la CRL puede ser almacenada temporalmente de manera local, para realizar consultas sin conexión. Para OCSP es necesario tener una conexión activa con el OCSP responder. Si bien esto puede parecer una ventaja de CRL sobre OCSP hay que tener en cuenta que mientras más tiempo esté la CRL sin actualizarse, menos fiable es la información que podemos obtener de ésta, ya que pueden haberse revocado algunos certificados entre actualización y actualización.
Esto da lugar a una diferencia en la utilización del ancho de banda para efectos exclusivos de consultas por estado de certificados.

Otra diferencia es que la respuesta OCSP contiene menos información que una CRL típica, por lo tanto pone menos peso en la red y en los recursos del cliente.

Ya que la respuesta OCSP tiene menos información que analizar gramaticalmente, las librerías del lado cliente que lo manejan pueden ser menos complejas que aquellas que manejan CRLs.

OCSP da a conocer al que responde que un host de red en particular utiliza un certificado determinado en un momento determinado . OCSP no impone el cifrado , por lo que otras partes puedan interceptar esta información.

Implementación básica de una PKI con OCSP:

1. Alicia y Pedro tienen claves públicas de certificados emitidas por Juan, la autoridad certificadora(CA).
2. Alicia desea realizar una transacción con Pedro, le envía entonces su certificado de clave pública.
3. Pedro preocupado que la clave privada de Alicia pudiera haber sido comprometida, crea una solicitud OCSP, que contiene el número de serie del certificado de Alicia y se lo envía a Juan.
4. El respondedor de OCSP de Juan lee el número de serie de la solicitud de Pedro. Entonces el respondedor OCSP usa dicho número para buscar el estado de

revocación del certificado de Alicia. Busca en la base de datos de la CA que mantiene Juan. en este escenario la base de datos de la CA Juan es la única ubicación confiable donde una alteración a un certificado sería registrada.

5. El respondedor OCSP de Juan confirma que el certificado de Alicia todavía es válido, y envía una respuesta OCSP exitosa y firmada hacia Pedro.
6. Pedro verifica criptográficamente la respuesta firmada de Juan. Pedro había almacenado la clave pública de juan hace un tiempo atrás, antes de esta transacción. Pedro utiliza la clave pública de Juan para verificar la respuesta recibida.
7. Pedro completa la transacción con Alicia.

Detalles del protocolo:

Un respondedor OCSP (un servidor típicamente ejecutado por el emisor de certificados) puede retornar una respuesta firmada, significando que el certificado especificado es "good"(bueno),"revoked"(revocado),"unknown"(desconocido). Si no puede procesar el requerimiento, puede devolver un código de error.

El formato de la solicitud OCSP soporta extensiones adicionales. Esto permite una personalización muy profunda para un esquema PKI particular.

OCSP puede ser vulnerable a ataques de repetición, donde una respuesta "good", firmada, es capturada por un intermediario malicioso y repetida al cliente más adelante después de que el sujeto del certificado pueda haber sido revocado. OCSP resuelve esto agregando un número arbitrario que sólo puede ser utilizado una vez en la solicitud que debe incluirse en la respuesta correspondiente. Sin embargo, ya que la mayoría de los OCSP responders y clientes no tienen soporte para esta extensión o no la utilizan y las autoridades certificadoras emiten respuestas con períodos de validez de múltiples días, el ataque por repetición es una amenaza mayor a este sistema de validación.

OCSP puede dar soporte a más de un nivel de CA. Las solicitudes OCSP pueden ser encadenadas entre respondedores pares para emitir una consulta a la CA emisora apropiada para el certificado sujeto, con los respondedores validando la respuesta entre unos y otros contra la CA raíz utilizando sus propias solicitudes OCSP.

La clave utilizada para firmar una respuesta no necesita ser la misma que fue utilizada para firmar el certificado. El emisor del certificado puede delegar otra autoridad para ser el OCSP responder. En este caso, el certificado del respondedor (el que es utilizado para firmar la respuesta) debe ser emitido por el emisor del certificado en cuestión, y debe incluir una extensión que lo marque como una autoridad firmante OCSP(SA).

Privacidad:

La comprobación de OCSP crea una preocupación de seguridad para algunos usuarios, ya que requiere que el cliente contacte a un tercero (aunque este sea una parte de confianza por el proveedor de software del cliente) para confirmar la validez del certificado. OCSP Stapling es una forma de verificar la validez sin informar comportamiento de navegación a la CA.

OCSP stapling:

En un escenario con stapling, el poseedor del certificado consulta al servidor OCSP directamente, en intervalos regulares, obteniendo una respuesta OCSP firmada con fecha. Cuando los visitantes del sitio intentan conectarse a este, esta respuesta es incluida("stapled/engrapada") con el saludo TLS/SSL mediante la extensión Certificate Status Request(Solicitud de estado de certificado). Si bien puede parecer que permitir al operador del sitio controlar las respuestas de verificación podría permitir que un sitio fraudulento emitir una verificación falsa para un certificado revocado, la respuesta stapled o engrapada no puede ser formada ya que necesitan ser firmadas directamente por la autoridad certificadora(CA), no el servidor. Si el cliente no recibe una respuesta engrapada. Simplemente se contactara al servidor OCSP por sí mismo. SIn embargo si el cliente recibiera una respuesta engrapada invalida abortaría la conexión.

El único riesgo que aumenta utilizando OCSP stapling es que la notificación de revocación para un certificado podría ser retrasada hasta que la ultima respuesta OCSP firmada expire.

Como resultado, los clientes continúan teniendo una garantía verificable de la autoridad

certificado, que el certificado es actualmente válido(o lo fue recientemente), pero no necesitan contactar individualmente al servidor OCSP. Esto significa que el esfuerzo de la carga del recurso recae en el poseedor del certificado. También significa que el software de cliente ya no necesita informar hábitos de navegación a una tercera parte.

El rendimiento en general presenta una mejora, cuando el cliente va a buscar la respuesta OCSP directamente a la CA, usualmente involucra la búsqueda del nombre de dominio del servidor OCSP de la CA en el servidor DNS, así como también establecer una conexión con el servidor OCSP. Cuando el engrapado OCSP es utilizado, la información del estado del certificado es enviada al cliente a través de un canal ya establecido, reduciendo la sobrecarga y por lo tanto mejorando el rendimiento.

Críticas:

Revocación basada en OCSP no es una técnica efectiva para mitigar el compromiso de una clave privada de un servidor HTTP. Un atacante que ha comprometido dicha clave típicamente necesita estar en una posición de man-in-the-middle(intermediario) en la red para abusar el uso de dicha clave e impersonar al servidor.
Un atacante en esta posicion, tambien esta en posicion para interferir con las consultas del cliente OCSP. Porque la mayoría de los clientes simplemente ignoran OCSP si el tiempo de respuesta de la consulta es demasiado largo. OCSP no es una forma confiable de mitigar el compromiso de una clave privada de un servidor HTTP.
La extensión MustStaple de TSL en un certificado puede requerir que el certificado sea verificado por una respuesta Stapled OCSP, mitigando este problema. OCSP también se mantiene como una defensa válida contra situaciones donde el atacante no utilice un man-in-the-middle.

Soporte de navegadores:

- Internet explorer está construido en CryptoAPI de Windows OS y soporta comprobaciones OCSP
- Todas la versiones de Mozilla Firefox soportan comprobación OCSP.
- Safari en MAC OS X soporta comprobaciones OCSP. Está habilitada por defecto a partir de MAC OS X(LION).
- Versiones de Opera de 8.0 en adelante soportan comprobaciones OCSP.

Google chrome es un caso especial. Google ha deshabilitado las comprobaciones OCSP por defecto en 2012, debido a problemas de latencia y privacidad. En su lugar utiliza su propio sistema de actualizaciones para enviar certificados revocados al navegador.

4- Instalación, adaptación y migración de PKIGrid CA U.N.L.P

En esta sección presentaremos una simulación práctica de la implementación de las soluciones propuestas en la sección anterior. Para esto utilizaremos una máquina virtual con EJBCA instalado la cual simulará ser la nueva PKIGrid UNLP. En esta máquina virtual se realizarán las adaptaciones pertinentes para poder atacar los problemas presentados por OpenCA. Será necesario mantener la funcionalidad vigente de PKIGrid UNLP teniendo en cuenta la política definida para la PKI.

4.1- Perfiles de certificado y de entidad final

EJBCA introduce dos términos utilizados a lo largo de su aplicación: perfiles de certificados y perfiles de entidad final.

Un *perfil de certificado* puede verse como el formato de un certificado, en su mayor parte este define las extensiones son las utilizadas en el certificado. Algunas extensiones son definidas con un valor como ser *CRLDistributionPoint*, otras simplemente se determina su presencia, en otras su valor es específico para cada usuario como ser el *Subject Alternative Name*, en otras se indica si son críticas o no. Un perfil de certificado también especifica si los certificados son publicados y bajo que publicador.
Un *perfil de entidad final* determina cual es la información que puede o debe estar presente cuando una entidad final (ej.: persona natural, servidor, etc) se registra bajo este perfil. Algunos valores pueden ser precargados como ser la *Organization(O)* en el *DN*. Al agregar una entidad final a la PKI, esta debe estar asociada a un perfil de entidad final. Un perfil de entidad final especifica uno o más perfiles de certificados utilizados al momento de generar los certificados.

Para agregar y editar perfiles de certificados en EJBCA ingresamos a la web de administración del EJBCA instalado e ingresamos en la opción del menú "*Certificate*

Profiles". En esta sección se nos muestra un listado de los distintos perfiles de certificados.

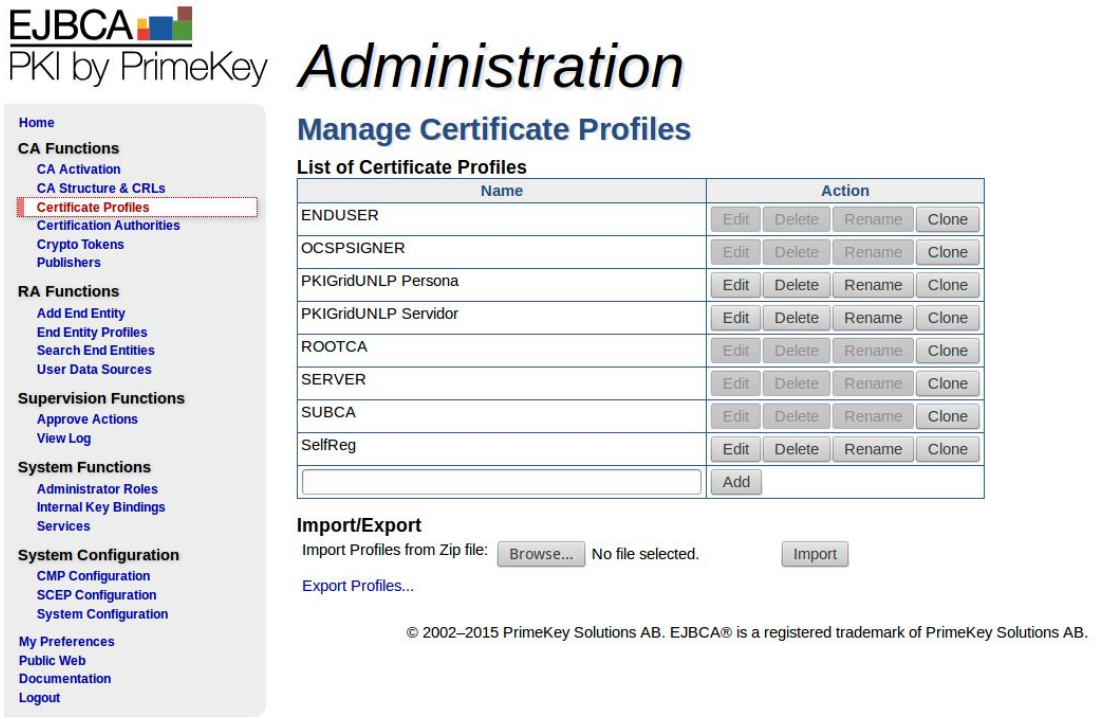


Figura 82. Perfiles de certificados.

Las instrucciones para agregar y editar perfiles de entidad final son las mismas que las enunciadas previamente, solo que debemos ingresar en la opción del menú "*End Entity Profiles*".

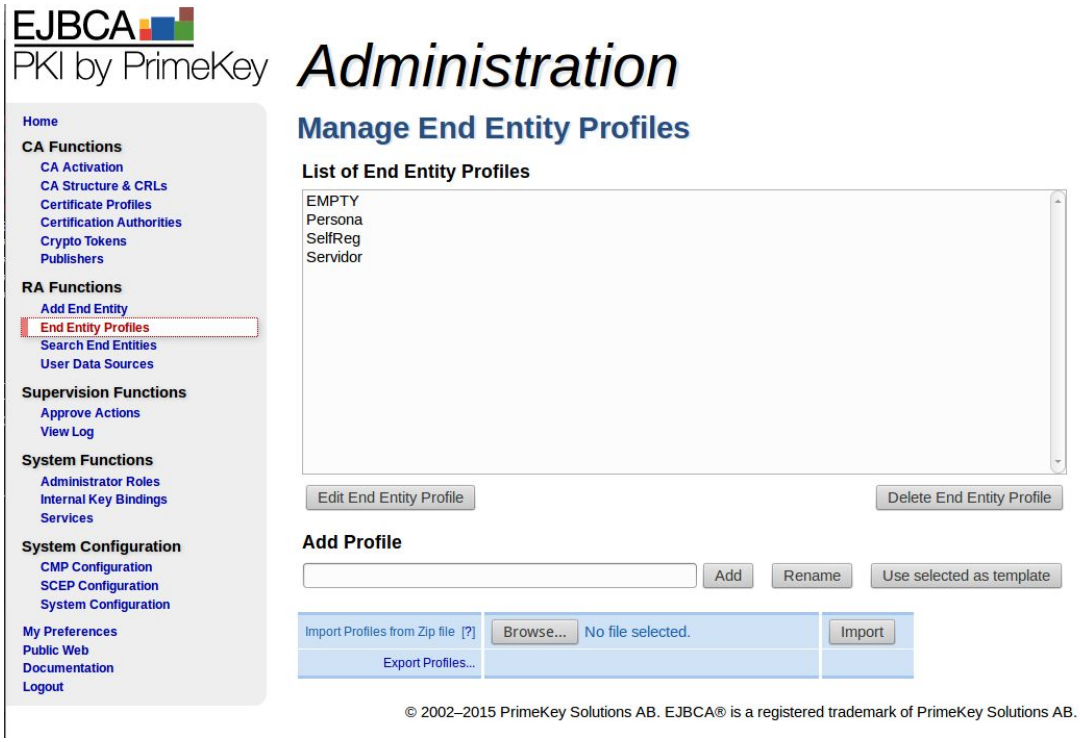


Figura 83. Perfiles de entidad final.

Haciendo uso de estas dos funcionalidades de EJBCA podremos generar los perfiles para poder mantener el formato de certificados definido en la política de certificados de PKIGrid UNLP. Para esto definiremos tres perfiles de certificados:

1. Perfil de certificado para personas naturales.
2. Perfil de certificado para servidor/servicios.
3. Perfil de certificado para la CA.

Primero crearemos el perfil de certificado para personas naturales con las configuraciones indicadas en las siguientes figuras.

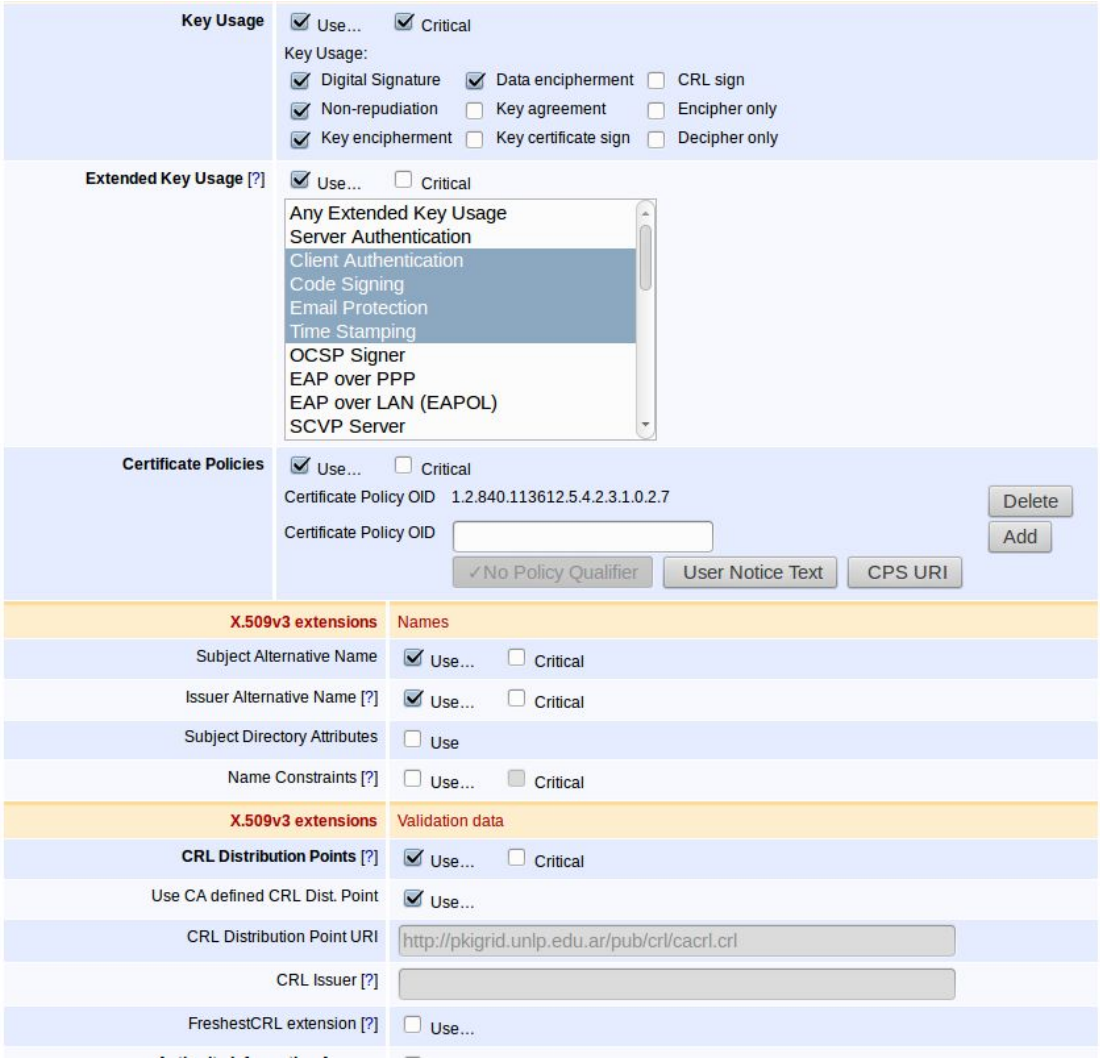
Certificate Profile: PKIGridUNLP Persona

	Back to Certificate Profiles
Certificate Profile Id	2021333706
Type	✓End Entity | Sub CA | Root CA
Available bit lengths	2048 bits 3072 bits 4096 bits 6144 bits 8192 bits
Signature Algorithm	SHA256WithRSA
Validity (*y *mo *d) or end date of the certificate [?]	360d ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2016-10-12 23:51:57+02:00'

Permissions	
Allow validity override [?]	☐ Allow
Allow extension override [?]	☐ Allow
Allow certificate serial number override [?]	☐ Allow No unique index for (issuerDN,serialNumber) on database table 'CertificateData'. "Allow certificate serialnumber override" not allowed.
Allow subject DN override by CSR [?]	☐ Allow
Allow subject DN override by End Entity Information [?]	☐ Allow
Allow Key Usage Override	☐ Allow
Allow back dated revocation [?]	☐ Allow

X.509v3 extensions	
Basic Constraints	☑ Use… ☐ Critical
Authority Key ID	☑ Use
Subject Key ID	☑ Use

X.509v3 extensions	Usages


Key Usage
Use...
Critical
Key Usage:
Digital Signature
Data encipherment
CRL sign
Non-repudiation
Key agreement
Encipher only
Key encipherment
Key certificate sign
Decipher only
Extended Key Usage [?]
Use...
Critical
Any Extended Key Usage
Server Authentication
Client Authentication
Code Signing
Email Protection
Time Stamping
OCSP Signer
EAP over PPP
EAP over LAN (EAPOL)
SCVP Server
Certificate Policies
Use...
Critical
Certificate Policy OID
1.2.840.113612.5.4.2.3.1.0.2.7
Delete
Certificate Policy OID
Add
✓No Policy Qualifier
User Notice Text
CPS URI
X.509v3 extensions
Names
Subject Alternative Name
Use...
Critical
Issuer Alternative Name [?]
Use...
Critical
Subject Directory Attributes
Use
Name Constraints [?]
Use...
Critical
X.509v3 extensions
Validation data
CRL Distribution Points [?]
Use...
Critical
Use CA defined CRL Dist. Point
Use...
CRL Distribution Point URI
http://pkigrid.unlp.edu.ar/pub/crl/cacrl.crl
CRL Issuer [?]
FreshestCRL extension [?]
Use...

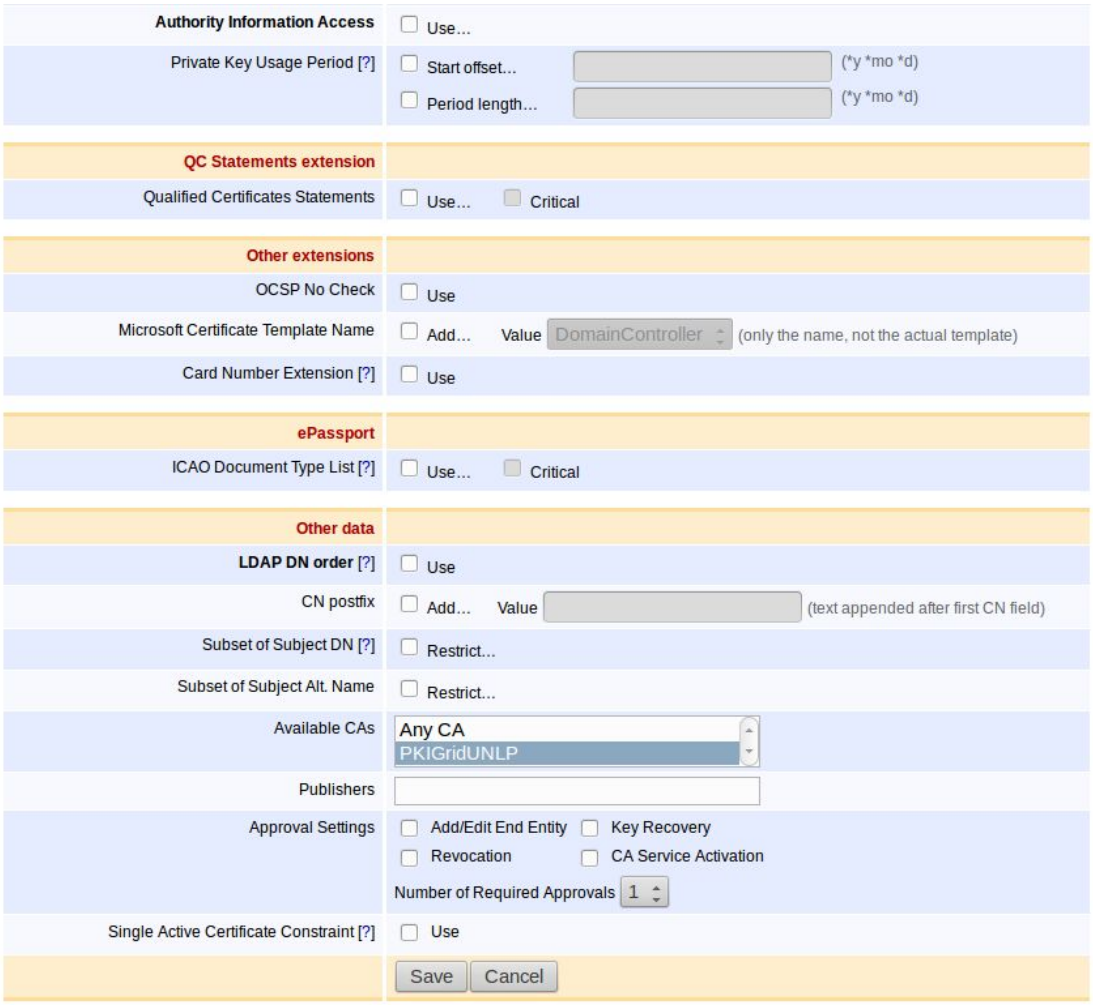


Figura 84. Configuración del perfil de certificado para Persona Natural.

Podremos notar que no figuran las extensiones "*Netscape*" especificadas en la CP/CPS, esto es debido a que son extensiones viejas y deprecadas que no tienen utilidad en el presente.

El siguiente paso para que un usuario sea capaz de registrarse como *Persona Natural* en la web de PKIGrid UNLP es crear el perfil de entidad final. Para esto seguimos las configuraciones indicadas en las siguientes figuras.

End Entity Profile : Persona

		Back to End Entity Profiles
	End Entity Profile Id	1813950237
	Username	
	Password (or Enrollment Code) [?]	Required ☑ Auto-generated English letters and digits of length 8
	Minimum password strength (bits) [?]	0
	Maximum number of failed login attempts [?]	Use : Default = Unlimited Modifiable
	Batch generation (clear text pwd storage)	Use : Default = Required
	End Entity E-mail	☑ Use (Use only the domain part of the address, without the '@' char) ejbca.localdomain ☑ Required ☑ Modifiable
	Directives	
	Reverse Subject DN and Subject Alt Name Checks [?]	Use
	Allow merge DN Web Services [?]	Allow
	Subject DN Attributes [?]	
Select for Removal	**Subject DN Attributes**	emailAddress, E-mail address in DN Add
☐	C, Country (ISO 3166)	AR ☑ Required Modifiable
☐	O, Organization	e-Ciencia ☑ Required Modifiable
☐	OU, Organizational Unit	UNLP ☑ Required Modifiable
☐	L, Locality	CeSPI ☑ Required Modifiable

☐	CN, Common name	☑ Required ☑ Modifiable
Remove		
	Other subject attributes	
Select for Removal	**Subject Alternative Name** [?]	RFC 822 Name (e-mail address) Add
Remove		
Select for Removal	**Subject Directory Attributes**	Date of birth (YYYYMMDD) Add
Remove		
	Main certificate data	
	Default Certificate Profile	PKIGridUNLP Persona
	Available Certificate Profiles	ENDUSER OCSPSIGNER PKIGridUNLP Persona PKIGridUNLP Servidor SERVER SUBCA SelfReg
	Default CA	PKIGridUNLP
	Available CAs	Any CA PKIGridUNLP
	Default Token	User Generated
	Available Tokens	User Generated P12 file JKS file PEM file

	Other certificate data	
	Custom certificate serial number [?]	☐ Use
	Certificate Validity Start Time [?]	☐ Use : Value INVALID: ☑ Modifiable (ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2016-10-12 23:53:30+02:00' or days:hours:minutes)
	Certificate Validity End Time [?]	☐ Use : Value INVALID: ☑ Modifiable (ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2016-10-12 23:53:30+02:00' or days:hours:minutes)
	Card number [?]	☐ Use ☐ Required
	Name Constraints, Permitted [?]	☐ Use ☐ Required
	Name Constraints, Excluded [?]	☐ Use ☐ Required
	Custom certificate extension data [?]	☐ Use
	Other data	
	Number of allowed requests	☐ Use : Default = 1
	Revocation reason to set after certificate issuance [?]	☐ Use : Value Active ☐ Modifiable
Delete all	**Send Notification** [?]	☑ Use : Default = ☑ ☐ Required
Delete	Notification Sender	ejbca@ejbca.localdomain
	Notification Recipient	USER
	Notification Events	STATUSNEW STATUSFAILED STATUSINITIALIZED STATUSINPROCESS STATUSGENERATED STATUSREVOKED STATUSHISTORICAL
	Notification Subject	PKIGrid UNLP
	Notification Message	Su nombre de usuario es ${USERNAME} Su password es

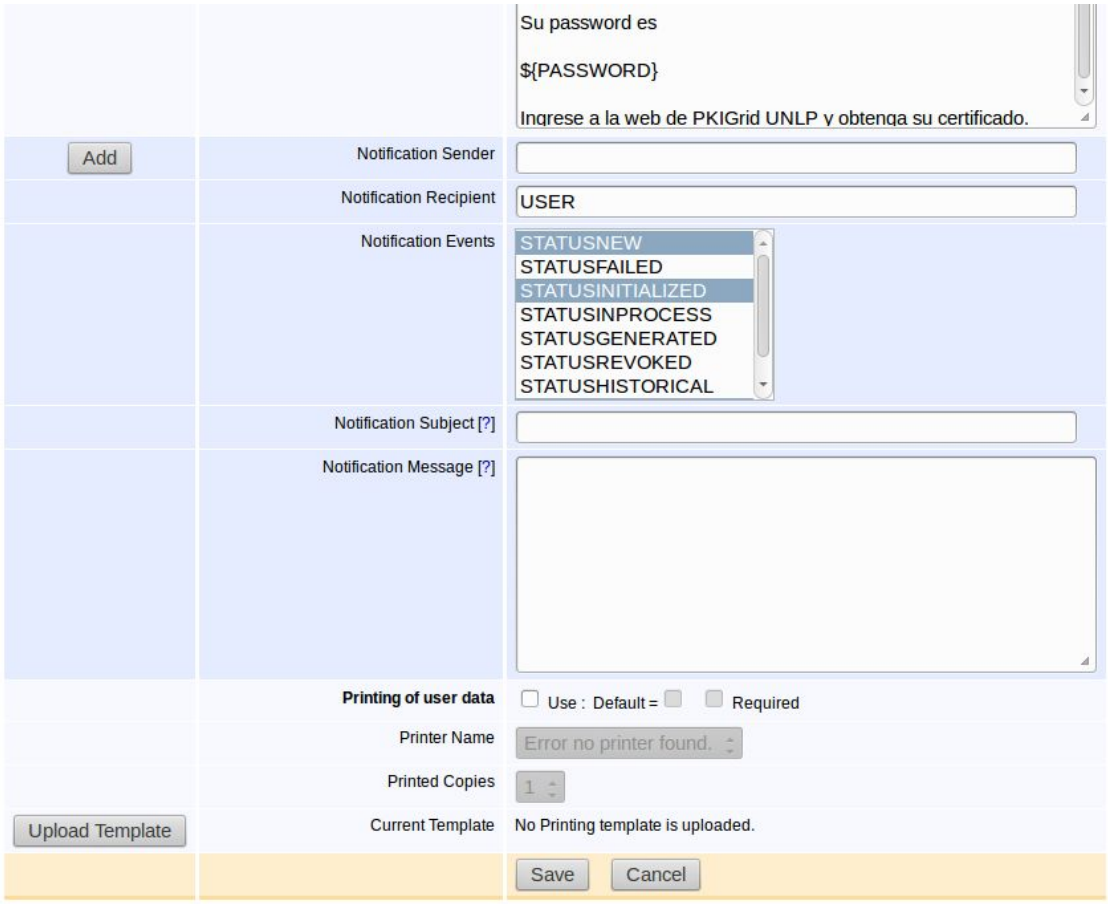


Figura 85. Perfil de entidad final Persona.

Siguiendo con el proceso, ahora deberemos realizar lo mismo pero para los certificados de tipo servidor/servicio. Para esto configuraremos los perfiles de certificado y entidad final como es indicado en las siguientes figuras.

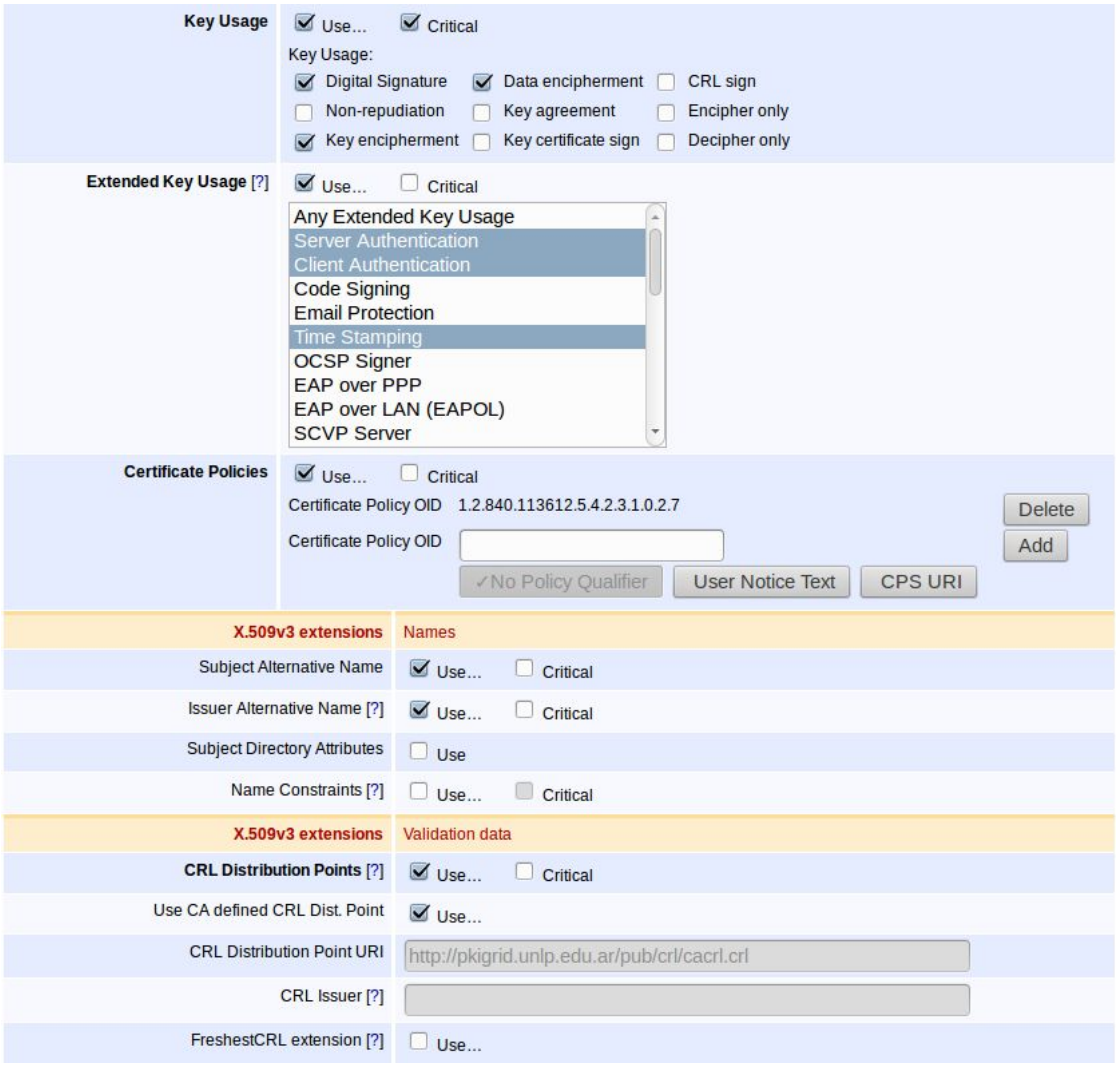
Certificate Profile: PKIGridUNLP Servidor

	Back to Certificate Profiles
Certificate Profile Id	1356637988
Type	✓End Entity | Sub CA | Root CA
Available bit lengths	512 bits 521 bits 1024 bits 1536 bits 2048 bits
Signature Algorithm	SHA256WithRSA
Validity (*y *mo *d) or end date of the certificate [?]	360d ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2016-10-12 23:52:53+02:00'

Permissions	
Allow validity override [?]	☐ Allow
Allow extension override [?]	☐ Allow
Allow certificate serial number override [?]	☐ Allow No unique index for (issuerDN,serialNumber) on database table 'CertificateData'. "Allow certificate serialnumber override" not allowed.
Allow subject DN override by CSR [?]	☐ Allow
Allow subject DN override by End Entity Information [?]	☐ Allow
Allow Key Usage Override	☐ Allow
Allow back dated revocation [?]	☐ Allow

X.509v3 extensions	
Basic Constraints	☑ Use... ☐ Critical
Authority Key ID	☑ Use
Subject Key ID	☑ Use

X.509v3 extensions	Usages


Key Usage
Use...
Critical
Key Usage:
Digital Signature
Data encipherment
CRL sign
Non-repudiation
Key agreement
Encipher only
Key encipherment
Key certificate sign
Decipher only
Extended Key Usage [?]
Use...
Critical
Any Extended Key Usage
Server Authentication
Client Authentication
Code Signing
Email Protection
Time Stamping
OCSP Signer
EAP over PPP
EAP over LAN (EAPOL)
SCVP Server
Certificate Policies
Use...
Critical
Certificate Policy OID
1.2.840.113612.5.4.2.3.1.0.2.7
Delete
Certificate Policy OID
Add
✓No Policy Qualifier
User Notice Text
CPS URI
X.509v3 extensions
Names
Subject Alternative Name
Use...
Critical
Issuer Alternative Name [?]
Use...
Critical
Subject Directory Attributes
Use
Name Constraints [?]
Use...
Critical
X.509v3 extensions
Validation data
CRL Distribution Points [?]
Use...
Critical
Use CA defined CRL Dist. Point
Use...
CRL Distribution Point URI
http://pkigrid.unlp.edu.ar/pub/crl/cacrl.crl
CRL Issuer [?]
FreshestCRL extension [?]
Use...

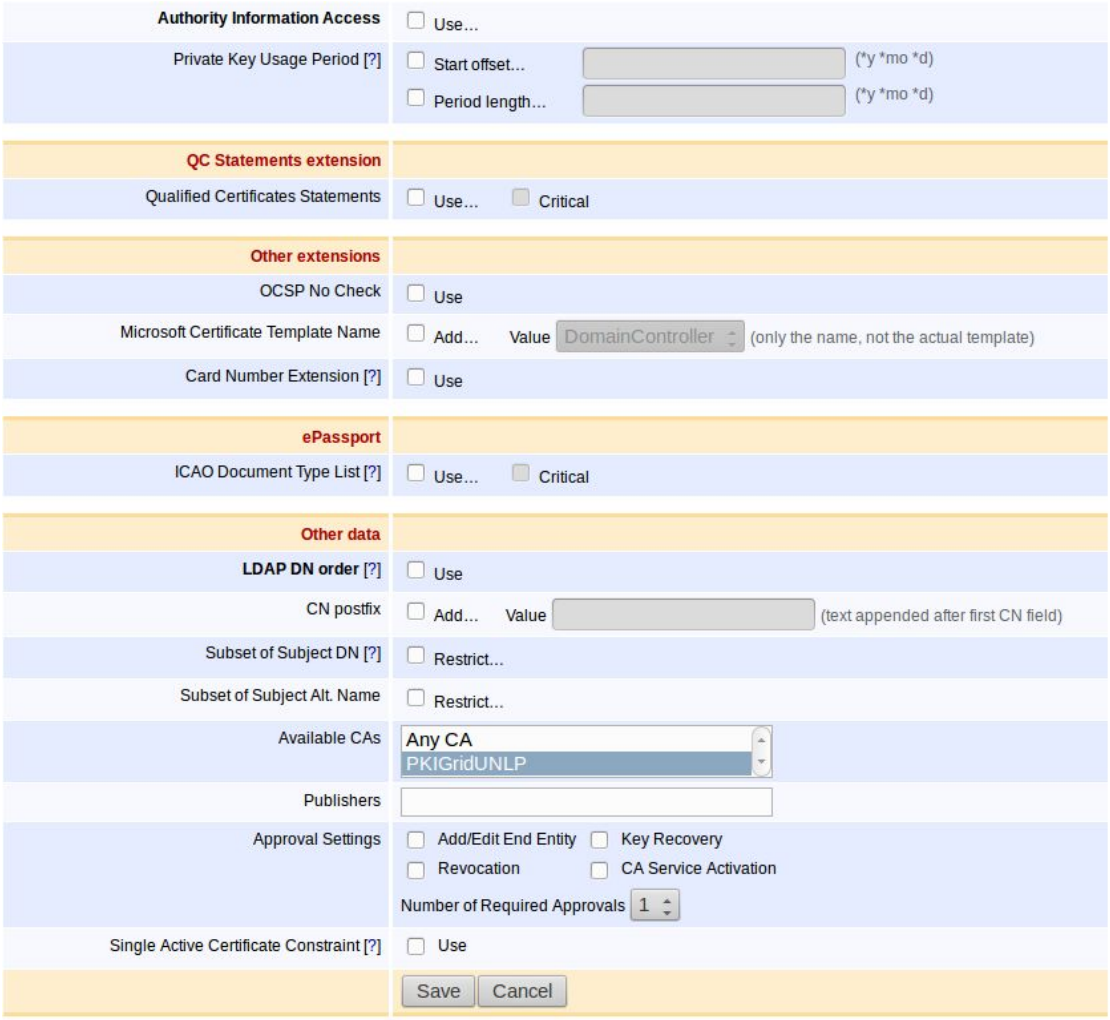


Figura 86. Configuración del perfil de certificado para Servidor/Servicio

End Entity Profile : Servidor

		Back to End Entity Profiles
	End Entity Profile Id	803536773
	Username	
	Password (or Enrollment Code) [?]	Required Auto-generated English letters and digits of length 8
	Minimum password strength (bits) [?]	0
	Maximum number of failed login attempts [?]	Use : Default = Unlimited Modifiable
	Batch generation (clear text pwd storage)	Use : Default = Required
	End Entity E-mail	Use (Use only the domain part of the address, without the '@' char) ejbca.localdomain Required Modifiable
	Directives	
	Reverse Subject DN and Subject Alt Name Checks [?]	Use
	Allow merge DN Web Services [?]	Allow
	Subject DN Attributes [?]	
Select for Removal	**Subject DN Attributes**	emailAddress, E-mail address in DN Add
	C, Country (ISO 3166)	AR Required Modifiable
	O, Organization	e-Ciencia Required Modifiable
	OU, Organizational Unit	UNLP Required Modifiable
	L, Locality	CeSPI Required Modifiable
	unstructuredName, Domain name	

☐	unstructuredName, Domain name (FQDN)	☑ Required ☑ Modifiable
Remove		
	Other subject attributes	
Select for Removal	**Subject Alternative Name** [?]	RFC 822 Name (e-mail address) Add
Remove		
Select for Removal	**Subject Directory Attributes**	Date of birth (YYYYMMDD) Add
Remove		
	Main certificate data	
	Default Certificate Profile	PKIGridUNLP Persona
	Available Certificate Profiles	ENDUSER OCSPSIGNER PKIGridUNLP Persona PKIGridUNLP Servidor SERVER SUBCA SelfReg
	Default CA	PKIGridUNLP
	Available CAs	Any CA PKIGridUNLP
	Default Token	User Generated
	Available Tokens	User Generated P12 file JKS file PEM file

	Other certificate data	
	Custom certificate serial number [?]	☐ Use
	Certificate Validity Start Time [?]	☐ Use : Value INVALID: ☑ Modifiable (ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2016-10-12 23:54:34+02:00' or days:hours:minutes)
	Certificate Validity End Time [?]	☐ Use : Value INVALID: ☑ Modifiable (ISO 8601 date: [yyyy-MM-dd HH:mm:ssZZ]: '2016-10-12 23:54:34+02:00' or days:hours:minutes)
	Card number [?]	☐ Use ☐ Required
	Name Constraints, Permitted [?]	☐ Use ☐ Required
	Name Constraints, Excluded [?]	☐ Use ☐ Required
	Custom certificate extension data [?]	☐ Use
	Other data	
	Number of allowed requests	☐ Use : Default = 1
	Revocation reason to set after certificate issuance [?]	☐ Use : Value Active ☐ Modifiable
Delete all	**Send Notification** [?]	☑ Use : Default = ☑ ☐ Required
Delete	Notification Sender	ejbca@ejbca.localdomain
	Notification Recipient	USER
	Notification Events	STATUSNEW STATUSFAILED STATUSINITIALIZED STATUSINPROCESS STATUSGENERATED STATUSREVOKED STATUSHISTORICAL
	Notification Subject	PKIGrid UNLP
	Notification Message	Su nombre de usuario es ${USERNAME} Su password es

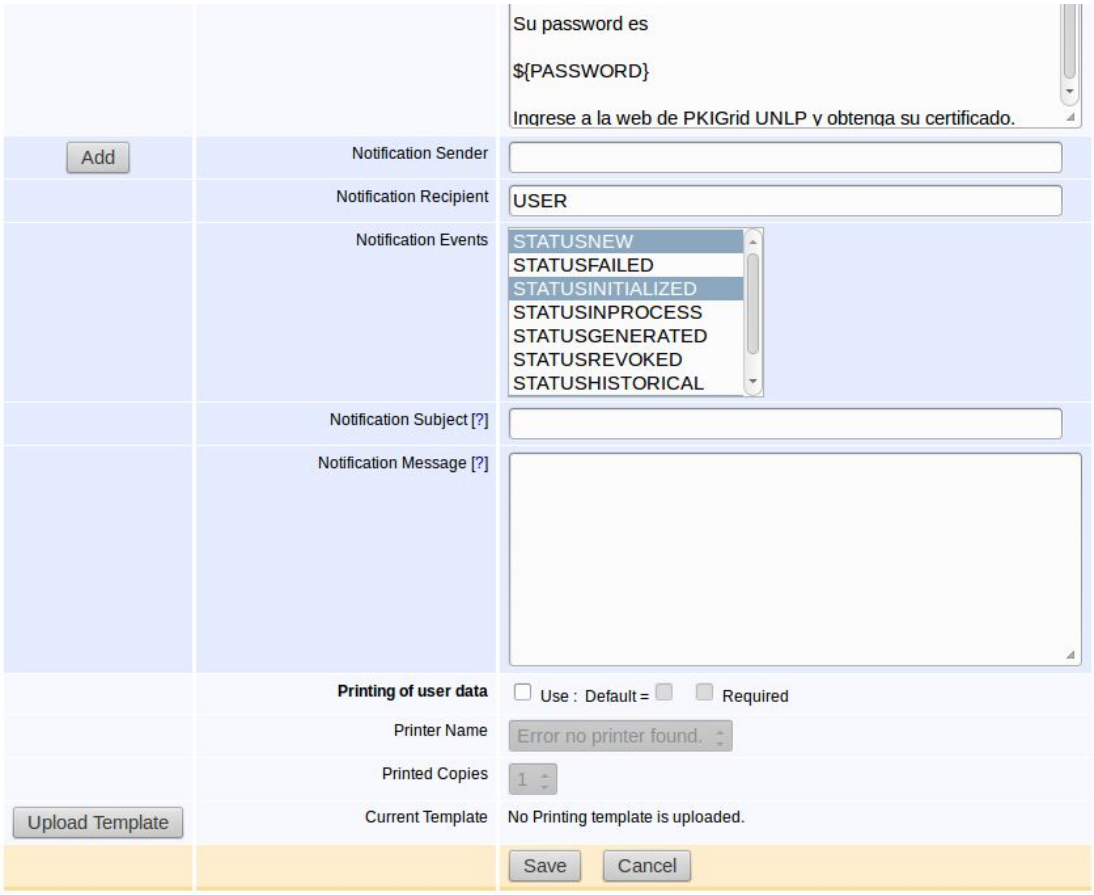


Figura 87. Perfil de entidad final para Servidor/Servicio.

Para crear el perfil de certificado correspondiente a la CA que utiliza PKIGrid UNLP realizamos la siguiente configuración:

Administration

Edit CA

CA Name : PKIGRIDunlpCA

	Back to Certificate Authorities
CA Id	866403837
Type of CA [?]	X509
Signing Algorithm	SHA1WithRSA
Crypto Token [?]	PKIGRIDunlpCA
defaultKey:	defaultKey
certSignKey:	signKey
crlSignKey:	signKey
keyEncryptKey:	defaultKey
hardTokenEncrypt:	defaultKey
testKey:	testKey

Permissions	
Allow validity override [?]	☐ Allow
Allow extension override [?]	☐ Allow
Allow certificate serial number override [?]	☐ Allow No unique index for (issuerDN,serialNumber) on database table 'CertificateData'. "Allow certificate serialnumber override" not allowed.
Allow subject DN override by CSR [?]	☐ Allow
Allow subject DN override by End Entity Information [?]	☐ Allow
Allow Key Usage Override	☐ Allow
Allow back dated revocation [?]	☐ Allow

X.509v3 extensions	
Basic Constraints	☑ Use... ☑ Critical
Path Length Constraint [?]	☐ Add... Value 0
Authority Key ID	☑ Use
Subject Key ID	☑ Use
X.509v3 extensions	Usages
Key Usage	☑ Use... ☑ Critical Key Usage: ☐ Digital Signature ☐ Data encipherment ☑ CRL sign ☐ Non-repudiation ☐ Key agreement ☐ Encipher only ☐ Key encipherment ☑ Key certificate sign ☐ Decipher only
Extended Key Usage [?]	☐ Use... ☐ Critical
Certificate Policies	☐ Use... ☐ Critical

X.509v3 extensions	Names
Subject Alternative Name	☑ Use... ☐ Critical
Issuer Alternative Name [?]	☐ Use... ☐ Critical
Subject Directory Attributes	☐ Use
Name Constraints [?]	☐ Use... ☐ Critical
X.509v3 extensions	Validation data
CRL Distribution Points [?]	☑ Use... ☐ Critical
Use CA defined CRL Dist. Point	☐ Use...
CRL Distribution Point URI	http://www.pkigrid.unlp.edu.ar/pub/crl/cacrl.crl
CRL Issuer [?]	
FreshestCRL extension [?]	☐ Use...
Authority Information Access	☐ Use...
Private Key Usage Period [?]	☐ Start offset (*y *mo *d)

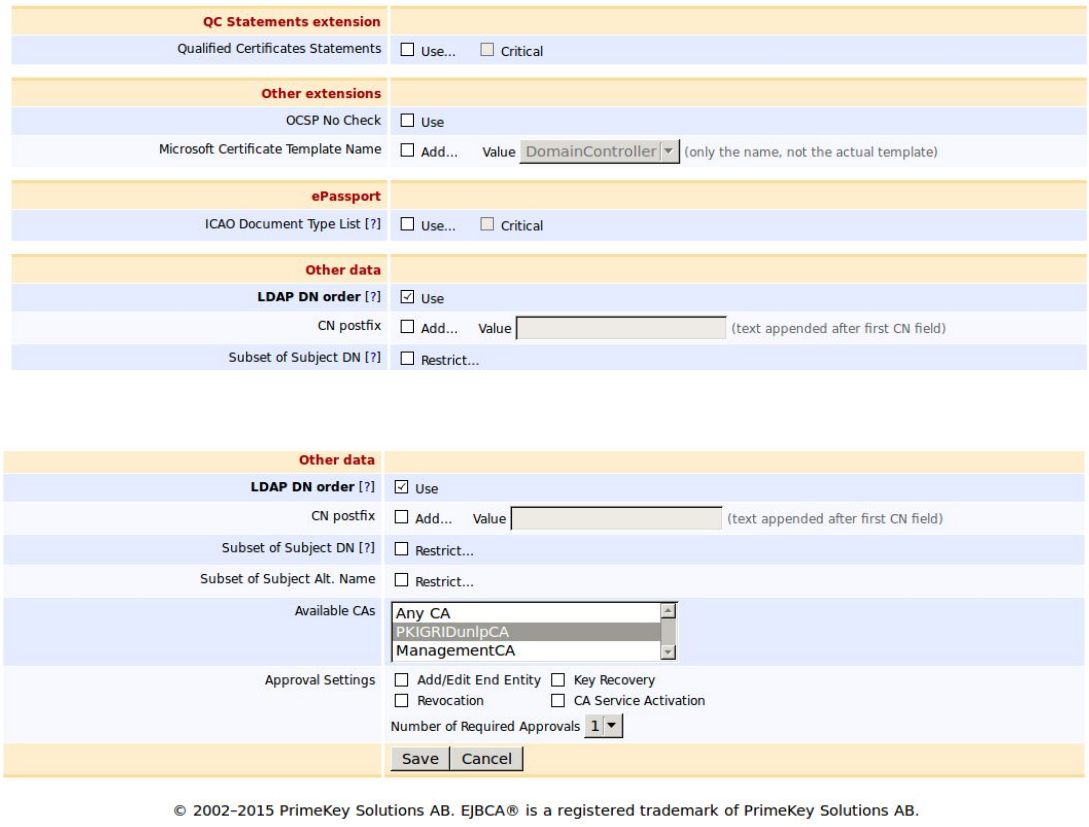


Figura 88. Perfil de certificado de la CA.

4.2- Web pública

EJBCA al ser iniciado como servidor brinda en el puerto 8080 una interfaz web cuyo propósito es poder hacer accesibles al público algunas operaciones de la PKI como ser requerir certificados, ver la CRL, etc.

La implementación actual de PKIGrid UNLP también posee disponible en internet una interfaz de acceso público (http://pkigrid.unlp.edu.ar/). A través de esta es posible realizar las acciones previamente enunciadas, así como también podremos descargar documentación importante y ver información institucional.

Se realizó una adaptación a la web pública de EJBCA para que posea la misma información y funcionalidad que la web pública actual de PKIGrid UNLP, esta adaptación consistió en las siguientes tareas:

- Reorganización de la información.
- Implementación de un diseño UX nuevo y moderno.
- Adaptación de las funciones de EJBCA en el nuevo diseño.
- Configuración de EJBCA para la registración de entidades finales desde la web pública.

Para poder hacer esto posible se utilizó el mecanismo de desarrollo de plugins de EJBCA. Este mecanismo consiste en crear bajo el directorio "*/opt/ejbca-custom/*" una estructura de directorios y archivos la cual sobreescribirá los archivos del EJBCA original luego de re-deployar la aplicación con el comando "`ant clean deploy`". Para este caso específico, se trabajó sobre las carpetas:

- "*publicweb-gui"* la cual posee el módulo de interfaz web pública de EJBCA.
- "*ejbca-renew-war*" la cual posee el módulo de renovación de certificados.
- "*conf*" la cual posee archivos `*.properties` los cuales indican la configuración de la aplicación.

Para las configuraciones de la web pública se editó el archivo "*conf/web.properties*" en donde tuvo que ser habilitado el módulo de renovación de certificados, así como también configurar los perfiles de certificados y entidad finales habilitados para la registración a través de la web pública. Estas configuraciones son:

```
# Allow users to self-register on public web, by entering their information.
# This creates an approval request for the admin.
# Default = false
web.selfreg.enabled=true

# Certificate types to make available for the user
web.selfreg.defaultcerttype=1
web.selfreg.certtypes.1.description=Persona natural
web.selfreg.certtypes.1.eeprofile=Persona
web.selfreg.certtypes.1.certprofile=PKIGridUNLP Persona
web.selfreg.certtypes.2.description=Servidor/Servicio
web.selfreg.certtypes.2.eeprofile=Servidor
web.selfreg.certtypes.2.certprofile=PKIGridUNLP Servidor

# Optional: Instead of asking the user for a username, EJBCA can generate
# the username from a field in the subject DN
web.selfreg.certtypes.1.usernamemapping=CN
web.selfreg.certtypes.2.usernamemapping=UNSTRUCTUREDNAME
```

```
# Deploy the request browser certificate renewal web application and show a
# link to it from the EJBCA public web.
# Default = false
web.renewalenabled=true
```

La restructuración y reorganización de la información tuvo como objetivo poseer una mayor facilidad para el usuario final de encontrar información. Esta es la estructura armada:

- **Inicio:** Contiene información institucional y datos de contacto.
- **Certificados > Requerir:** Permite requerir un certificado de usuario y de servidor según los perfiles configurados desde la web del administrador.
- **Certificados > Mi certificado:** Permite generar y descargar/instalar el certificado requerido (una vez aceptado desde la administración). También permite requerir la renovación del mismo.
- **Certificados > Certificado Raíz:** Permite descargar e instalar el certificado de la CA.
- **Información:** Contiene todos los documentos PDF (CP/CPS, Tutoriales, Formularios/notas, Obligaciones, Buenas prácticas, Administrador de documentos, Cómo crear una RA).

Para la implementación de sus vistas EJBCA utiliza la tecnología JSP. Se decidió mantener este lineamiento e implementar el nuevo diseño sobre las vistas existentes. Para la estructura HTML y CSS se utilizó como base una plantilla open-source la cual utiliza *Twitter Bootstrap,* un framework para el diseño web HTML y CSS mundialmente reconocido.

A continuación podremos ver en las siguiente figuras la nueva web pública de PKIGrid UNLP.

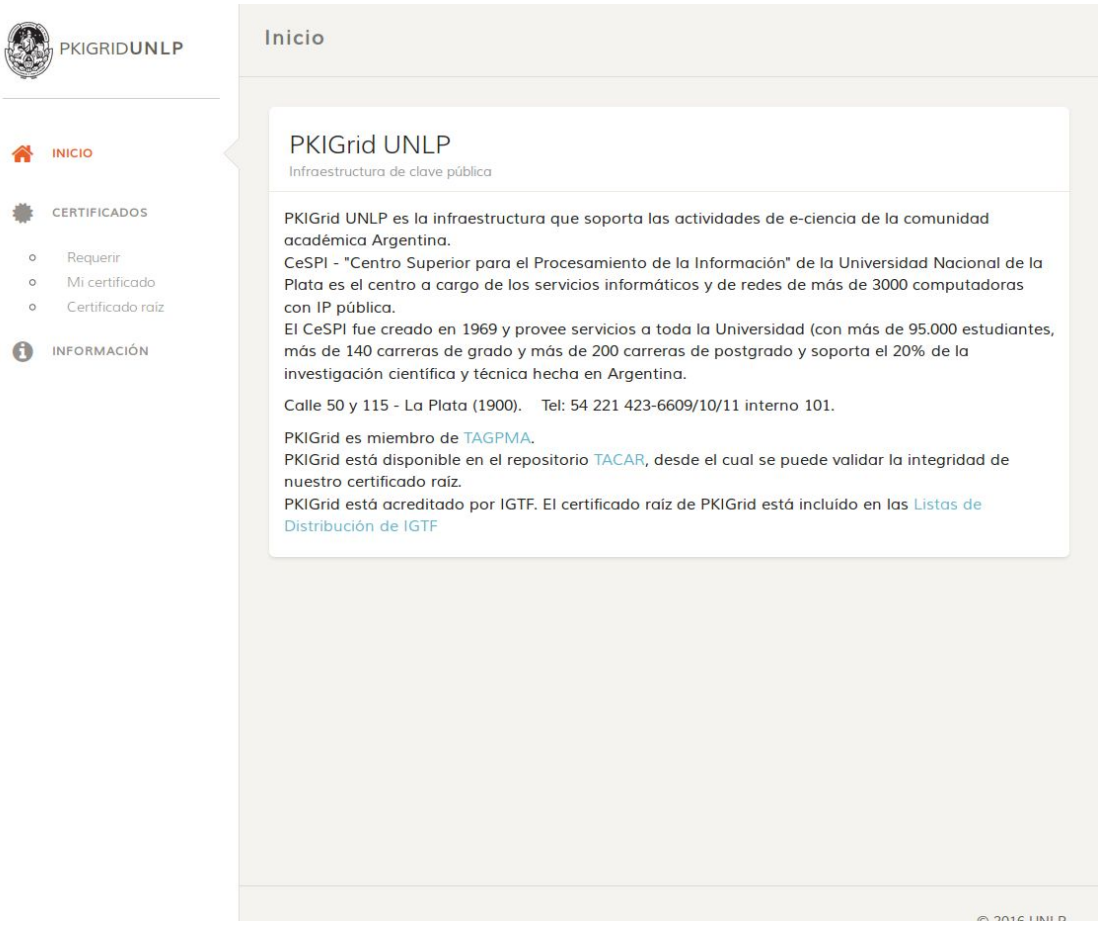


Figura 89. Página de inicio.

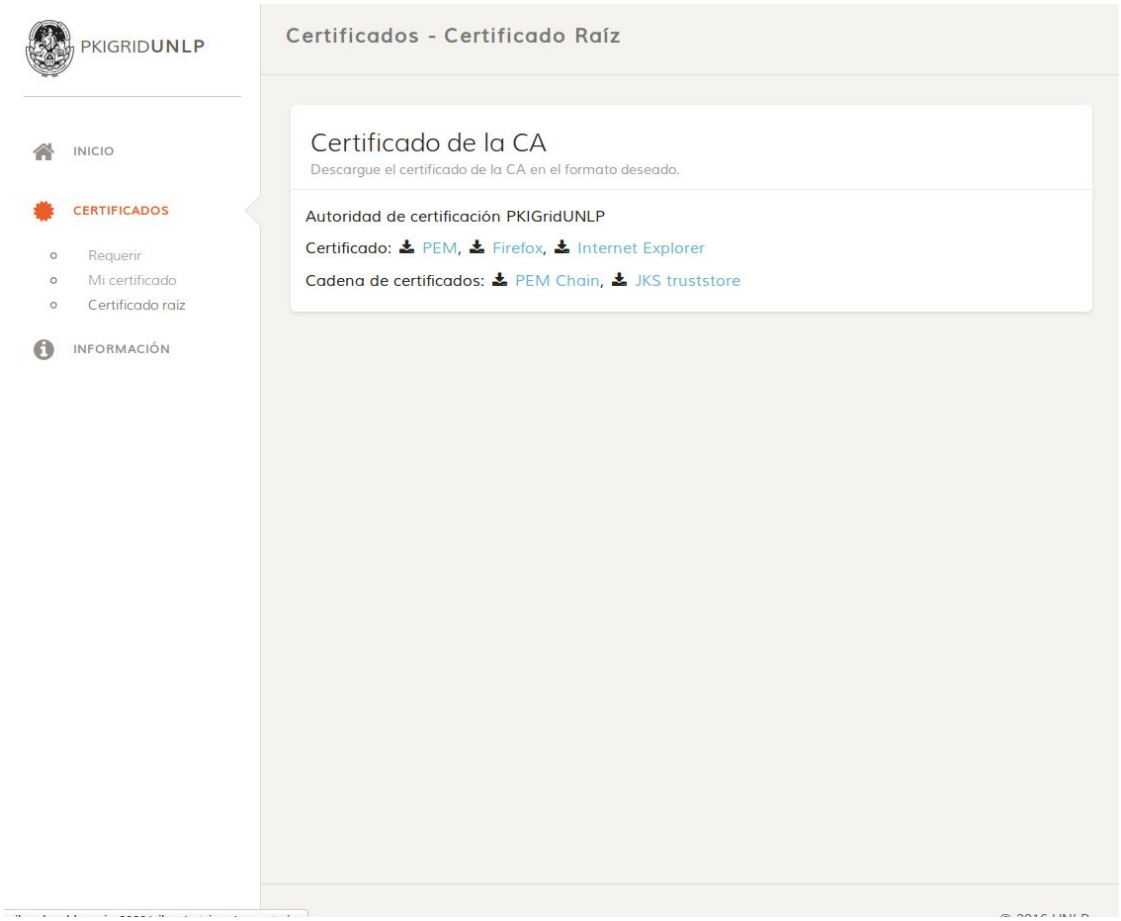


Figura 90. Página de obtención del certificado raíz.

Figura 91. Página de información.

A modo de ejemplo requerimos un certificado para persona natural. Primero debemos ingresar a la web pública y ingresar en la opción del menú "*Certificados* > *Requerir*", seleccionamos "*Persona Natural*" y hacemos click en siguiente.

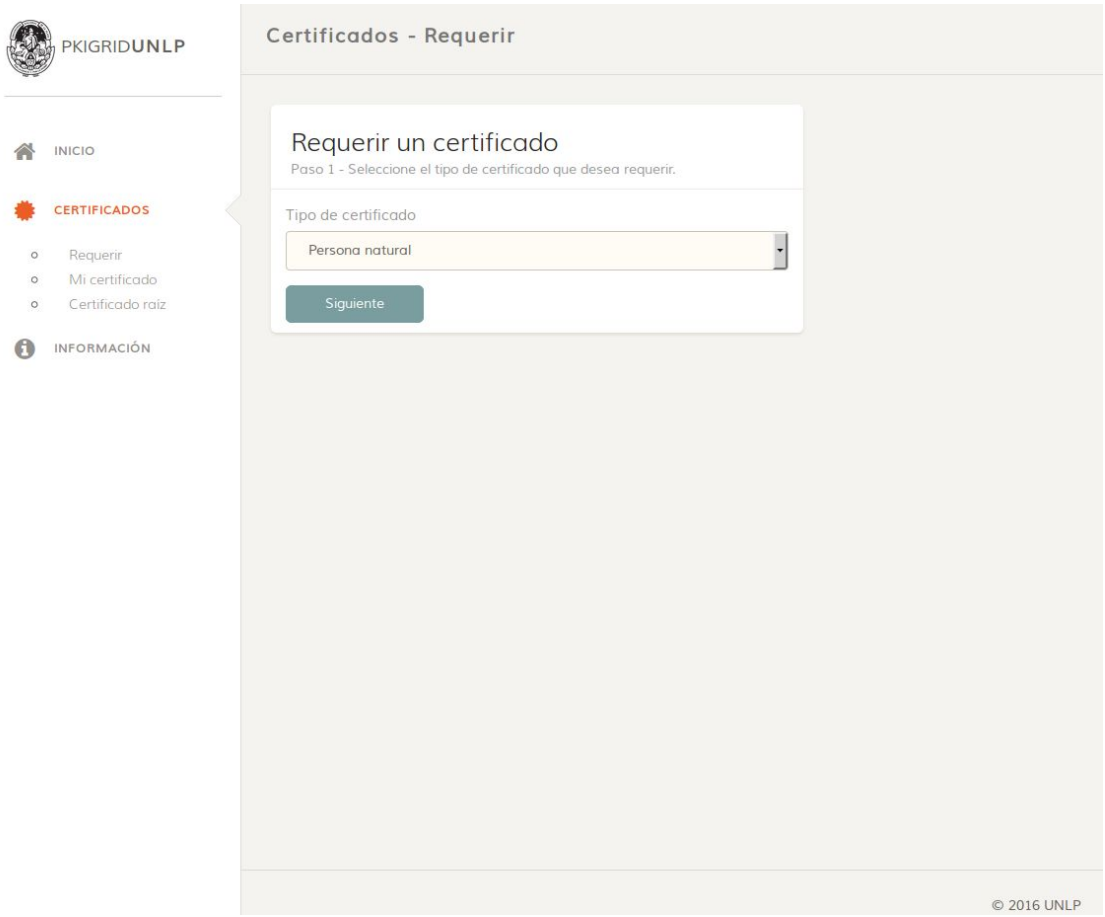


Figura 92. Página para requerir un certificado.

Siguiendo con el proceso, completamos nuestros datos personales en el formulario que se nos presenta (nombre y email) y hacemos clic en Requerir certificado.

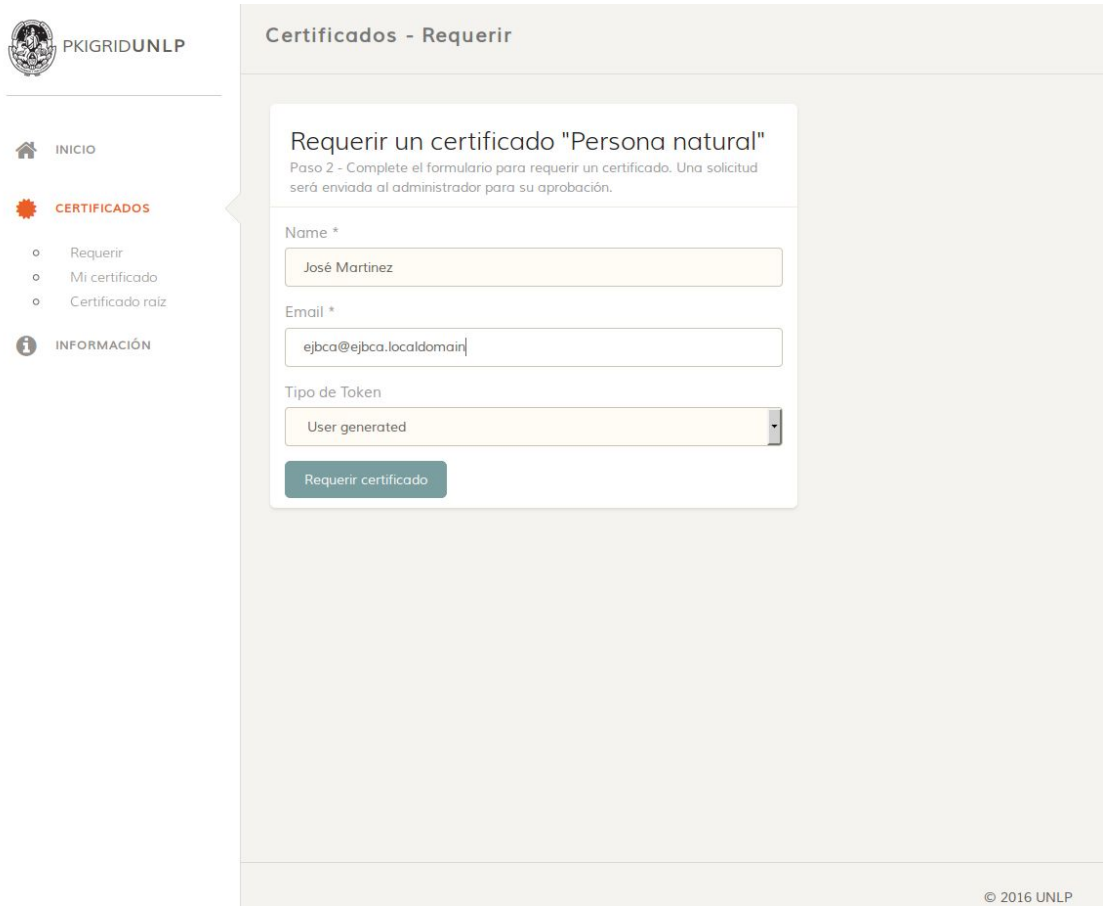


Figura 93. Formulario para requerir un certificado de persona natural.

Se nos presenta una pantalla de confirmación de la solicitud de certificado enviada con las instrucciones a seguir a partir de aquí.

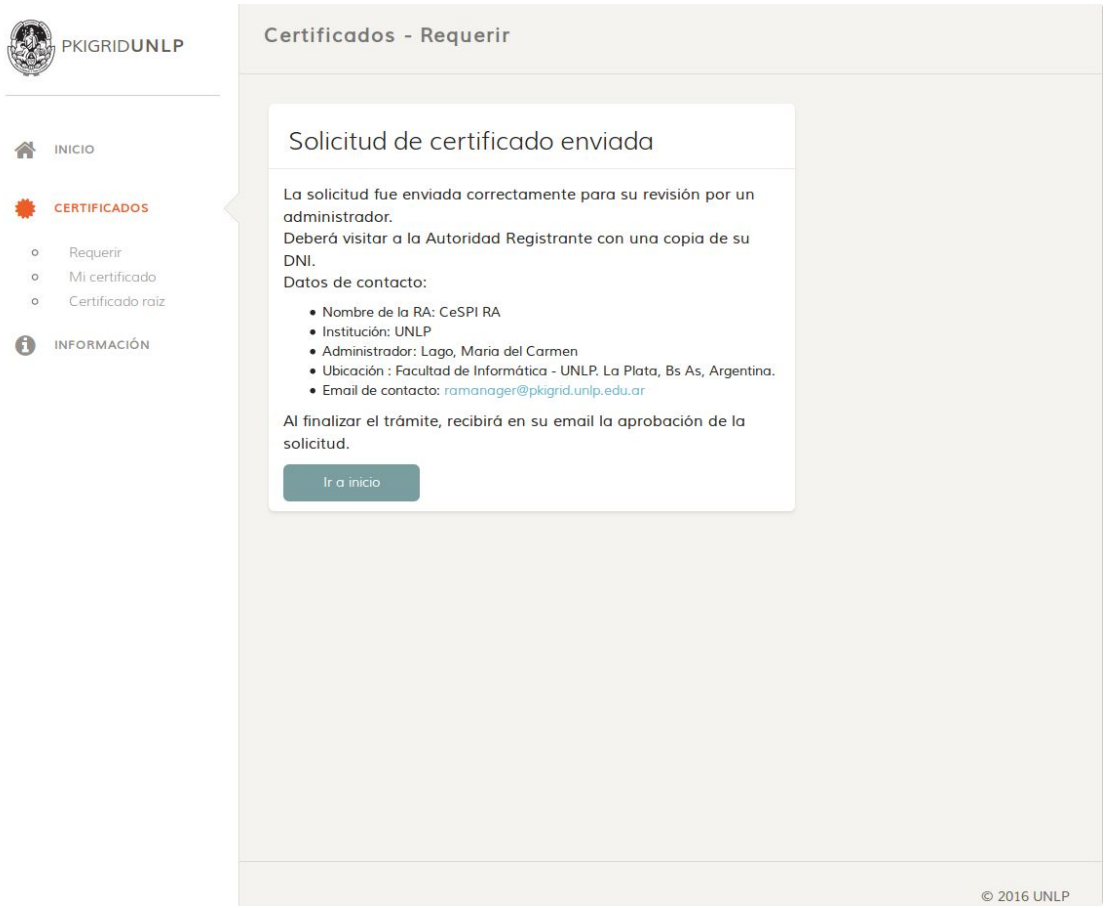


Figura 94. Confirmación de solicitud enviada.

Una vez enviada la solicitud, un operador de la PKI deberá verificar los datos de la persona que requiere el certificado y aceptar la solicitud desde la web de administración de EJBCA, ingresando en la opción del menú "*Approve Actions*".

Figura 95. Solicitudes pendientes.

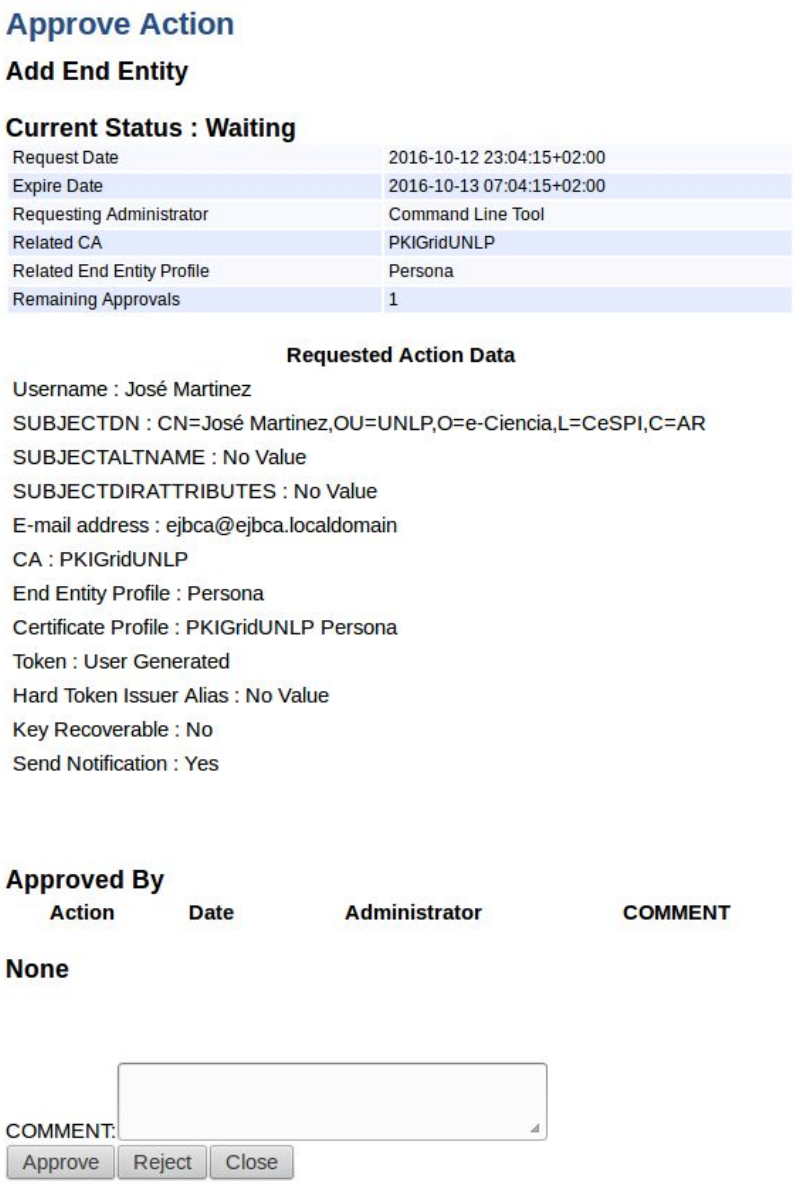


Figura 96. Aprobación de la solicitud de certificado.

Una vez aprobada la solicitud de certificado, se enviará un email a la persona que lo requirió con los datos usuario y contraseña necesarios para obtener el certificado a través de la web pública en la opción de menú "*Certificados > Mi certificado*".

Seguimos los pasos indicados en las pantallas y obtendremos como resultado nuestro certificado instalado en nuestro navegador en conjunto con el certificado de la CA.

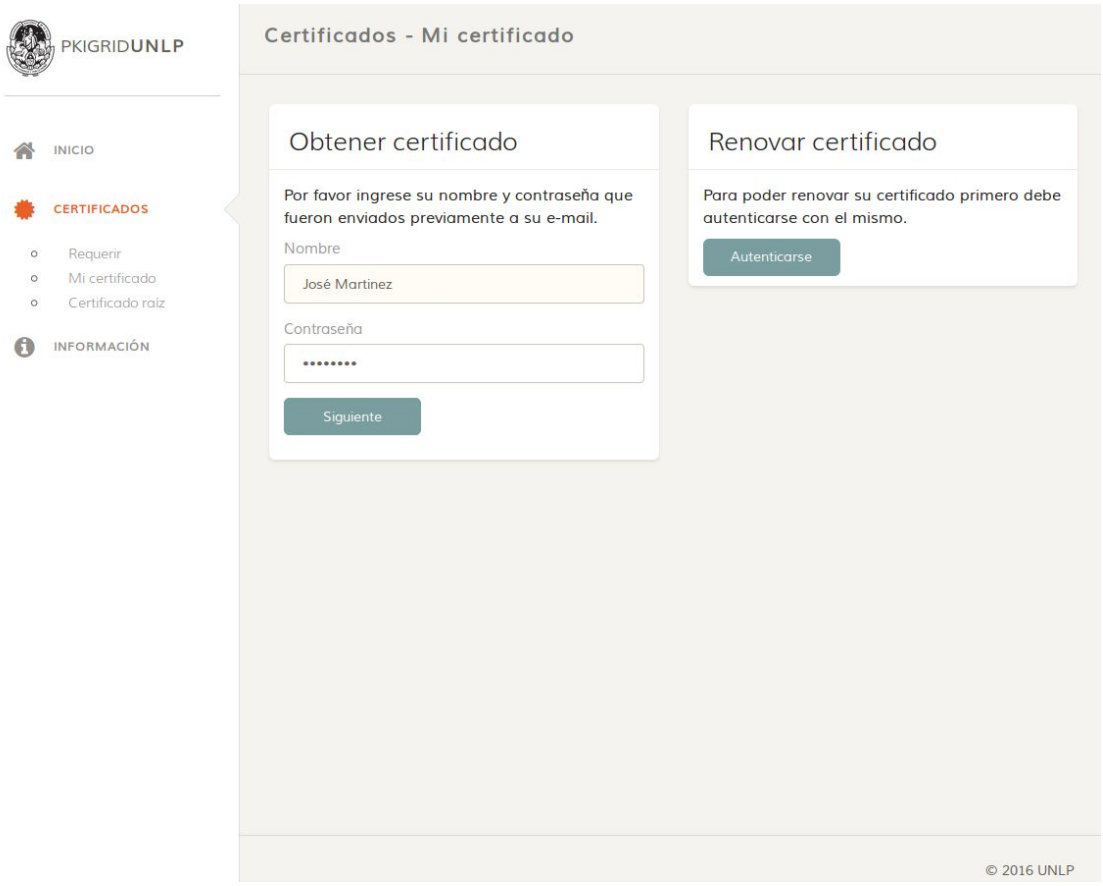


Figura 97. Obtención del certificado.

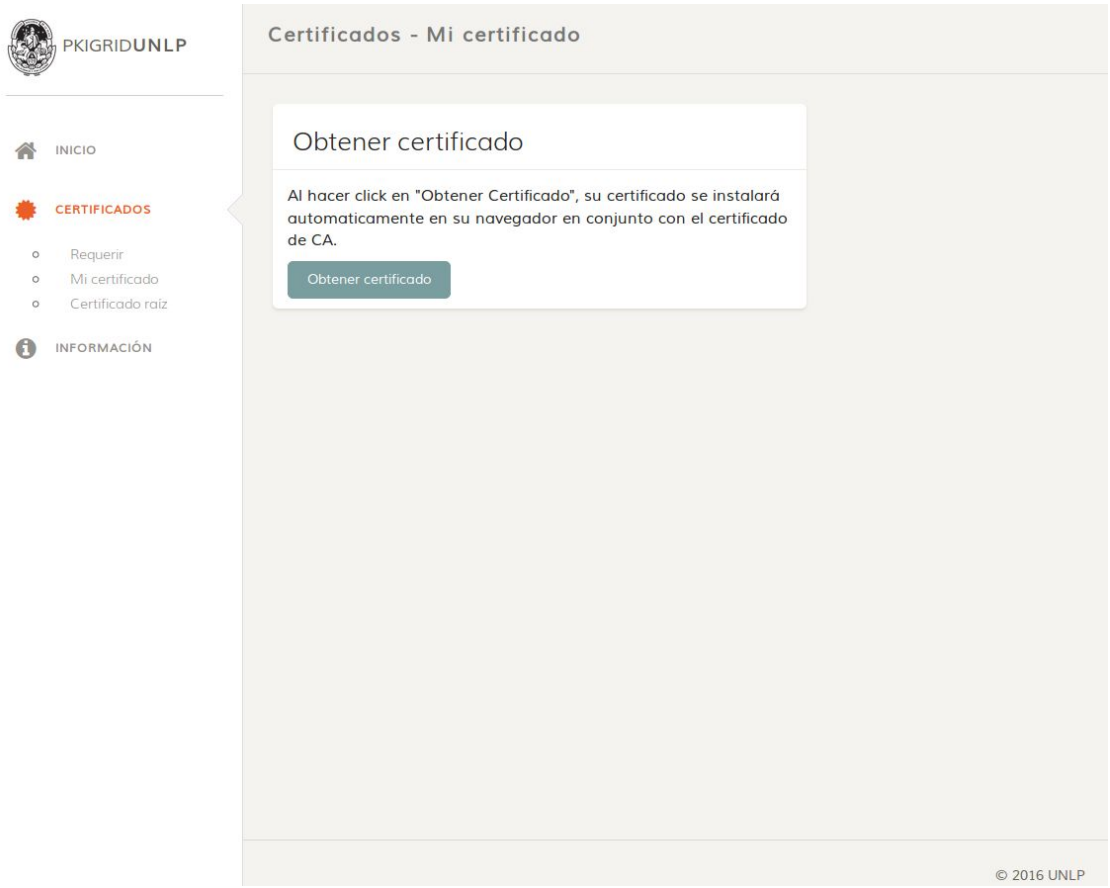


Figura 98. Generación del certificado.

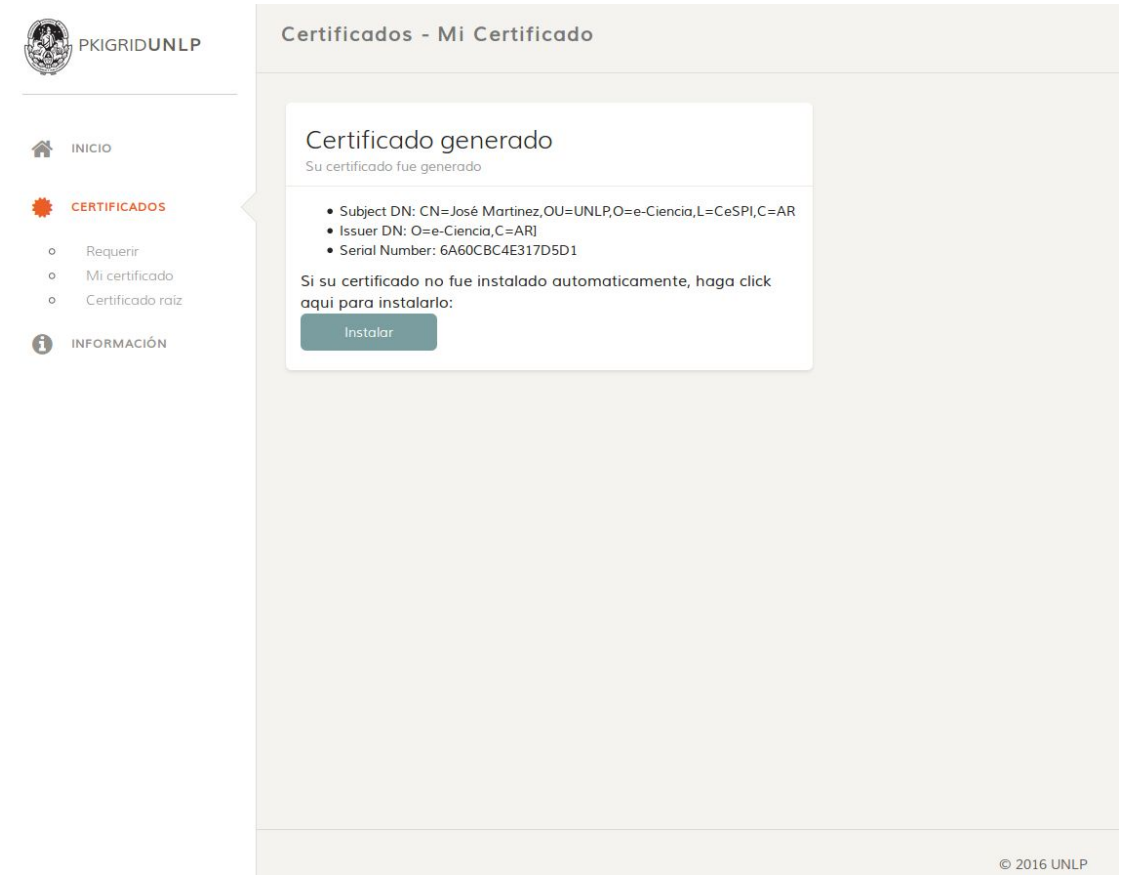


Figura 99. Confirmación de la generación del certificado.

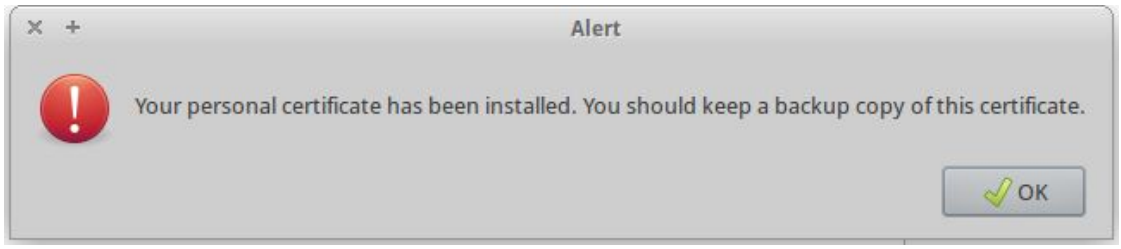


Figura 100. Confirmación de certificado instalado en Firefox.

Podemos notar que la persona utiliza un nombre con acento y este es mostrado sin problemas en la interfaz.

Los pasos necesarios para generar un certificado de tipo Servidor son los mismos que los enunciados previamente. El único cambio es que en la pantalla "*Certificados* > *Requerir*" debemos seleccionar la opción "Servidor/Servicio".

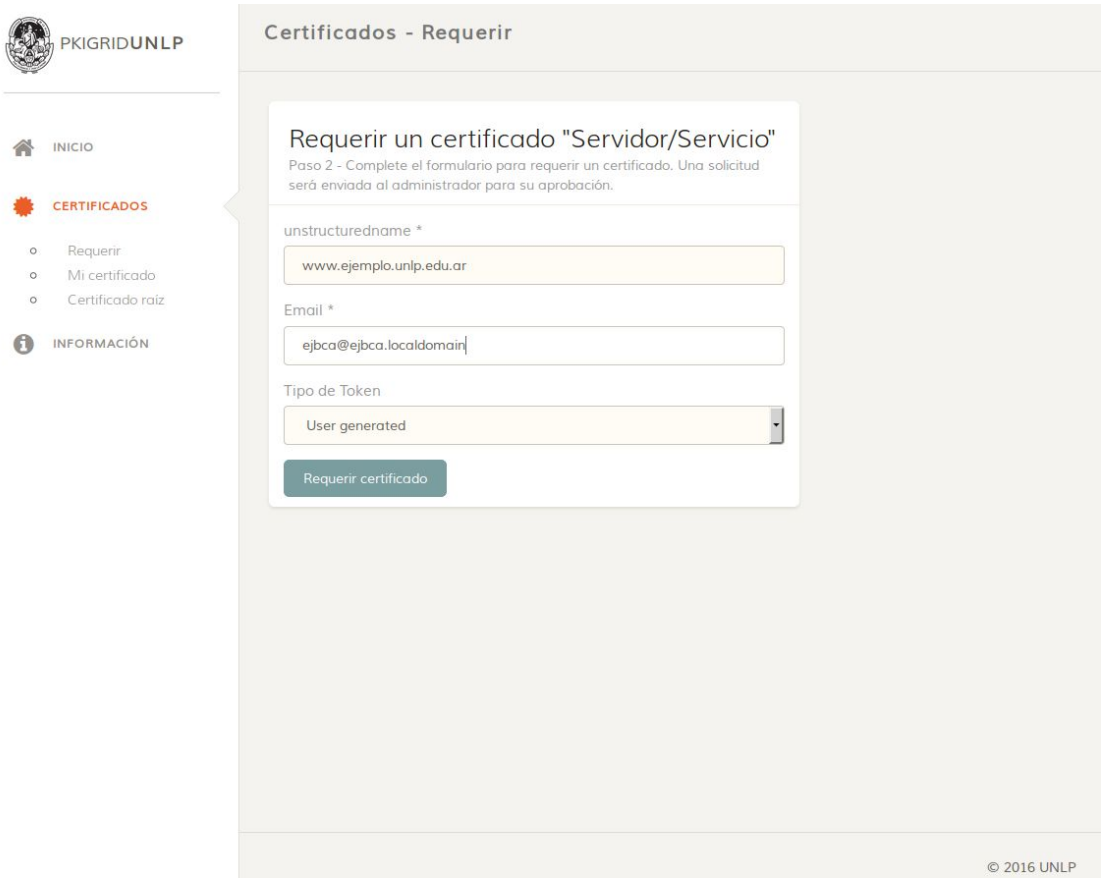


Figura 101. Formulario para requerir un certificado "Servidor/Servicio".

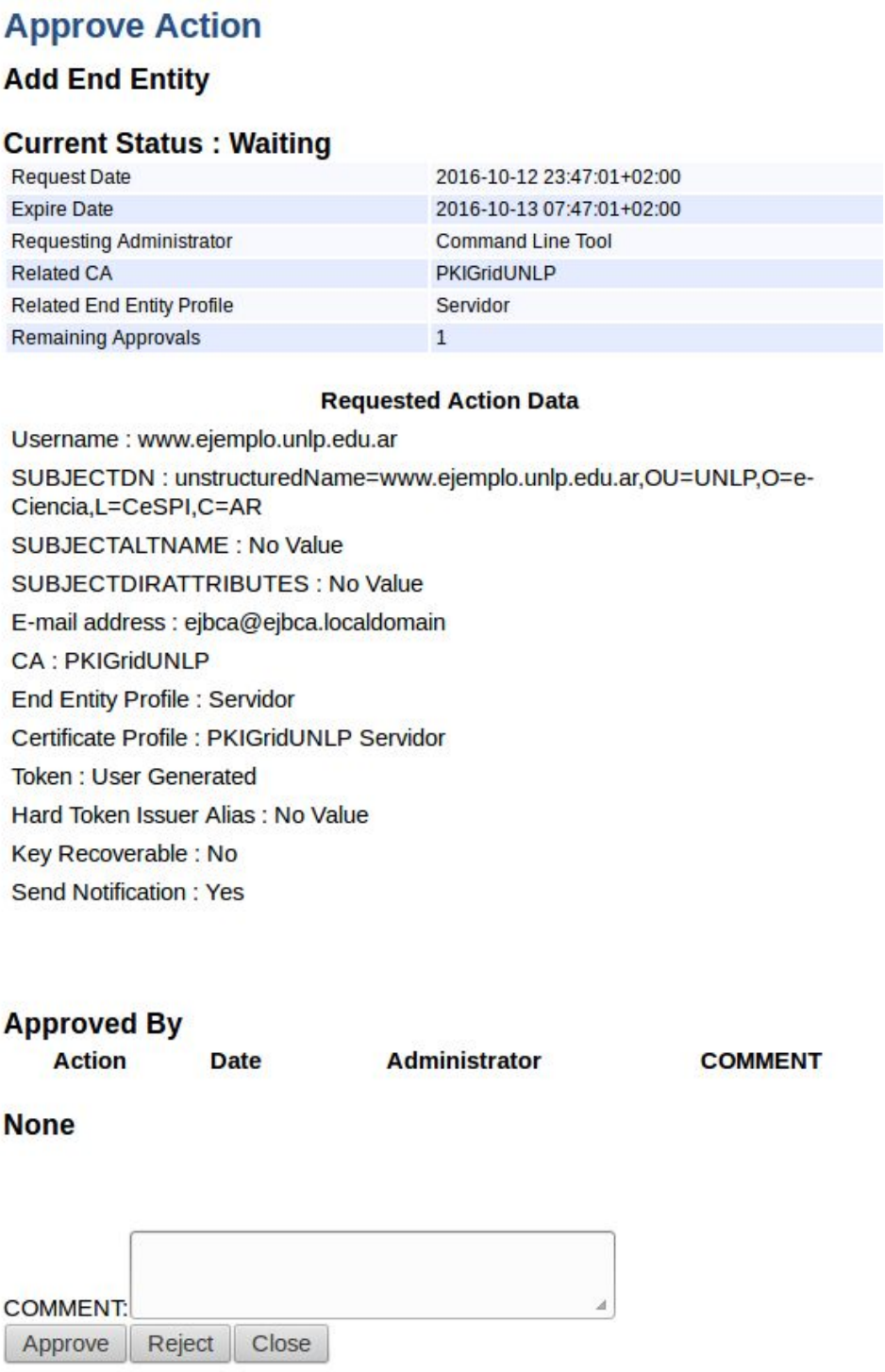


Figura 102. Solicitud de creación de certificado

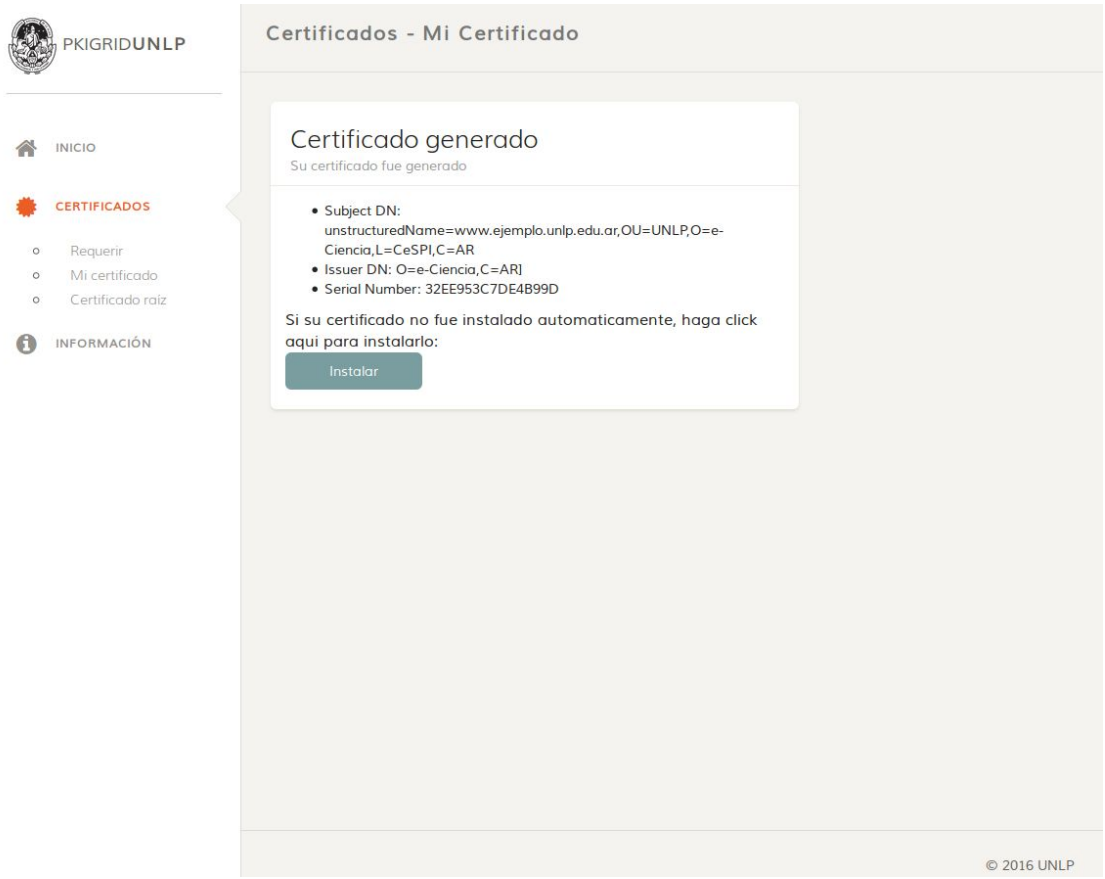


Figura 103. Confirmación de certificado generado.

4.3- Uso de token de autenticación

Un token de seguridad (también conocido como token criptográfico o de autenticación), es un dispositivo que se le otorga a un usuario que esté autorizado para facilitar el proceso de autorización.

Los tokens son de tamaño pequeño, permiten ser trasladados cómodamente. Son utilizados para guardar claves criptográficas, como datos biométricos o firmas digitales. Dependiendo del diseño se pueden hacer a prueba de modificaciones.
Actualmente existen muchos tipos de token diferentes, Están los OTP(One time password) y los que se denominan tokens USB entre otros, los ultimos permiten almacenar contraseñas y certificados.

Para esta sección nos enfocaremos específicamente en el eToken SafePro64 que es la tecnología actualmente utilizada por los operadores de PKIGrid UNLP

Figura 104. eToken SafePro64.

4.3.1- Ejemplo de utilización de eToken para autenticación en EJBCA

Lo primero que necesitamos para utilizar el eToken es el software denominado SafeNet Authentication Client. Una vez instalado utilizaremos el programa para guardar en el eToken el certificado que nos servirá para autenticarnos en distintas ocasiones.

Pasos para la instalación del certificado en el eToken y empleo de este en Firefox:

Insertamos el eToken en la computadora, abrimos el software instalado previamente e ingresamos la clave del eToken si tiene una asignada.

Luego, hacemos click en la opción importar certificado, para el ejemplo utilizaremos el certificado de *"superadmin"*, proporcionado por EJBCA por default.

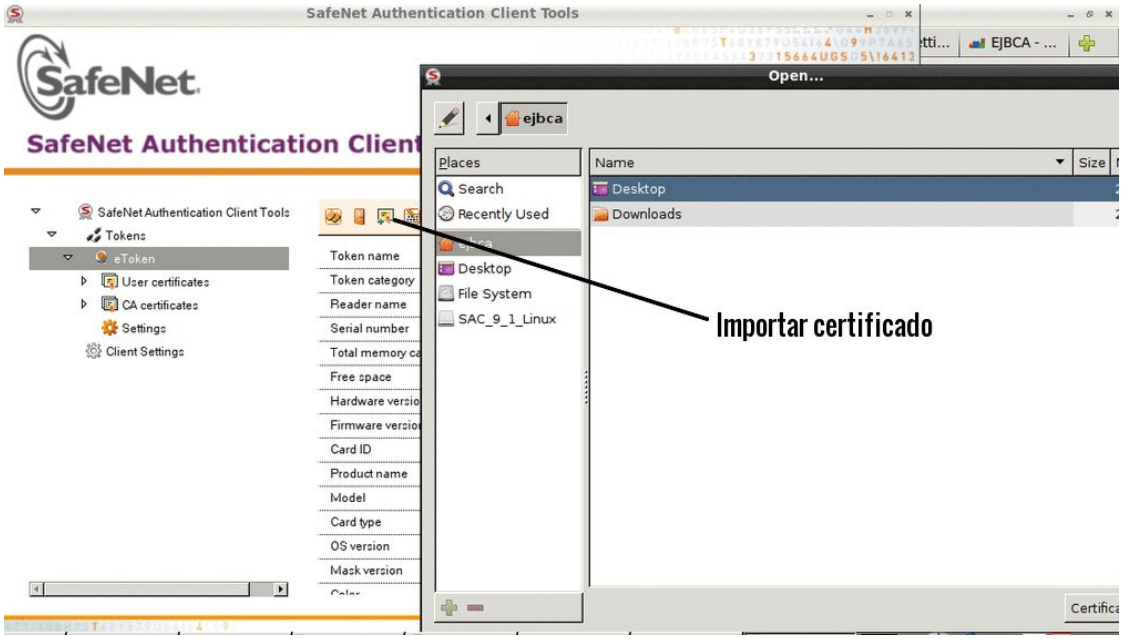


Figura 105. Importación de certificado.

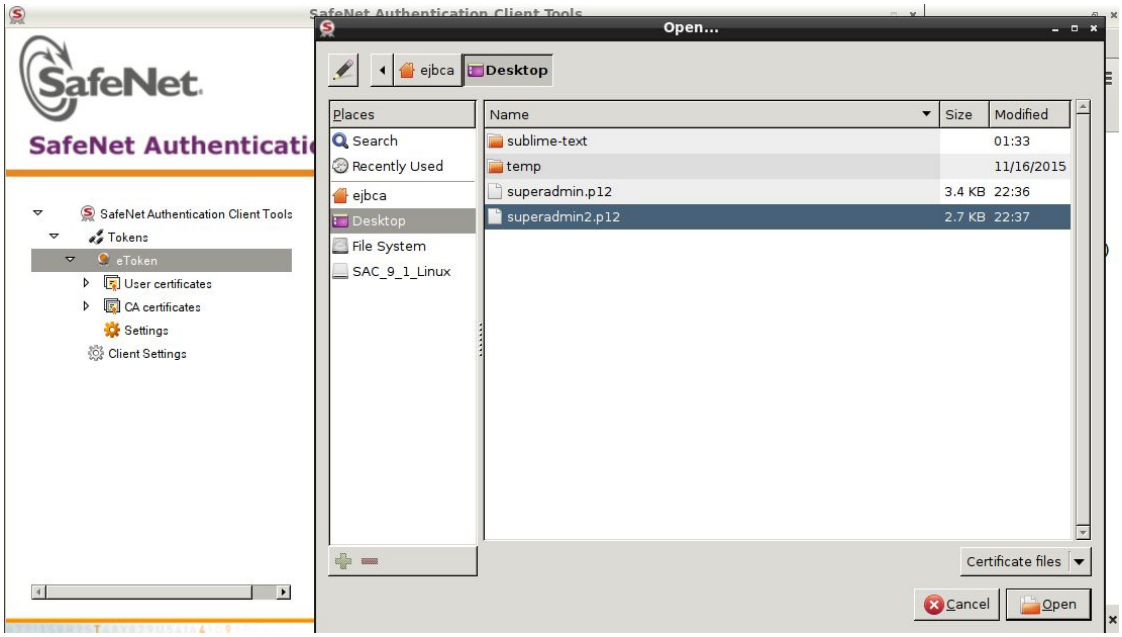


Figura 106. Selección del certificado a importar.

Al seleccionar el certificado se nos pedirá que ingresemos el password para poder insertarlo en el eToken, este password no es el mismo que el del Token, es el del certificado.

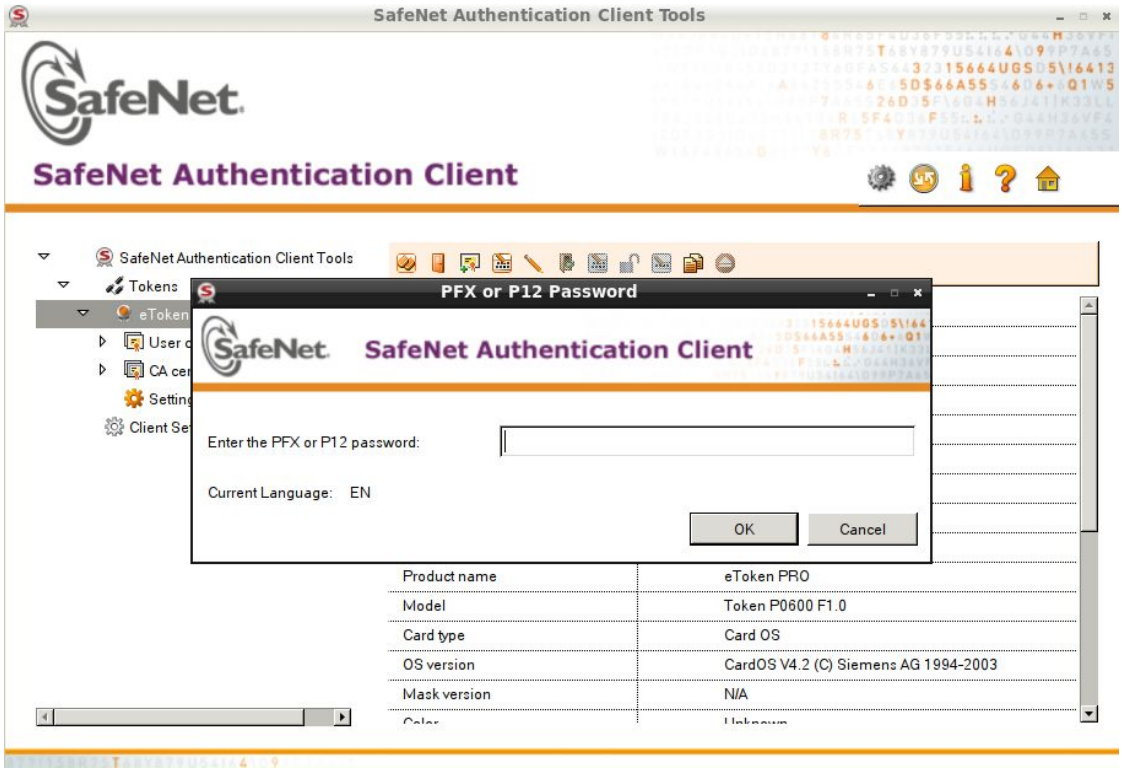


Figura 107. Contraseña requerida para importación de certificado.

Un problema a tener en cuenta es que nos puede pasar, que el tamaño de la clave del certificado no sea soportado por el eToken, esto se soluciona cambiando el tamaño de la clave y generando un nuevo certificado.

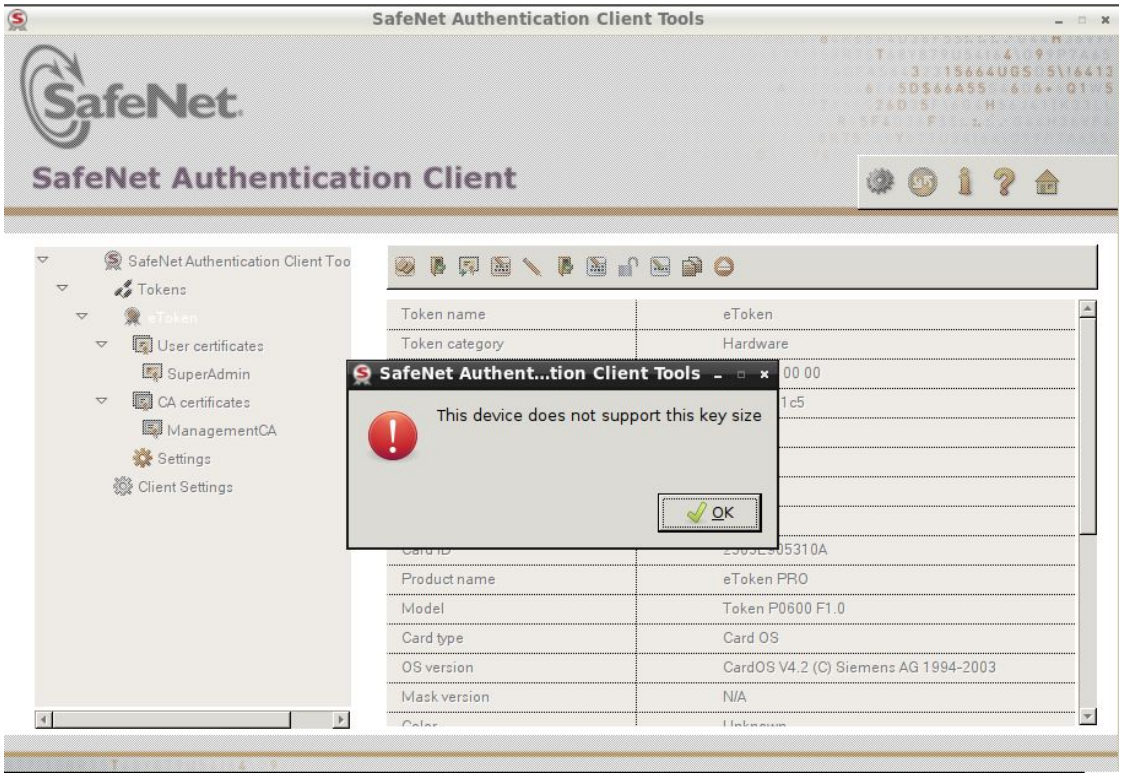


Figura 108. Posible error de tamaño de clave

Una vez configurado el eToken para que guarde nuestro certificado, continuaremos con la configuración de Firefox para que utilice el certificado almacenado en el eToken.

Para esto iremos a la sección preferencias del navegador, una vez ahí, nos desplazamos hasta la pestaña "*Advanced"*, Seleccionaremos la sección de certificates y hacemos click en el botón *"Security devices"*.

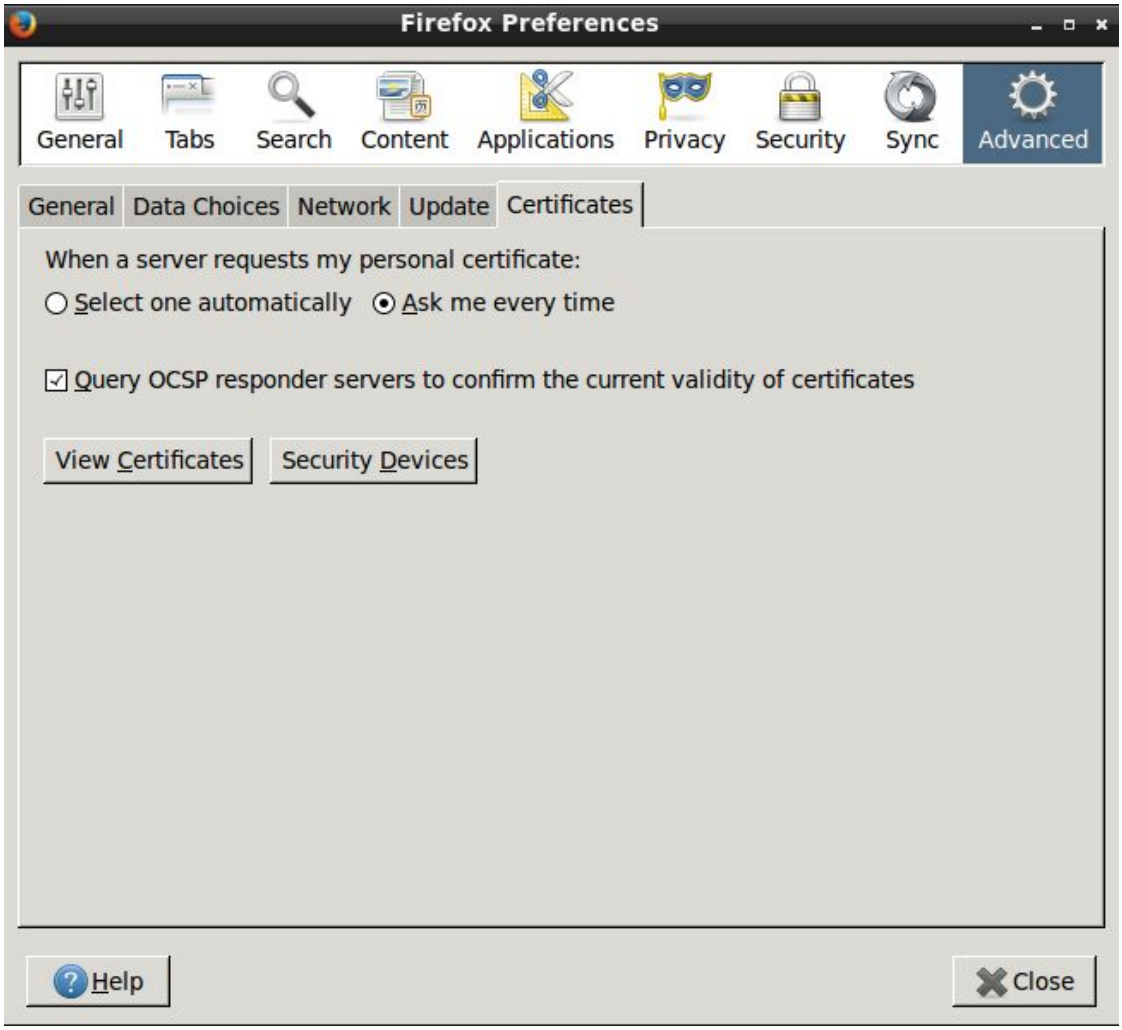


Figura 109. Configuración avanzada de Firefox.

Se nos desplegará el *"Device Manager"*, en el cual cargaremos nuestro módulo para la utilización de certificados del eToken, hacemos click en load

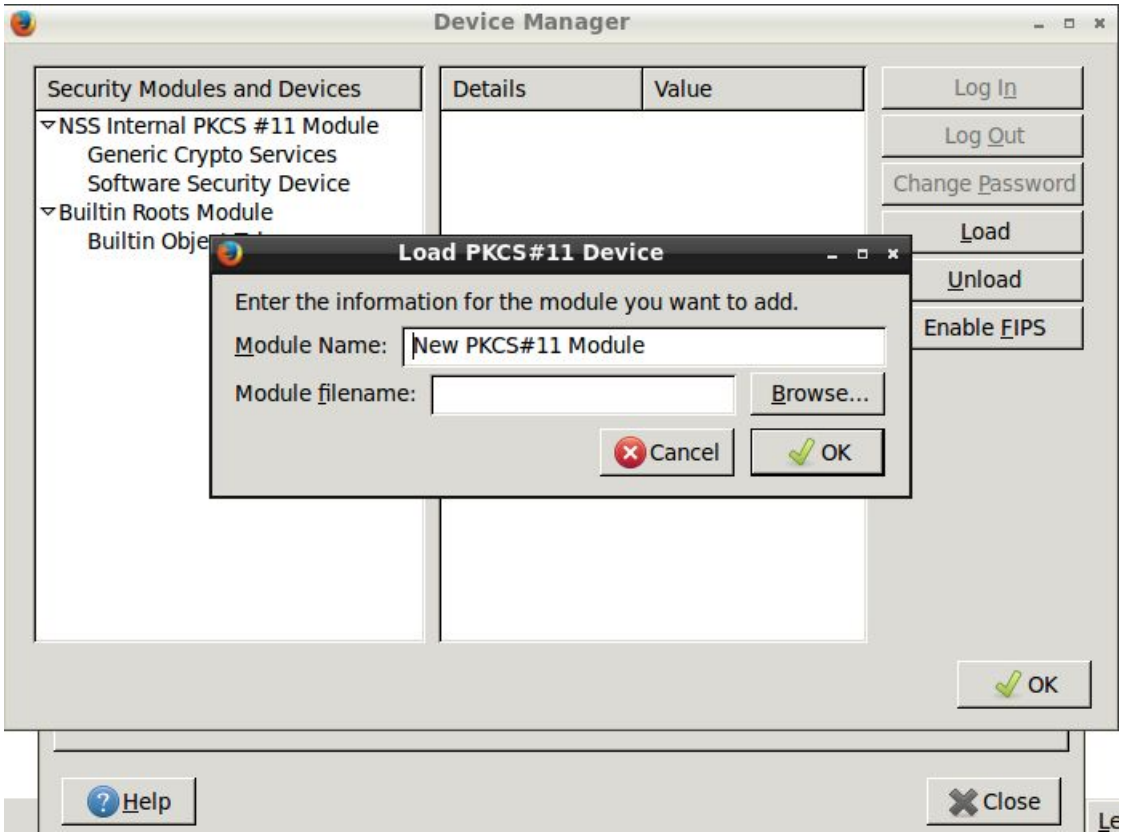


Figura 110. Importación del módulo para poder utilizar eToken en Firefox.

Elegimos la librería que corresponda según la versión de software instalado previamente.

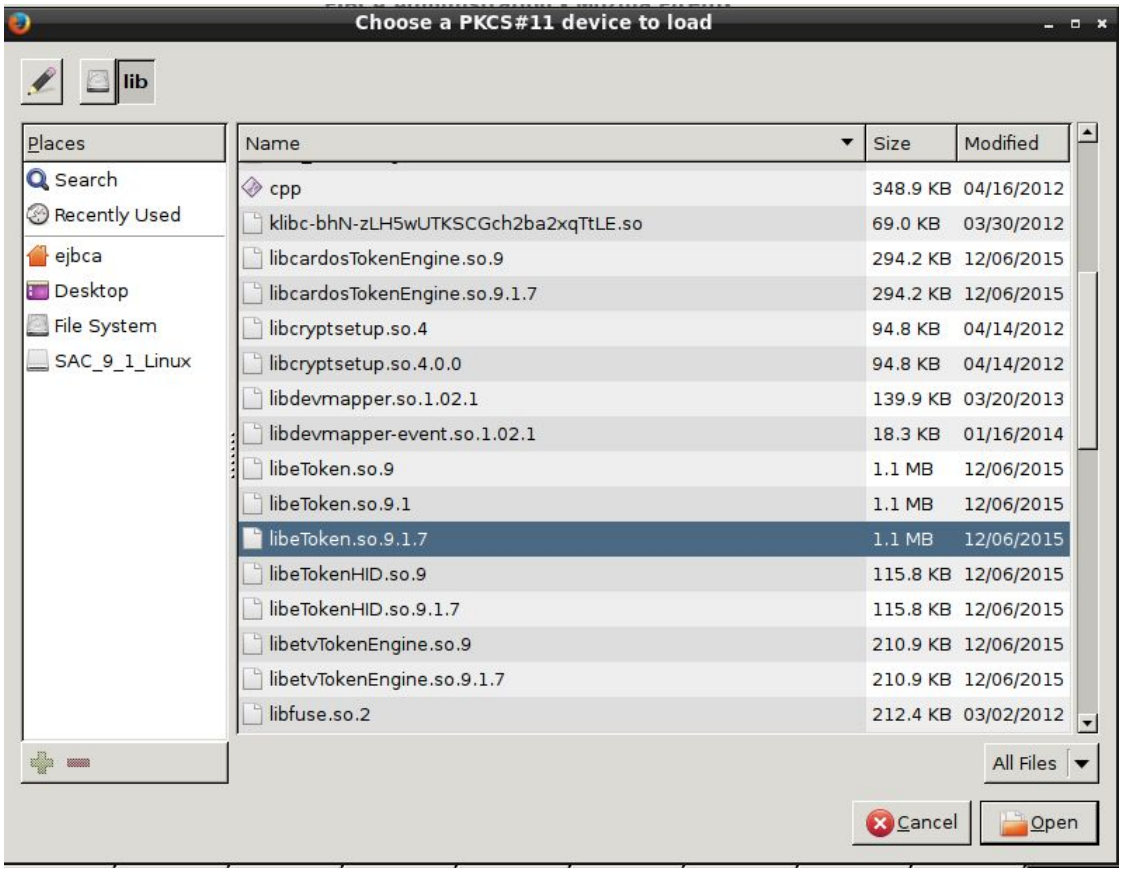


Figura 111. Selección de la librería para utilizar el eToken en Firefox.

Nombramos nuestro módulo con un nombre distintivo para poder identificarlo después. y hacemos click en el botón *"OK"*.

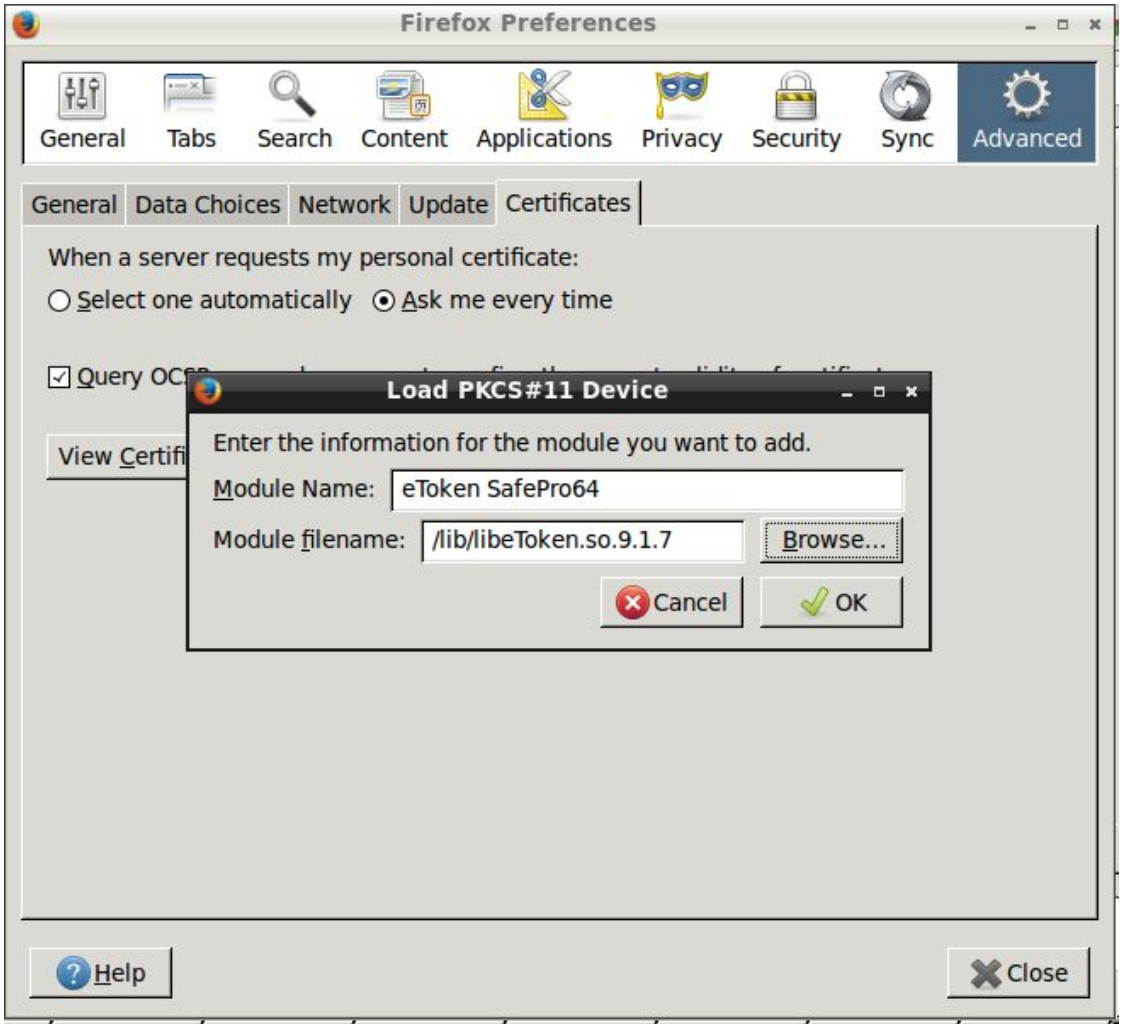


Figura 112. Paso final para la carga del módulo.

Una vez cargado nuestro módulo, tendremos disponibles los eToken conectados a la computadora, seleccionamos el token con el que queremos interactuar y hacemos click en el botón *"Log In"*.

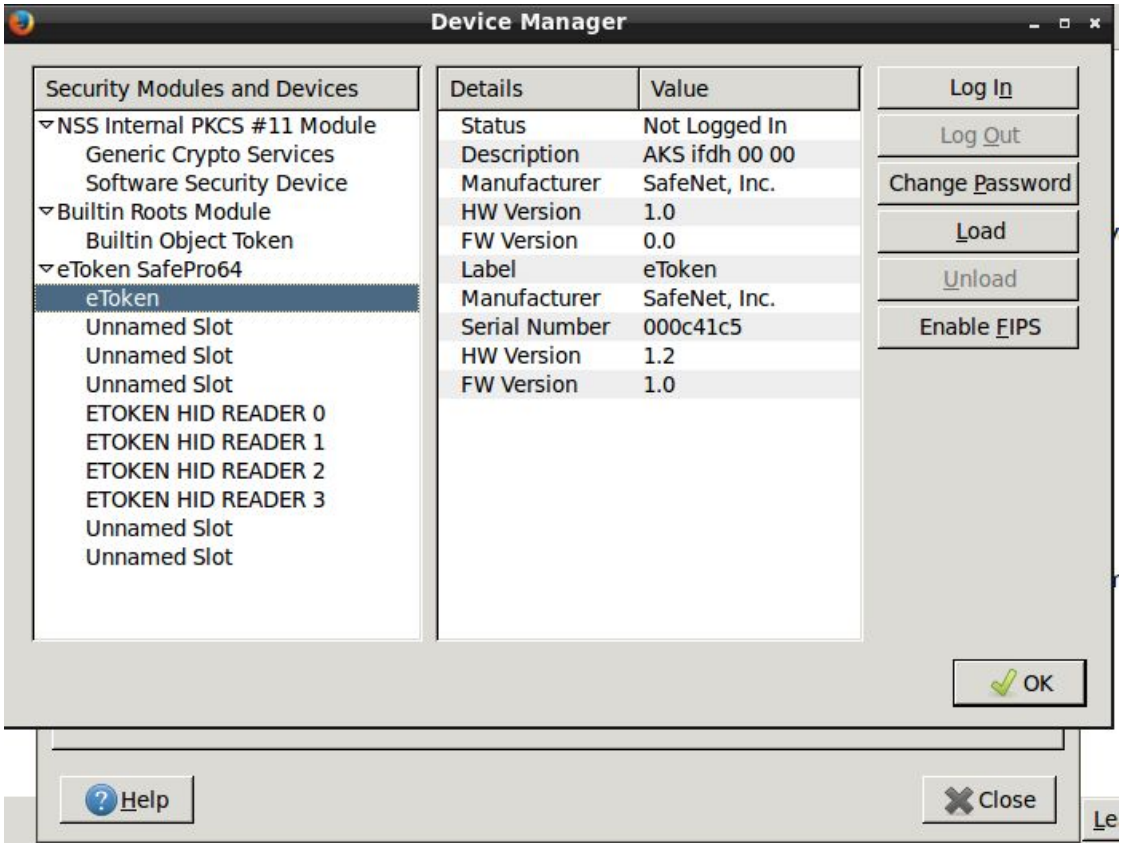


Figura 113. Detalles del módulo.

Nos solicitará el password del eToken, lo ingresamos, y ya tendremos el certificado listo para ser utilizado por nuestro navegador.

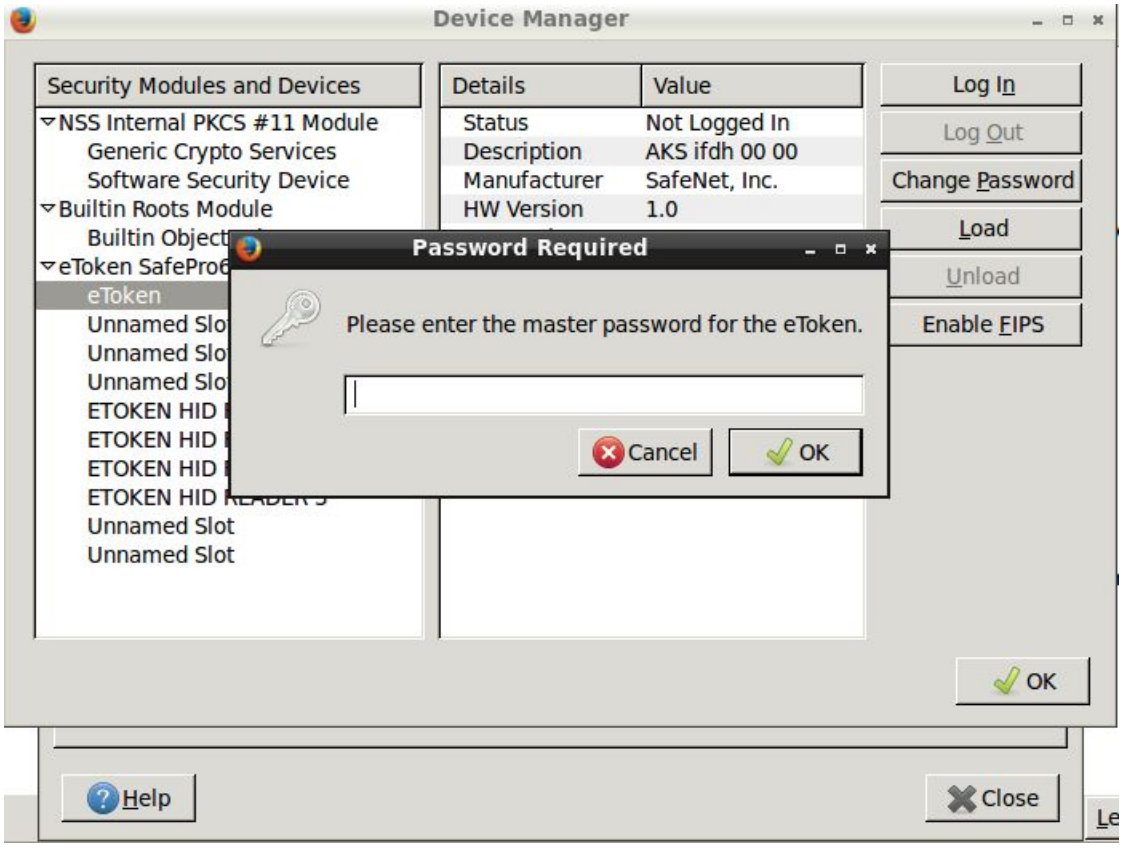


Figura 114. Ingreso del password del eToken.

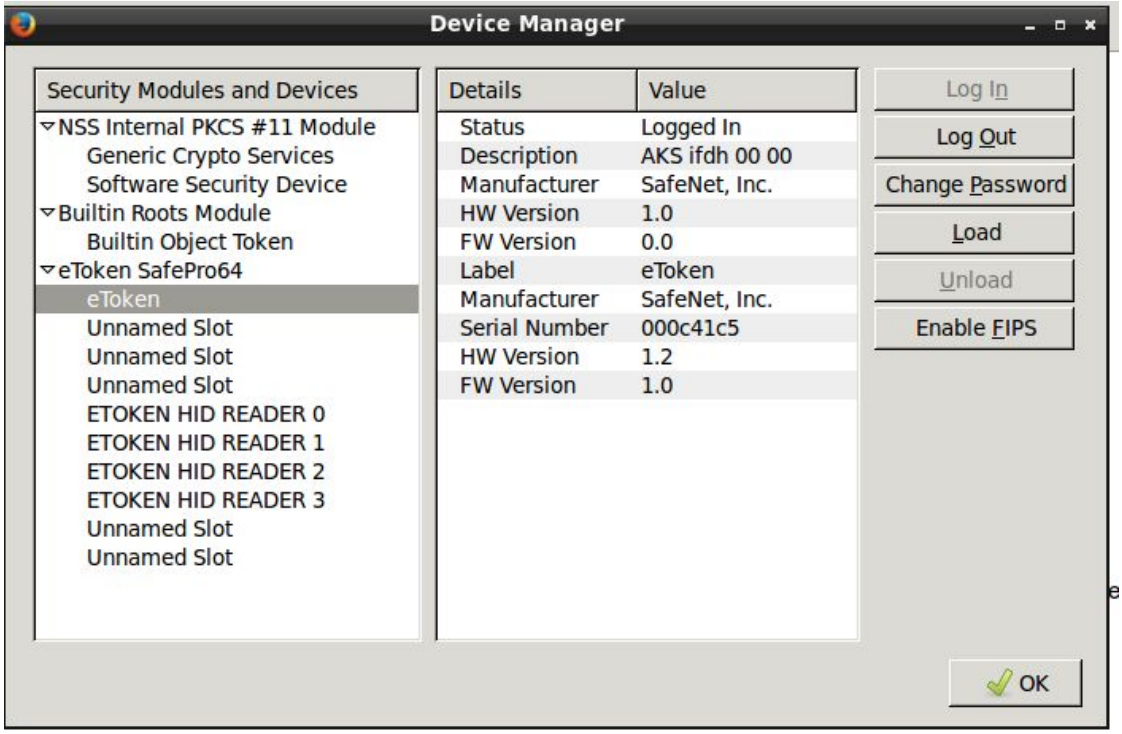


Figura 115. Detalles del token logueado.

Para probar si funciona nuestra configuración podemos ingresar a la sección admin de EJBCA, la cual generalmente nos solicita un certificado para validar nuestro acceso, veremos que accederemos instantáneamente en caso de haber configurado todo correctamente.

4.4- Configuración de OCSP

OCSP es usado por clientes PKi para verificar la validez de los certificados en tiempo real. Esto se realiza mediante el envío de solicitudes de estado de certificado a un respondedor OCSP.

El respondedor puede o no puede ser el mismo que la CA. El respondedor OCSP envía una respuesta firmada, conteniendo la información solicitada sobre el estado al cliente.El cliente usa esta información de estado para determinar si el certificado es válido o está revocado.

El OCSP servlet recibe la solicitud OCSP por http(s) y envia una respuesta de estado firmada por la CA.

El servicio OCSP recibe solicitudes en la dirección de host *"http://localhost:8080/ejbca/publicweb/status/ocsp"*.

El servlet puede procesar solicitudes de certificados firmados por una CA ejecutándose en EJBCA, mientras el servicio Cas OCSP este activo.

Para que una CA sea considerada válida como un respondedor OCSP tiene que tener la configuración *"KeyUsage - Digital Signature"* en el perfil de certificado utilizado para crear dicha CA. Este *"KeyUsage"* debe ser incluido si la CA es utilizada para firmar respondedores OCSP. El perfil default utilizado por las CA ya incluye la propiedad configurada.

Ejemplo para generar una solicitud OCSP utilizando OpenSSL(Funciona con respondedores OCSP internos y externos).

```
openssl ocsp -issuer Test-CA.pem -CAfile Test-CA.pem -cert Test.pem -req_text -url
http://localhost:8080/ejbca/publicweb/status/ocsp
```

Para emitir solicitudes GET para realizar pruebas la siguiente metodología puede ser utilizada

```
openssl ocsp -noverify -no_nonce -respout ocsp.resp -reqout ocsp.req -issuer
ManagementCA.cacert.pem -cert ejbca-test2.primekey.se -url
"http://ejbca-test2.primekey.se:8080/ejbca/publicweb/status/ocsp" -header "HOST"
"ejbca-test2.primekey.se" -text
openssl enc -in ocsp.req -out ocsp.req.b64 -a
curl --verbose --url
http://ejbca-test2.primekey.se:8080/ejbca/publicweb/status/ocsp/MEkwRzBFMEMwQTAJBgUr
DgMCGgUABBT1x9iHOmegR7sYp5Fzdo/u3FMBRgQUH1DTnzl9lscAe3WBTUPCWR+xwpQCCDlpiyU2q5Rb
```

4.5- Migración de certificados y CRL existentes

EJBCA soporta la importación de certificados en una base de datos existente. Esto puede ser utilizado para migrar datos de otros sistemas.
Los certificados pueden ser importados individualmente o en bloque.

Importar un certificado individualmente:

Para importar un certificado individualmente se utiliza el siguiente comando

```
bin/ejbca.sh ca importcert <username> <password> <caname> <status> --email <email>
<certificate file> [--eeprofile <endentityprofile>] [--certprofile
<certificateprofile>] [--revocation-reason <reason>] [--revocation-time <time>]
```

Los parámetros son los siguientes:

username - Username o nombre de usuario de la entidad que debería ser el dueño del certificado, si la entidad no existe, será creada automáticamente. Si existe, el certificado importado será asociado con esta, y las propiedades de dicha entidad serán actualizadas.

password - Contraseña que será configurada para la entidad(generalmente conocida como “enrollment code” en la web pública).

caname - El nombre de la CA que ha emitido el certificado. La firma del emisor del certificado será verificada contra la CA especificada. Si la firma no concuerda, el certificado no será importado. El certificado de la CA debe estar presente en la base de datos(aunque sea como una CA externa).. The CA certificate must be present in the database (at least as an external CA).

status - El estado que debería ser configurado para el certificado importado. Los valores soportados son ACTIVE (para certificados válidos) y REVOKED(para certificados revocados). La razón de revocación para el estado REVOKED será *"Unspecified"*.

email - E-mail que va a ser utilizado para la entidad. Si el E-mail se configura en null, será tomado del certificado en sí.f.

certificate file - Path hacia el archivo PEM BASE64-encoded que contiene el certificado de la entidad.

endentityprofile - Perfil de entidad utilizado para el certificado. El certificado será verificado contra las restricciones que estén especificada en el perfil. Si el certificado no cumple estas restricciones no continuará el proceso de importación. Si no se especifica un perfil de entidad se utilizará el perfil por default que corresponde al *"EMPTY"*.

certificateprofile - Perfil de certificado utilizado para el certificado en sí. Una vez que el certificado es importado, será marcado como perteneciente al perfil de certificado especificado. Si no se especifica ningún perfil de certificado será utilizado el perfil *"ENDUSER"* por defecto.

revocation reason - Permite a un certificado revocado ser importado con una razón de revocación específica. Será *"UNSPECIFIED"*, si no se especifica ninguna.

revocation time - Permite que un certificado revocado sea importado con una fecha de revocación específica con el siguiente formato yyyy.MM.dd-HH:mm. Si no es especificado se utilizará el tiempo actual.

Ejecutar el comando sin ningún parámetro nos muestra una ayuda de como utilizarlo y una lista de parámetros aceptables.

Ejemplo de uso:

```
bin/ejbca.sh ca importcert myuser mypassword SomeCAName ACTIVE --email
mymail@example.com mycertificate.pem --eeprofile EMPTY --certprofile ENDUSER
```

Este comando creará una nueva entidad(o utilizara una existente en caso de estar creada) en EJBCA llamada myser con el password especificado y agrega un nuevo certificado activo para esta entidad de usuario bajo el perfil EMPTY,usando el perfil de certificado ENDUSER.

Importar certificados por bloque:

Para importar certificados por bloque se utiliza el siguiente comando

```
bin/ejbca.sh ca importcertdir <username-source> <caname> <status> <certificate dir>
--eeprofile <endentityprofile> --certprofile <certificateprofile> [-resumeonerror]
[--revocation-reason <reason>] [--revocation-time <time>]
```

Y los parámetros son los siguientes:

username-source - Especifica cómo derivar el nombre de usuario de la entidad que va a ser dueña del certificado. Los valores soportados son CN,DN, y FILE. Si es configurado para DN, el nombre de usuario va a ser igual al atributo “common name” presente en el sujeto del certificado. Si es configurado para DN, el nombre de usuario va ser igual al “distinguished name”, del certificado. Si es configurado para FILE, el nombre de usuario va a ser el mismo que el del archivo(incluida la extensión. Si es configurado para ser CN y no hay un CN presente en el sujeto, el import va a intentar utilizar el DN, si tampoco está disponible utilizara el nombre del archivo. Si es configurado para DN y no se encuentra, se utilizara el nombre de archivo. La entidad se creará automáticamente si no existe. Si existe el certificado importado será asociado con dicha entidad y las propiedades actualizadas.

Los siguientes parámetros funcionan de la misma manera que en el comando mencionado anteriormente, excepto por la salvedad que se aplican al bloque de los certificados importados

- caname
- status
- endentityprofile
- certificateprofile
- revocation reason
- revocation time

certificate dir - Ruta de directorio que contiene los archivos de certificados. Cada archivo debe contener un solo certificado. El directorio especificado no debe tener directorios ni archivos que no sean certificados. La extensión de los archivos no es tomada en cuenta, todos los archivos en el directorio serán procesados.

-resumeonerror -Este parámetro opcional se puede utilizar para forzar a la importación a continuar incluso si se encuentran errores graves. Por default esta opción no se encuentra habilitada, los errores que contempla entre otros, pueden ser violación de restricciones de perfil de entidad final, problemas leyendo certificados, problemas parseando certificados,etc.

Ejecutar el comando sin ningún parámetro mostrará una ayuda básica, y una lista de valores aceptables de parámetros.

Certificados que ya están presentes en la base de datos serán salteados durante la importación. Certificados que tengan una CA diferente a la CA especificada serán ignorados también. Cualquier otra discrepancia detendrá el proceso de importación, a menos que se utilice la opción mencionada anteriormente para ignorar errores. Los certificados que fueron importados antes de encontrar la discrepancia se mantendrán en la base de datos, por cada certificado ignorado el comando imprimirá su número de serie y el nombre del archivo.

Una vez que el proceso de importación ha terminado, un resumen corto es mostrado en la consola, de los certificados importados, número de certificados redundantes y número de certificados rechazados.

No se mostrará ningún resumen si se encuentran errores anormales durante el proceso(A menos que se utilice el comando -resumeonerror).

Si la opción -resumeonerror se encuentra habilitada entonces el resumen contendrá también el número de certificados que no pudo ser leído, el número de certificados que han violado las restricciones de entidad final y número de certificados que no se pudieron leer por otros errores.

5- Conclusión

La Universidad Nacional de La Plata cuenta con una infraestructura de clave pública (PKIGrid UNLP) para emisión de certificados para Argentina acreditada por TAGPMA la cual soporta las actividades de e-ciencia de la comunidad académica Argentina.

Actualmente dicha PKI se encuentra implementada utilizando una tecnología denominada OpenCA. Durante el desarrollo del trabajo se demostró que no solo esta tecnología es difícil de operar, sino también de mantener.

Se realizó una comparación entre las tecnologías actuales que utiliza PKIGrid UNLP (OpenCA) y la tecnología propuesta (EJBCA), se llegó a la conclusión que esta última no solo es más fácil de administrar, sino que presenta ventajas al momento de adaptar y de configurar la PKI, haciendo de estas tareas un trabajo mucho menos tedioso para el encargado de mantener, y verificar el sistema. EJBCA tiene mucho potencial, es robusto, conciso y la documentación existente respecto a dicha tecnología es en comparación, abundante. Si bien es entendible que la migración puede llevar a traer problemas, tanto de inconsistencia como de tiempo, se puede observar que las mejoras que presenta EJBCA sobre OpenCA superan en amplio margen cualquier tipo de contingencia generada por el proceso de migración. Queda entonces demostrado que, no solo es factible y recomendado el cambio de tecnología, sino que representaría una gran mejora en el sistema utilizado actualmente.

6- Trabajo a futuro

La naturaleza de las aplicaciones open source implica que cualquiera puede realizar mejoras sobre el código fuente de una manera continua. El trabajo a futuro propuesto en esta sección se basa en mejoras para que cualquiera pueda tomarlas y desarrollarlas sobre el código fuente de EJBCA.

- Mejorar la arquitectura de EJBCA, separando las capas de la lógica con las de la vista y el modelo.
- Para el front-end proponemos reemplazar la tecnología de JSP e incluir una tecnología más moderna y robusta como lo es Angular.js. Utilizando esta tecnología separamos estrictamente la vista del resto de la aplicación, además, nos beneficiaremos del two-way databinding, la carga asincrónica de datos, la internacionalización de los textos, y el uso de otras tecnologías como Grunt para separar las librerías del código fuente.
- Para el manejo del back-end proponemos utilizar Spring MVC aprovechando el hecho de que el base-code está escrito en Java. Spring MVC es un framework robusto de nivel empresarial que provee herramientas que facilitan el desarrollo de futuras mejoras.
- Para la compilación proponemos dejar de lado la tecnología ANT y utilizar Maven o Gradle, tecnologías muy utilizadas actualmente a nivel empresarial. Al utilizar estas tecnologías también separamos la inclusión de librerías en el código fuente.

7- Referencias bibliográficas

[1] The Open-source PKI Book: A guide to PKIs and Open–source Implementations.

[2] TAGPMA - http://www.tagpma.org/.

[3] Introduction to Modern Cryptography (2nd edition) - Jonathan Katz and Yehuda Lindell.

[4] Adaptando OpenCA para implementar una PKI para e-Science - Díaz, Ambrosi, Luengo, Macia, Molinari, Venosa.

[5] The Americas Grid Policy Management Authority Charter v2.5.

[6] OpenCA Guide.

[7] Documentación oficial de PKIGrid UNLP.

[8] RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

[9] RFC 2560 - X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP).

[10] RFC 5019 - The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments.

[11] RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1.

[12] RFC 2818 - HTTP Over TLS.

[13] RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax.

[14] RFC 2595 - Using TLS with IMAP, POP3 and ACAP.

[15] RFC 4945 - The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX.

[16] RFC 5055 - Server-Based Certificate Validation Protocol (SCVP).

[17] RFC 4210 - Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP).
[18] RFC 4211 - Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF).
[19] RFC 4387 - Certificate Store Access via HTTP.
[20] RFC 6960 - X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP.
[21] RFC 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile.
[22] RFC 3647 - Cert Policy and Certification Practices Framework.
[23] RFC 3820 - Internet X.509 Public Key Infrastructure Proxy Certificate Profile.
[24] RFC 3494 - Lightweight Directory Access Protocol version 2 (LDAPv2).
[25] https://technet.microsoft.com
[26] Cryptography for Dummies
[27]http://encuentrofd-2012.congresos.unc.edu.ar/wp-content/blogs.dir/12/files/UNLP-KPIgrid-octubre-12-v2.pdf
[28] http://www.karapanza.net/pki-con-open-source-parte-i/
[29] http://www.pkigrid.unlp.edu.ar/docs/unlpca-cp-cps_SPA-v2.7.pdf
[30] http://www.pkigrid.unlp.edu.ar/docs/BuenasPracticas_SPA.pdf
[31] http://www.pkigrid.unlp.edu.ar/docs/subscriber-obligations_SPA.pdf
[32] http://www.pkigrid.unlp.edu.ar/docs/ca-obligations-structure-operation_SPA.pdf
[33] http://www.pkigrid.unlp.edu.ar/docs/ra-obligations-structure-operation_SPA.pdf
[34] https://www.ejbca.org/docs/installation.html
[35] https://www.ejbca.org/docs/adminguide.html
[36] https://www.ejbca.org/docs/userguide.html
[37] https://www.ejbca.org/docs/userguide-ocsp.html
[38] https://www.ejbca.org/docs/installation-ocsp.html