Trabajo de Grado

Implementando Firma Digital con J2EE

Alumnos: María Patricia Clemens – Alejandro César Falcone
Director : Prof. Lic. Javier F. Díaz

Licenciatura en Informática
Facultad de Informática
Universidad Nacional de La Plata
Agosto 2005

AGRADECIMIENTOS

Queremos agradecer:

- *A Javier, quién a pesar de sus múltiples ocupaciones, aceptó guiarnos en este trabajo.*

- *A Tomas Gustavsson (Tom), que nos ayudó a solucionar los inconvenientes de implementación de EJBCA y nos brindó prontamente toda la información que le requerimos.*

- *A PrimeKey Solutions, por el material disponible sobre EJBCA.*

- *Personalmente quiero agradecer muy especialmente a María Elena y a Nicolás, por todo el tiempo que me dispensaron para invertirlo en este trabajo y la paciencia infinita que ambos tuvieron por mis largas horas de ausencia. Este trabajo no hubiera sido posible sin ellos.*

Alejandro

Indice

Sección III – EJBCA – Enterprise Java Beans Certificate Autority

Esta página fue dejada en blanco de manera intencional.

Motivación

En la actualidad, y fundamentalmente a partir del crecimiento que ha experimentado Internet, la problemática de la seguridad tanto en las redes públicas como en las privadas (intranets) es un tema aún no resuelto en su totalidad. Una de las opciones tecnológicas disponibles para lograrlo es la Firma Digital, que está siendo utilizada a nivel mundial por muchas organizaciones y está comenzando a tener auge en nuestro país a partir de normas que reglamentan su funcionamiento.

Este trabajo está enfocado a proponer una arquitectura para montar Firma Digital con J2EE, basándonos en la utilización de herramientas y entornos Open Source. El entorno J2EE nos brinda la posibilidad de independizarnos de la plataforma (una aplicación que utilice las APIs estándar puede ser instalada en cualquier servidor de aplicaciones conforme a J2EE); por otro lado, existen en la actualidad una gran cantidad de aplicaciones certificadas J2EE, muchas de ellas gratuitas y Open Source.

La idea de proponer implementaciones basadas en Open Source (Código Abierto) surge en el hecho de que cada día son más las herramientas de este tipo disponibles y existe una marcada tendencia de usuarios en todos los niveles – empresas, gobiernos, educativos, etc. – a migrar gran parte de sus aplicaciones hacia él. Mucho se ha debatido, y aún sigue la polémica, a cerca de la conveniencia de este tipo de aplicaciones e implementaciones. Lo cierto es que gran cantidad de empresas especializadas (Gartner Group, Netcraft Ltd., Zdnet, SPEC Consortium, Bloor Research, por citar solo algunas) han efectuado mediciones y comparaciones entre sistemas "abiertos" y "cerrados", cuyos resultados han sido publicados en diversos informes. Tales comparaciones fueron efectuadas sobre distintos aspectos - cuota de mercado, estabilidad, rendimiento, escalabilidad, seguridad, TCO (Total Cost of Ownership) - y en su gran mayoría favorecían ampliamente a las implementaciones Open Source. Existen por otra parte muy conocidos sitios que utilizan código abierto en parte de sus aplicaciones, como ser la Casa Blanca, Google, Rackspace, Yahoo, Sony Japón.

Es nuestra intención que la arquitectura propuesta pueda servir como base para posteriores implementaciones utilizando las especificaciones J2EE, tanto dentro del ámbito académico de la Facultad como en uno externo a ella. Nos resulta de sumo interés por tanto, aportar los análisis e investigaciones que hemos efectuado para este trabajo, y que estas sean de utilidad para quienes incursionen en la implementación de una infraestructura PKI.

Esta página fue dejada en blanco de manera intencional.

Estructura del Trabajo

El propósito de este trabajo es estudiar y proponer una arquitectura que permita montar Firma Digital con J2EE, mediante la utilización de herramientas y entornos Open Source. Para ello hemos relevado y estudiado productos que nos permitan lograr esta arquitectura, para luego volcar la experiencia adquirida en el presente informe.

Seguidamente, brindamos un breve resumen de los tópicos que cubren cada capítulo de este trabajo. Para una mejor organización, lo hemos dividido en Cuatro Secciones:

Motivación

Presenta el contexto general y el objetivo de la tesis propuesta.

Sección I – CONCEPTOS GENERALES

Capítulo 1 – Introducción y Conceptos Generales

En este capítulo se aborda la problemática de la seguridad, motivo principal del trabajo en curso. Se explican conceptos generales sobre Criptografía, tipos de encriptación y Hashing, conocimientos básicos requeridos para comprender el informe.

Capítulo 2 – Infraestructura de Clave Pública

Explica las características de la infraestructura PKI, sus componentes, estándares requeridos para implementarla.

Capítulo 3 – Certificados Digitales

Se describen aquí las características de los certificados digitales, formatos y tipos de certificados, revocación de certificados y Listas de Revocación.

Capítulo 4 – Firma Digital

Este capítulo explica los conceptos relacionados con la firma digital y el proceso de generación de la misma.

Sección II –J2EE – Java 2 Platform Enterprise Edition

Capítulo 5 – Introducción y Modelo

Se introduce aquí al concepto de J2EE, sus características y modelo. Se explican las componentes que define la especificación J2EE (cliente, web, empresariales).

Capítulo 6 – Arquitectura J2EE

Este capítulo explica la arquitectura J2EE, contenedores y servicios, tipos de contenedores y APIs de J2EE.

Sección III – EJBCA – Enteprise Java Beans Certificate Authority

Capítulo 7 – Conociendo EJBCA

En este capítulo se realiza una introducción a EJBCA, presentando sus características generales y arquitectura. Se presentan una serie de conceptos y terminologías aplicables a entornos PKI en general y específicos de EJBCA.

Capítulo 8 – Construcción y Configuración de EJBCA

Este capítulo aborda los pasos requeridos para la construcción de EJBCA y consideraciones generales previas a la configuración general de la aplicación. Brinda información sobre la conexión a la base de datos y describe las tablas del sistema. Explica los pasos requeridos para efectuar la instalación inicial, construcción de la root CA e importación del primer certificado de super-administrador generado durante la instalación.

Capítulo 9 – Interfase WEB

Se describen aquí las características y opciones de la interfase de web pública que ofrece EJBCA.

Capítulo 10 – Administración de EJBCA mediante Interfase Web

Se explica el concepto de Roles de EJBCA. Se realiza un análisis detallado de la Interfase de Administración Web, explicando todas sus opciones y posibilidades de configuración. Se describen las Reglas de Acceso en EJBCA.

Capítulo 11 – Administración de EJBCA por Línea de Comandos

Se presentan los comandos existentes para la administración mediante línea de comandos. Se explica la funcionalidad cada uno de los subcomandos existentes.

Capítulo 12 – Dependencias

En este capítulo se describen cada una de las herramientas y aplicaciones requeridas para construir, instalar, configurar y depurar EJBCA. Se describen las características generales de estas herramientas y sitios de descargas de ellas.

Capítulo 13 – APIs en EJBCA

Este capítulo contiene la enumeración y breve descripción de las APIs provistas por EJBCA.

Sección IV – Construcción de una PKI mediante EJBCA

Capítulo 14 – Construyendo una PKI paso a paso con EJBCA

Se describe paso a paso la construcción de una PKI medianamente compleja, creando CAs, Ras, Administradores y Perfiles necesarios para su funcionamiento.

Conclusiones

Presentamos las conclusiones del presente informe.

Terminología y Abreviaciones

Define los términos y abreviaciones utilizados con frecuencia en el trabajo.

Referencias

Da las referencias presentes en el transcurso del trabajo.

Bibliografía

Lista la bibliografía empleada para la realización del trabajo.

Capítulo 1
Introducción y Conceptos Generales

En la actualidad, el intercambio electrónico de información es utilizado diariamente por miles de empresas, organizaciones y personas, ya sea a través de redes públicas (Internet) o privadas (intranets de empresas). A partir del vertiginoso crecimiento de Internet esta modalidad se ha acentuado aún más, constituyendo hoy una forma natural de efectuar innumerables tipos de transacciones: compartir información confidencial mediante el correo electrónico o la red, acceder y operar cuentas bancarias desde el hogar o la oficina, pagar impuestos, efectuar compras on-line, etc.

El manejo de toda información, y mas aún cuando se trata de información sensible como la descripta en el párrafo anterior, requiere mecanismos que permitan protegerla y controlar el acceso a la misma, evitando así que se puedan interceptar y falsificar mensajes, robar o alterar información, y espiar usuarios – sean estos empresas o individuos-. Esta necesidad obliga a dotar de seguridad a los datos, servicios, transacciones y las partes involucradas en ellas.

Por otra parte, en las transacciones basadas en documentación escrita, existen mecanismos que garantizan:

- *Confidencialidad* – asegura que el contenido del mensaje es privado, es decir que solo puede ser visto por el destinatario.
- *Integridad* – garantiza que el contenido del mensaje no ha sido alterado.
- *Autenticidad* – garantiza que el mensaje proviene de la persona que efectivamente lo está enviando.
- *No repudio* – impide que el emisor niegue haber enviado un mensaje o el receptor niegue haberlo recibido.

Dado que los servicios de Internet carecen de la implementación de estos conceptos de seguridad, los cuales son fundamentales para la validez legal de las transacciones de comercio electrónico, es necesario incorporar mecanismos que los garanticen.

Para este fin, las técnicas criptográficas - tales como encriptación y firma digital – constituyen piezas fundamentales en la implementación de los mismos. Todas las tecnologías propuestas hasta el momento, incluyendo tokens, protocolos seguros para transmisión de datos, certificados digitales y estándares para confiar en Sitios Web involucran alguna forma de criptografía.

Las firmas digitales, que son creadas y verificadas utilizando criptografía, pueden ser usadas para firmar documentos digitales, asegurar su integridad, certificar código de aplicación y componentes, setear topologías de redes públicas seguras y encriptar datos.

CRIPTOGRAFIA

El concepto de asegurar los mensajes mediante la criptografía tienen una larga historia. Uno de los primeros sistemas cifrados fue creado por Julio César, quien lo utilizaba para enviar mensajes militares secretos a sus generales.

El objetivo central de la criptografía es la protección de información (la cual puede contener texto plano, imágenes, sonido u otros objetos binarios), de manera tal que ésta tenga significado solo para el destinatario. Es altamente utilizada para proteger claves e información extremadamente sensible a través de redes inseguras (por ejemplo internet). Para esto, la criptografía lleva a cabo el proceso de *cifrado*, mediante el cual transforma texto plano (es decir, el mensaje original en texto legible) en texto cifrado y el proceso inverso, llamado *descifrado*, el cual transforma un mensaje cifrado en el correspondiente texto plano.

Ambos procesos, cifrado y descifrado, son realizados mediante la aplicación de algoritmos matemáticos y la utilización de claves.

Existen dos diferentes tipos de encriptación:

- Simétrica
- Asimétrica

Encriptación Simétrica:

En este esquema, el más clásico de la criptografía, se utiliza la misma clave - llamada *clave secreta* – tanto para encriptar como para desencriptar el mensaje. El mayor problema con este sistema es el transporte de la clave secreta entre el emisor y el receptor de mensaje sin comprometer la seguridad.

Algunos sistemas (por ejemplo el Sistema Kerberos del MIT [Ref.1]) utilizan solamente claves secretas simétricas para comunicarse de manera segura sobre redes públicas, pero esto es difícil de implementar en grandes organizaciones haciendo necesario procedimientos extras de seguridad tales como servidores centrales "seguros y confiables". Algunos ejemplos de implementaciones de encriptación simétrica son el algoritmo DES (Data Encryption Standard) [Ref.2], Triple DES [Ref.3], IDEA (International Data Encryption Algorithm) [Ref.4].

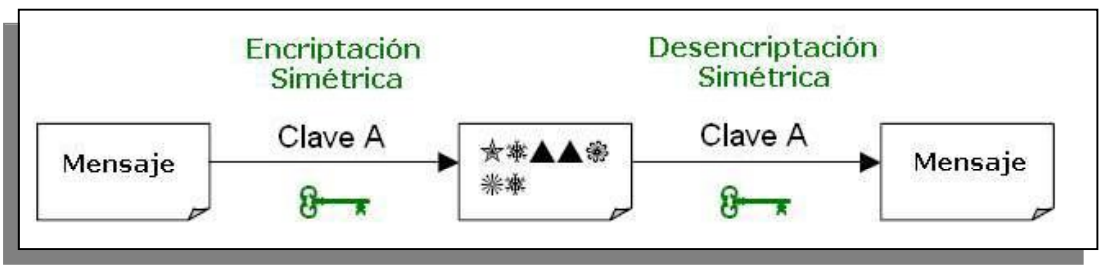


Fig. 1 – Encriptación Simétrica

Encriptación Asimétrica:

Como mencionamos anteriormente, el principal inconveniente que presenta la criptografía de clave simétrica, es la distribución de las claves de manera segura. La criptografía de clave asimétrica utiliza claves diferentes para cifrar y descifrar un mensaje, así lo único que se transmite de entre los usuarios es el mensaje cifrado.

Los criptosistemas asimétricos, o también llamados criptosistemas de *clave pública*, utilizan dos claves:

clave pública, la cual es conocida por todos, para encriptar un mensaje y
clave privada, secreta y conocida solo por su dueño, para desencriptarlo.

Un punto importante a destacar aquí: es virtualmente imposible conociendo una de las dos claves, obtener o deducir la otra.

La principal desventaja de este sistema es que, debido a la mayor complejidad, es menor la velocidad de procesamiento de los datos respecto al sistema simétrico.

Ejemplos de algoritmos asimétricos son RSA (Rivest Shamir Adleman) el cual es permutable - una clave puede cifrar o descifrar el mensaje -, ECDSA (Elliptic Curve Digital Signature Algorithm) el cual es una variante del DSA y permite implementar algoritmos existentes mediante la utilización de curvas elípticas (la mayor ventaja es que las claves son más pequeñas sin comprometer la seguridad y consecuentemente ganar en velocidad de procesamiento).

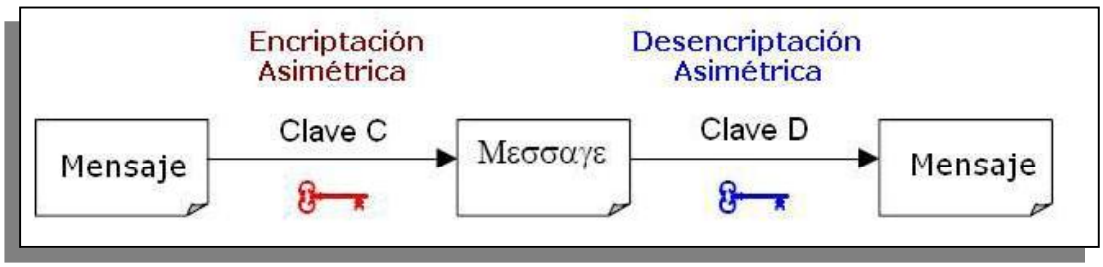


Fig. 2 – Encriptación Asimétrica

HASHING

Hashing es el método utilizado para obtener una *huella digital* (también llamado *digesto*) de un mensaje y consiste en la aplicación de una función matemática al mensaje; esta huella digital, es utilizada posteriormente para validar la integridad de dicho mensaje. El código de hash tiene una longitud fija – normalmente 128 o 160 bits – y tiene la particularidad de ser único, es decir diferentes mensajes producen diferentes digestos.

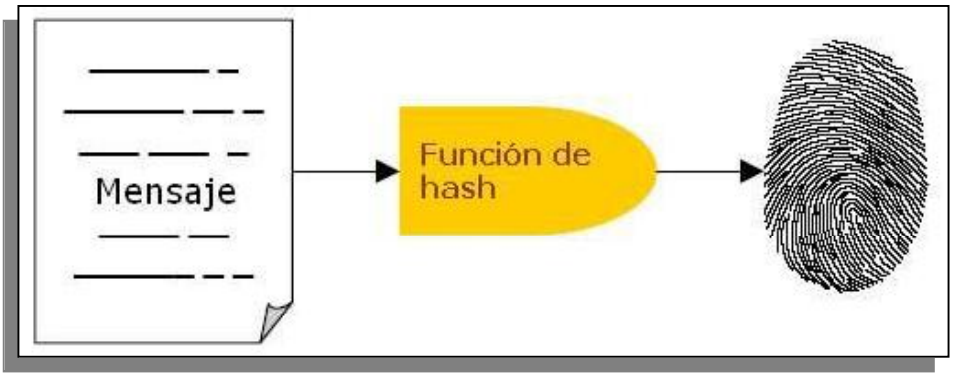


Fig. 3 – Hashing

Los algoritmos de Hashing, son también conocidos como checksum criptográfico, huella digital, MIC (message integrity check). Algunos ejemplos de dichos algoritmos son MD2, MD4, MD5 (el cual utiliza 128 bits, creado por Ron Rivest) y SHA1 (Secure Hash Algorithm, que utiliza 160 bits y fue desarrollado por el Instituto Nacional de Ciencia y Tecnología de Estados Unidos).

Capítulo 2
Infraestructura de Clave Pública

Introducción

La PKI o Infraestructura de Clave Pública, consiste en un conjunto de protocolos, servicios y estándares que brindan soporte a aplicaciones de clave pública.

El propósito de una Infraestructura PKI es proveer claves y manejo de certificados de manera confiable y eficiente, a fin de lograr la habilitación de la autenticación, no repudiación y confidencialidad.

Un punto importante para los sistemas basados en clave pública, es que cada vez que ellos confían en una clave pública, deben estar seguros que la clave privada asociada pertenece al sujeto con el cual se están comunicando. Sin embargo, el par de claves no tiene una asociación intrínseca con una persona. Esta confianza se basa en la utilización de *Certificados de Clave Pública*, los cuales son estructuras de datos que ligan a personas con sus claves públicas. Este vínculo entre las personas y sus claves, se realiza mediante una *Autoridad de Certificación* (*CA*), quien se encarga de verificar la identidad del sujeto y firmar digitalmente cada certificado que emite.

Esta infraestructura permite a los usuarios interactuar con otros usuarios y aplicaciones, obtener y verificar identidades o claves, y realizar registración a través de autoridades de certificación, las cuales actúan como tercer parte confiable y realizan la comprobación de la identidad de los usuarios.

Componentes de una PKI:

Las entidades que básicamente conforman una PKI son las que podemos encontrar en la tabla siguiente:

ENTIDAD	ROL PRINCIPAL
Autoridad de Certificación (CA)	es la encargada de emitir y revocar los certificados a los suscriptores y ocasionalmente publicar CRLs. También pueden soportar una variedad de funciones administrativas, aunque estas a menudo son delegadas en una o varias Autoridades de Registración.
Autoridad de Registración (RA)	Es un componente opcional que puede asumir un gran número de funciones administrativas de la CA. Normalmente se asocia con el proceso de registración del suscriptor, pero puede participar en varias otras áreas también.
Suscriptor – Sujeto – Entidad Final	Es un término genérico utilizado para denotar usuarios finales, dispositivos (por Ej. Servidores, routers), o cualquier otra entidad que se pueda identificar en el campo sujeto de un certificado de clave

	pública. Típicamente son los usuarios de los servicios PKI.
Parte Confiante	Es el usuario que recibe del suscriptor la información firmada digitalmente y necesita de la PKI para verificar la firma.
Repositorios	Es un término genérico empleado para referirse a cualquier método para el almacenamiento de certificados y CRLs, de tal forma que puedan ser recuperadas por los usuarios finales.

AUTORIDADES DE CERTIFICACIÓN

Las claves públicas son distribuidas bajo el formato de certificados de clave pública. La CA constituye la base fundacional de una PKI, ya que es la única entidad que puede otorgar esos certificados. Los certificados son firmados digitalmente por la CA que los otorga. Las CAs son también responsables de la publicación de CRLs, a excepción que esta tarea haya sido delegada en un "Publicador de CRL" (puede considerarse como una componente extra de la PKI; su existencia es opcional y a ella la CA le delegará la tarea de publicar las CRLs).

Proveen los siguientes servicios de manejo de certificados:

- Comprobación de la autenticidad de las personas para las cuales se emiten y se revocan.
- Emisión, revisión y publicación de certificados.
- Entrega, almacenamiento y archivo de certificados y listas de revocación.

A fin de asegurar la integridad de los datos contenidos y la identidad del usuario, la Autoridad de Certificación firma digitalmente los certificados que ella emite. Estos certificados pueden ser publicados en un repositorio o hacerse disponible por otros medios.

Un requisito fundamental para el funcionamiento del servicio es que la Autoridad de Certificación sea de confianza tanto para el firmante como para el receptor de la información firmada.

Una CA tiene la obligación y responsabilidad de asegurar la confiabilidad y la seguridad de la certificación. Para ello, debe cumplir con las políticas que se determinan en los requerimientos de manejo, de operación, de sistema y de facilidades.

Múltiples Autoridades de Certificación

No todas las entidades necesariamente deberán confiar en la misma CA para tener sus certificados. Una estructura entre múltiples CAs provee uno o más caminos de certificación (conocido como *path de certificación*) entre un suscriptor y una

aplicación que utiliza certificados; esta funcionalidad es crítica para tener una PKI utilizable.

El *path* es una secuencia de uno o más nodos conectados entre el suscriptor y la autoridad de certificación raíz. Autoridad de certificación raíz, se denomina a aquella CA en la cual la aplicación confía, importando y guardando su certificado de manera segura. El camino de certificación, incluye certificados de múltiples CAs; por ello se hace necesario desarrollar sistemas de clave pública que soporten la existencia de múltiples autoridades de certificación y las relaciones de confianza existentes entre ellas.

Existen distintos tipos de estructuras para soportar estas relaciones de confianza, y dependen de distintos factores: comunidad de usuarios, naturaleza de las aplicaciones que requieren los certificados, alcance geográfico, etc.

Un tipo posible es la denominada estructura *jerárquica top-down*: en esta estructura existe una CA de 'primer nivel', y cada CA certifica a sus 'hijos' (puede ser una CA de nivel inferior o un sujeto). Todos los usuarios deberán utilizar la CA de primer nivel como su CA raíz y requiere que todos los usuarios obtengan una copia de la clave pública de la CA de primer nivel antes de poder utilizar la PKI.

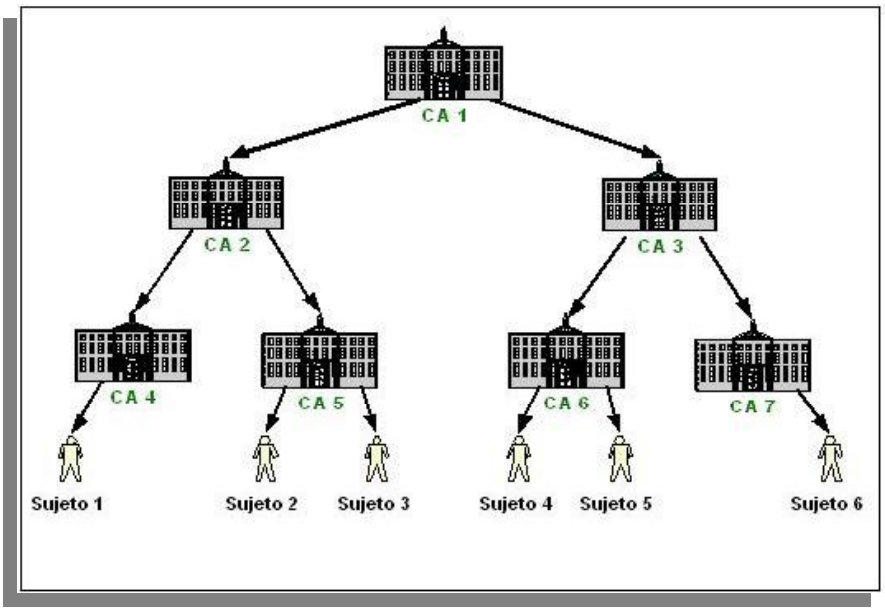


Fig. 7 – Estructura Jerárquica Top-Down

Como todos usuarios deben confiar completamente en la CA de primer nivel para todos propósitos, este tipo de la jerarquía puede ser poco práctico para una PKI global o muy extendida.

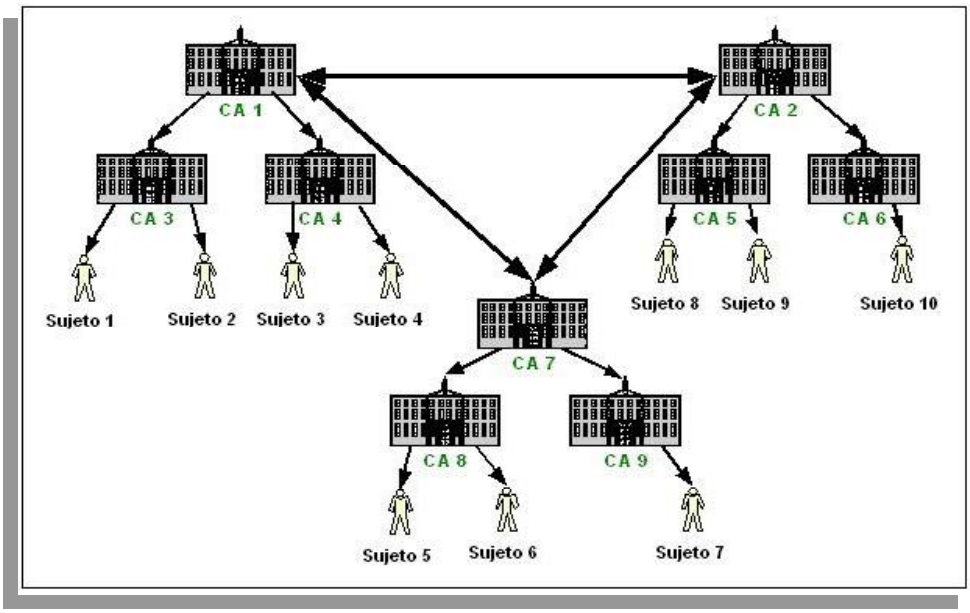


Fig. 8 – Estructura basada en Islas de Confianza

No sería muy realista pensar que toda la población de internet confiara en una única CA. La confianza se desarrolla en forma de "*islas de confianza*", dentro de las cuales la comunidad confía en una CA en particular (en el gráfico CA1, CA2 y CA7). A su vez cada CA Raíz de una isla, confía en la CA raíz de las demás islas.

AUTORIDADES DE REGISTRACIÓN

La Autoridad de Registración (RA) es un componente opcional que puede ser utilizado para descargar muchas funciones administrativas que la CA debe asumir en ausencia de la RA. En una primera vista, la RA se asocia con el proceso de registración del sujeto; esto incluiría la verificación de la identidad del sujeto que intenta registrarse con el PKI. Sin embargo, una RA puede realizar otras funciones, incluyendo:

- Validación de los atributos del sujeto que solicita el certificado,
- Verificación que el sujeto tiene posesión de la clave privada que será registrada (conocido como *prueba de posesión*),
- Generación de secretos compartidos para soportar los procesos de inicialización y certificación,
- Generación del par de claves (pública / privada),
- Realizar la interacción con la CA (o varias CAs) como intermediario de la Entidad Final (sujeto), incluyendo notificaciones de compromiso de claves y requerimientos de recuperación de claves,
- Validación de parámetros de claves públicas presentadas para registración.

Hay que destacar que aunque la RA puede aliviar muchas funciones de la CA, nunca puede ser el emisor de un certificado de clave pública. Ésta autoriza requerimientos, tanto de emisión como de revocación de certificados, y luego contacta a la CA para que esta última realice la acción que se le está solicitando. La CA cuenta con la suficiente autoridad para rechazar un requerimiento proveniente de una RA si

dicho requerimiento no cumple con lo impuesto por la política a seguir en sus prácticas.

Disponer de RAs puede proporcionar dos ventajas principales. En primer lugar, puede ayudar a reducir los costos generales; esto es así en grandes organizaciones, dispersas geográficamente que requieren la presencia física de sus usuarios antes de permitir ciertas actividades de PKI. Un ejemplo típico puede ser la registración de suscriptores, o también requerimientos de revocación de certificados, o recuperación de claves; para estos casos es necesario contar con autoridades de registración locales que realicen tal verificación. También pueden tenerse en cuenta otras consideraciones, como por ejemplo una organización que desea tercerizar los servicios de CA, manteniendo el control del proceso de registración. En segundo lugar, aliviando las tareas administrativas de la CA, permite a las organizaciones operar con sus CA de manera off-line, reduciendo así las oportunidades de ataques externos sobre la CA.

SUJETO – SUSCRIPTOR – ENTIDAD FINAL

Son conocidos también como *usuarios finales*, pero el término *entidad final* es mucho más genérico. Una entidad final puede ser un usuario final, un dispositivo tal como un router o un server, un proceso o algo que pueda ser identificado en el nombre sujeto de un certificado; también podemos verlos como los consumidores de los servicios de PKI. Existen casos inclusive donde un proveedor de servicios de PKI puede ser considerado una entidad final; por ejemplo, una RA es considerada una entidad final desde el punto de vista de una CA.

Los suscriptores, que serán ligados a certificados, deben registrarse antes de poder participar como miembros de una PKI. Esto implica un paso inicial de registración, seguido de la inicialización y certificación.

REPOSITORIOS

El término repositorio, a menudo se asocia con directorio, pero no es necesariamente el caso. Dentro del contexto de una PKI, un repositorio denota cualquier método para el almacenamiento y recuperación de información relacionada al PKI, tal como certificados de clave pública y CRLs. Un repositorio puede ser un directorio basado en X.500, con clientes que acceden vía LDAP (Lightweight Directory Access Protocol), o algo mucho más sencillo como ser la recuperación de un archivo plano en un servidor remoto vía FTP o HTTP. El grupo de trabajo PKIX, que trataremos con mayor detalle más adelante en este capítulo, ha desarrollado varios protocolos operacionales para facilitar la distribución de certificados y CRLs, incluyendo LDAP, HTTP y FTP.

También es posible aliviar ciertas funciones de los sistemas clientes a través de terceras partes confiables. Por ejemplo, el OCSP (ONLine Certificate Status Protocol) [Ref. 1] puede utilizarse para "consultar" a una tercera parte confiable sobre el estado de revocación de uno o más certificados.

FRAMEWORKS Y ESTÁNDARES PARA UNA POLÍTICA BASADA EN PKI

La interacción entre los participantes de una política basada en PKI está administrada por dos documentos: *Certificate Policity* (*CP*) y la *Certification Practice Statement* (*CPS*); estos, juntos, proporcionan la información que los participantes de una PKI pueden necesitar para asegurar el nivel de confianza que pueden tener en un

certificado y en una clave privada. Ellos proveen información sobre los parámetros de seguridad y de procedimientos que pueden influir en la "fuerza" que tiene un certificado. Por ejemplo:

- el proceso de identificación utilizado para ligar una clave pública con la identidad de un usuario, puede basarse en diferentes mecanismos de prueba (pasaporte, dirección de correo electrónico, etc).
- La clave privada puede almacenarse o protegerse solo con una contraseña, o en un hardware seguro (por ejemplo una smart card o token USB).

Los requerimientos relevantes son prescriptos en el *CP*. El *CPS* describe como una CA aplica esos requisitos. Una guía ampliamente adoptada sobre CPs y CPSs es la escrita por el IETF [Ref. 2] denominada "*Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*" [Ref. 3]

El *CP* es un documento de alto nivel de una política basada en PKI; su propósito es determinar el nivel de confianza que la CA se esfuerza por proporcionar, indicando que debe hacer la CA.

El *CPS* muestra el camino por el cual la CA aplica el CP, por ejemplo, como la CA cumple los requerimientos. Puntos importantes que debe publicar un CPS incluye una descripción detallada de:

- Procedimientos operativos, así como auditorias y controles para alcanzar los niveles de seguridad requeridos por la CA;
- Estipulaciones respecto a las obligaciones de la CA hacia el mercado;
- Medios de protección de la clave privada: smart-card u otros token permitidos, etc.;
- Documentación y otra evidencia utilizada para la autenticación de certificados

PKIX

En 1995, el IEFT creó el grupo de trabajo denominado PKIX (Public Key Infrastructure X.509), orientado a desarrollar estándares de internet necesarios para proporcionar una PKI basada en X.509 [Ref.4]. Uno de los aportes más importantes de este grupo, es la definición de los protocolos que gestionan interacciones con la PKI y el hecho de que dichas gestiones se realicen sobre diferentes "medios de transporte", entre ellos de forma on-line mediante conexiones TCP. Para esto, PKIX define una serie de operaciones, como así también el formato de los mensajes utilizados, por ejemplo para las tareas de inicialización/actualización y revocación (CMP/CRMF – [Ref. 5]), así como la validación de certificados a través del protocolo OCSP.

El grupo PKIX desarrolló documentos que cubren 5 áreas diferentes en relación a la puesta en marcha de las infraestructuras PKI. La primer área abarca las características de los estándares de criptografía de clave pública (PKC) X.509 v3 y CRL X.509 v2. La segunda, los protocolos operacionales relativos a confianzas para obtener información, como por ejemplo claves públicas o estado de ellas. Por ejemplo podemos mencionar LDAP (Lightweight Directory Access Protocol) y OCSP. La tercer área incluye los protocolos de manejo utilizados por las entidades para intercambiar información necesaria para el propio manejo de la PKI. La cuarta, informa acerca de las políticas y prácticas de certificación y adicionalmente áreas de seguridad que no están tratadas en el resto de PKIX. Finalmente, la quinta se refiere a los servicios de certificación y

time stamping que pueden brindarse, los cuales pueden utilizarse para construir servicios tales como no repudio.

Gracias a la contribución realizada por PKIX, es posible construir un verdadero soporte al comercio electrónico mediante PKI.

Interoperabilidad PKI

En una infraestructura PKI sus componentes deben interactuar e interoperar.

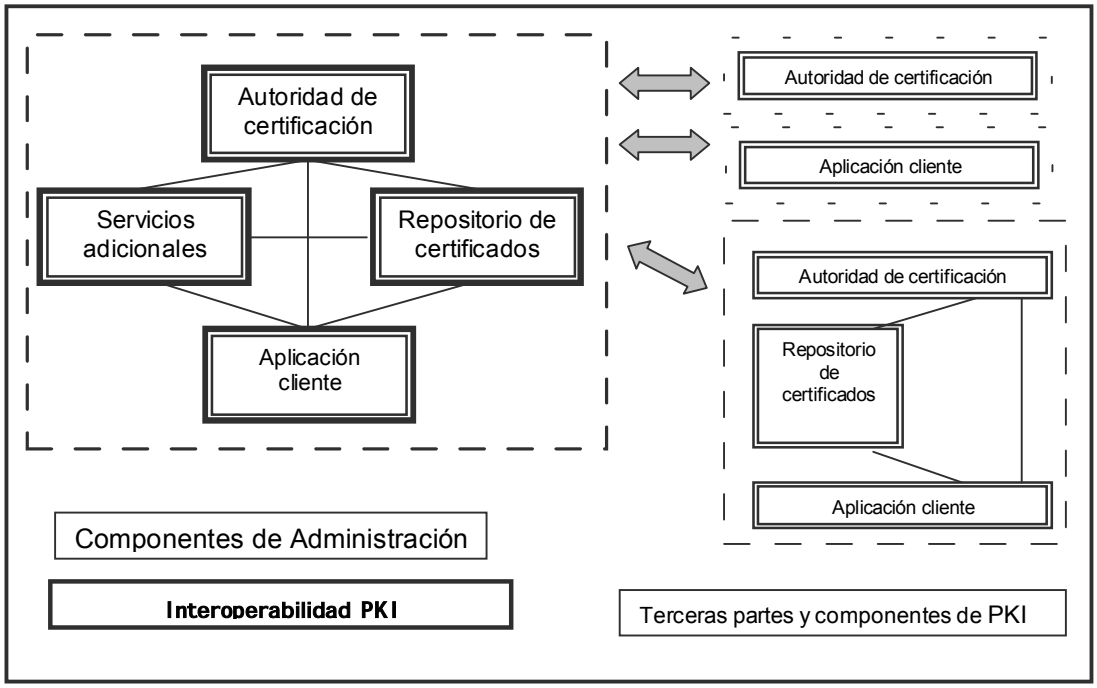


Fig. 4 – Interoperabilidad PKI

Para ello deben llevar a cabo una serie de operaciones:

- Generación de certificados: esto incluye la generación de certificados de clave pública y listas de revocación con un formato definido y sintaxis para asegurar la interoperabilidad con otras aplicaciones y otras PKIs. También incluye la generación de certificados "cross" para permitir la interoperabilidad entre CAs.

- Distribución de certificados: En el orden de conducir operaciones de clave pública, ya que un usuario debe tener acceso a otros certificados así como a las CRLs.

- Manejo de certificados: El manejo de claves y certificados representan las operaciones más comunes de una PKI. Los protocolos para requerir, renovar, hacer copias de seguridad, reparar y revocar claves y certificados requieren interoperabilidad entre aplicaciones cliente y la Autoridad de Certificación.[Ref. 6]

Esta página fue dejada en blanco de manera intencional.

Capítulo 3
Certificados Digitales

Definición

Como indicábamos anteriormente, el par de claves pública y privada no tienen una asociación intrínseca con una persona o sujeto, por lo cual se requiere un mecanismo que los vincule. Para ello existen los Certificados, que son un registro electrónico conteniendo la clave pública del dueño del certificado y la confirmación de que el mismo posee la correspondiente clave privada.

Una de las principales funciones entonces de un certificado es ligar un par de claves a una entidad particular. Esta entidad puede ser una persona (en este caso se lo conoce como suscriptor), un dispositivo de hardware o un proceso de software.

Los certificados de clave pública, que son emitidos por una Autoridad de Certificación (CA), contienen el nombre del usuario, la clave pública y otra información de identidad, como por ejemplo la dirección de mail, período de validez del certificado, etc. A fin de asegurar la autenticidad de la identidad y del contenido del certificado, la autoridad de certificación que lo emite firma el mismo digitalmente; dicha firma digital puede verificarse utilizando la clave pública de la CA. Por lo tanto, el nivel de confianza que el certificado provee, respecto del sujeto al que está certificando, depende fundamentalmente del nivel de confianza que se tenga respecto a la CA que lo emite.

Estos cerificados firmados por la CA, son publicados en directorios públicos o pueden hacerse disponible mediante otros medios, de manera que puedan ser recuperados cuando sea necesario para verificar firmas o encriptar documentos. De esta forma si deseamos enviar un correo electrónico cifrado a otra persona, podemos acceder a estos repositorios para obtener el certificado del destinatario, incorporarlo a nuestro cliente de correo y posteriormente, utilizando ese certificado, enviarle el mensaje cifrado.

Formatos de Certificados

Dado que los certificados son emitidos por distintas Autoridades de Certificación, y que los mismos pueden contener otros datos además de la clave pública y la identidad del sujeto, surge la necesidad de definir un estándar que permita la interoperabilidad entre aplicaciones que utilizan certificados emitidos por distintas autoridades.

En 1998, el Sector de Estandarización de la ITU[1], junto a la Comisión Electrotécnica de la ISO (IEC), publicó el formato estándar X.509 de los certificados, como parte de las recomendaciones del Directorio X.500.

La versión 1 del formato de certificados X.509 (v1) fue ampliada en 1993 por la versión 2 únicamente en dos campos, identificando de forma única el emisor y usuario del certificado; se buscó con esto soportar el control de acceso al servicio de directorio.

En 1996, surge la versión 3 que introduce cambios significativos en el estándar. El cambio fundamental es hacer el formato de los certificados y los CRLs extensibles. A

[1] *ITU: International Telecommunication Union*

partir de esta versión, quienes implementen X.509 pueden definir el contenido de los certificados como crean conveniente. Además se han definido extensiones estándares para proveer una funcionalidad mejorada.

Los campos que componen un certificado X509 versión 1 y X509 versión 2 son:

- *Versión*: indica la versión del formato del certificado (1, 2 o 3). Están previstas versiones futuras.
- *Número de serie*: especifica un identificador numérico del certificado único en el dominio de todos los certificados de clave pública emitidos por la autoridad de certificación. Cada certificado emitido por una CA dada debe tener un número de serie único. Cuando un certificado es revocado dicho número de serie es colocado en la lista de revocación firmada por la CA.
- *Algoritmo de firma*: Identifica el algoritmo usado por la CA para firmar el certificado. Es un número registrado con el reconocimiento de una organización estándar, y especifica tanto el algoritmo de clave pública como el de hashing. Sirve exclusivamente para identificar el algoritmo utilizado.
- *Nombre del emisor X500*: Este campo especifica la identidad de la CA que emite y firma dicho certificado.
- *Período de validez*: Muestra la fecha y la hora de comienzo de validez del certificado y la fecha y la hora de expiración del mismo.
- *Nombre X.500 del sujeto*: Especifica la identidad de la entidad a la cual pertenece el certificado.
- *Información de la clave pública del sujeto*: identifica dos piezas de información importantes: el valor de la clave pública que pertenece al sujeto y el identificador del algoritmo con el cual la clave pública es usada.
- *Identificador único del emisor*: Este campo fue agregado como parte de la definición de la versión 2 del estándar. Este campo opcional permite la reusabilidad del nombre del emisor.
- *Identificador único de sujeto*: Fue añadido como parte de la definición de la versión 2 del estándar de manera que identifique en forma única al nombre X509 del sujeto. Esto se debe al hecho de que un mismo nombre X509 puede haber sido asignado a más de un sujeto a la vez.

Certificados X.509 V3

El estándar, internacionalmente aceptado para Certificados Digitales, es el denominado X.509 Versión 3. Con esta versión, sucesora de la versión 2, no hace falta aplicar restricciones sobre la estructura del las CAs, gracias a la definición de las extensiones de certificados. Se permite que una organización pueda definir sus propias extensiones para contener información específica dentro de su entorno de operación.

Cada extensión posee tres campos: tipo de extensión, indicador de criticidad, valor del campo.

- *Tipo de la extensión*: indica el tipo de dato que contiene el valor del campo. Puede ser una cadena de texto, un valor numérico, etc. A fin de promover la interoperabilidad, los tipos de extensiones deben ser registrados en una organización de estándares internacional.
- *Criticidad*: es un flag de 1 bit; si la extensión está indicada como crítica, significa que el valor de la extensión contiene información de tal importancia que debe ser reconocida; si no es crítica, la misma puede ignorarse. Si una extensión está indicada como crítica y la misma no puede ser reconocida,

entonces el certificado debe descartarse. Las extensiones críticas son aquellas reservadas para información tan importante que deben ser interpretadas por todas las aplicaciones (como ejemplo podemos indicar como crítica la que previene del mal uso o uso inseguro de un certificado). Por ello, debe analizarse cuidadosamente cuando se agrega una extensión crítica a un certificado, ya que esto puede acarrear problemas de interoperabilidad con otros dominios de CA y otras aplicaciones.

- *Valor*: contiene los datos actuales para la extensión.

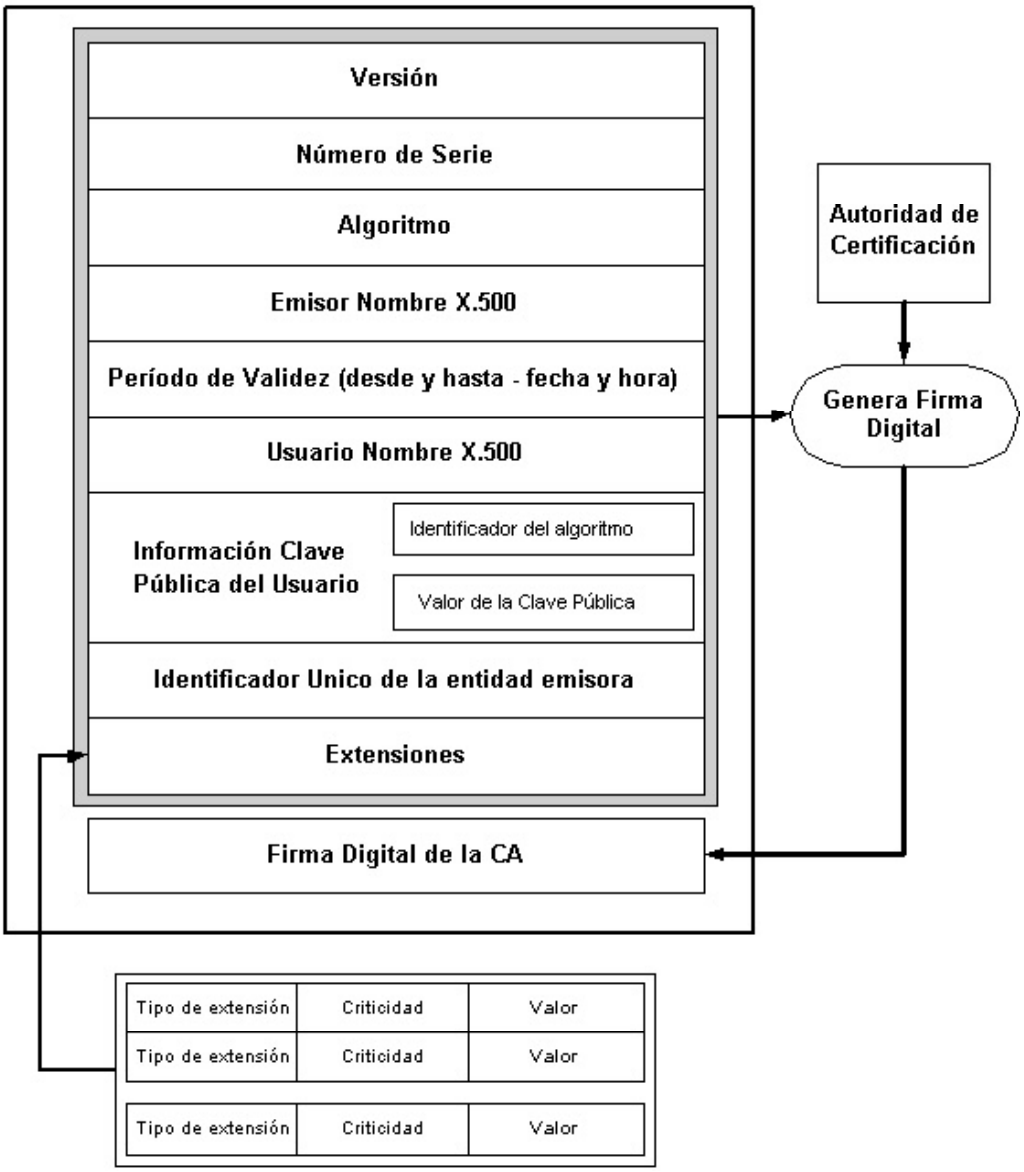


Fig. 5 – Formato de Certificado X.509 v3

Las extensiones estándar para los certificados de clave pública se pueden dividir en los siguientes grupos: *Información de la Clave*, *Información de la Política*, *atributos del usuario y de la CA* y *límites del camino de certificación*.

ITU e ISO/IEC [Ref. 2] desarrollaron y publicaron un conjunto de extensiones estándares en una versión corregida del estándar X.509 v3. Estas extensiones son:

- Límites Básicos: indica si el sujeto del certificado es una CA y la longitud máxima de la cadena de certificación.
- Política del certificado: contiene la política bajo la cual la CA emitió dicho certificado y los propósitos inherentes a la misma.
- Uso de la clave: indica el propósito de la clave pública presente en el certificado. Los posibles tipos de uso para la clave son: firma digital, no repudio, cifrado de claves, cifrado de datos, acuerdo de claves, firma de certificados, firma de CRLs, solo cifrado, solo descifrado.

Revocación de Certificados

Como indicamos anteriormente, uno de los campos en un certificado es la fecha de validez del mismo, período durante el cual debería estar vigente. Sin embargo y debido a diversos motivos, es posible que dicho certificado se torne inválido antes de cumplirse dicho período de vigencia. Por ejemplo cambio de asociación entre el sujeto y la CA – un empleado que finaliza su empleo en una organización -, cambio de nombre, compromiso o sospecha de compromiso en la clave privada.

Es deseable entonces, disponer de un mecanismo que permita mantener información de los certificados durante el período en el que ellos son válidos, y que permita a los sistemas que utilizan dichos certificados informarse sobre su revocación.

Podemos citar opciones que implementan esta funcionalidad:

- Listas de Revocación (CRL)
- OCSP (On-line Certificate Status Protocol) [Ref.3]
- XKMS (XML Key Management Specification) [Ref.4]

X.509 define un método para revocación de certificados, según el cual cada CA deberá emitir periódicamente una *Lista de Revocación de Certificados (CRL)*. Esta es una enumeración de los certificados que han dejado de tener validez antes de su fecha de expiración. Cada certificado revocado contiene un número de serie único mediante el cual se identifica en la CRL. Un sistema que utiliza certificados, además de controlar la firma y validez del mismo, deberá acceder a la lista más reciente de la CA y constatar que el número de serie del certificado no se encuentre en esta CRL.

Una CRL es una estructura de datos, firmada digitalmente por la CA que la publica, conteniendo la fecha y hora de publicación, identificación de la CA y los números de serie de todos los certificados que han sido revocados sin que haya expirado el período de validez de los mismos.

CRLs X.509

ITU-T [Ref. 5] e ISO/IEC publicaron en 1988 un estándar para CRL, conocido como CRL X.509 versión 1. Posteriormente se modificó para permitir los campos de extensión, dando lugar a la versión 2. Los campos básicos de una CRL X.509 son:

- *Versión:* indica la versión del formato que se utiliza. Si existe algún campo de extensión el valor aquí es 2, en otro caso se omite el campo.
- *Firma:* identificador del algoritmo utilizado para firmar la CRL.
- *Nombre del Emisor:* entidad que emite y firma la CRL.
- *Ultima actualización:* fecha y hora de emisión de la CRL.
- *Próxima actualización:* indica la fecha y hora de la emisión de la siguiente CRL. En caso que los sistemas que utilizan certificados emitidos por la CA conozcan la frecuencia de actualización del las CRLs, este campo puede omitirse.
- *Certificado de usuario:* número de serie del certificado revocado.
- *Fecha de revocación:* fecha efectiva de la revocación del certificado.

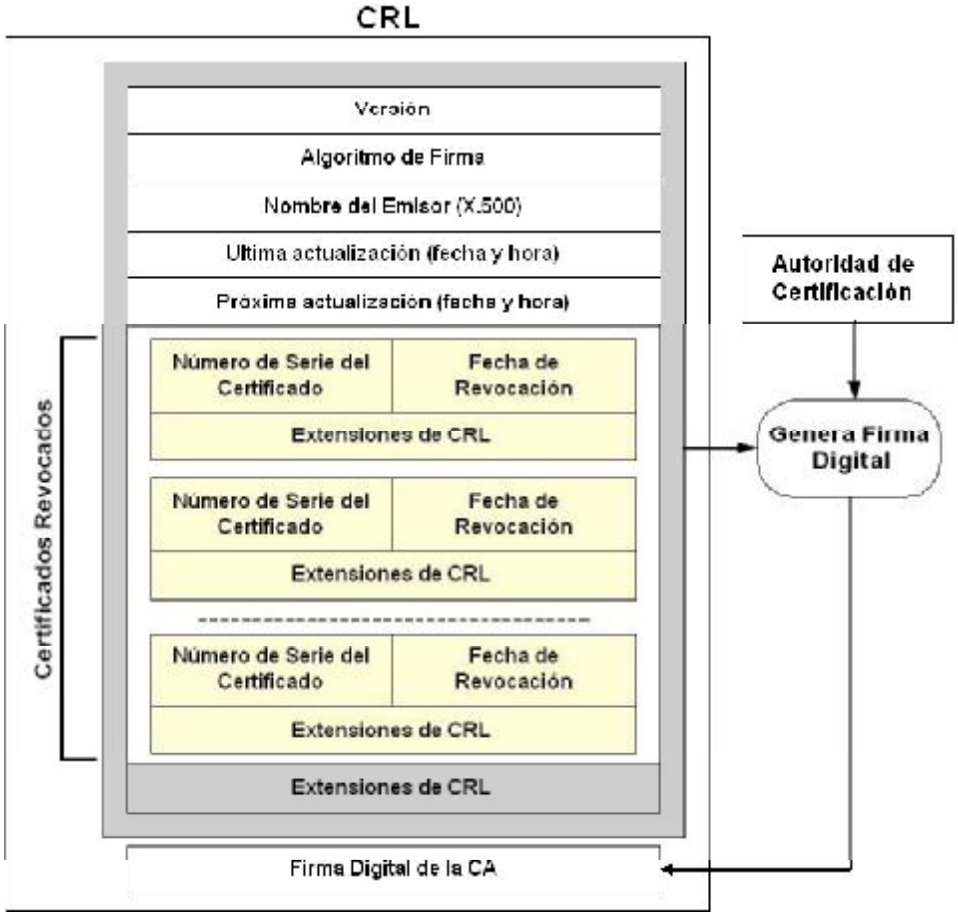


Fig. 6 – Formato de CRL X.509 v2

Las extensiones de CRL permiten que cada comunidad PKI pueda definir sus propios campos de extensión y codificar en ellos la información que necesiten en cuanto a las CRLs. Un ejemplo podría ser un campo que especifique el motivo por el cual se revocó el certificado. ANSI X9, ISO/IEC e ITU definieron un conjunto de

extensiones estándares para CRLs X.509 versión 2, utilizando para ello los mismos subcampos de los certificados X.509 v3; esto permite que las organizaciones puedan definir sus propias extensiones acorde a sus necesidades.

Existen diversos mecanismos que permiten distribuir la información de las CRLs. Podemos citar por ejemplo:

- Obtención de CRLs: una aplicación que utiliza certificados, accede a una CA y obtiene la CRL más reciente.
- Promoción de CRLs: aquí la CA puede promover CRLs a las aplicaciones que utilizan certificados toda vez que un certificado es revocado.
- Chequeo en línea (On-Line Checking Status Protocol): la aplicación que utiliza certificados efectúa una consulta en línea a la CA para obtener el estado de revocación de un certificado en particular.

Capítulo 4
Firma Digital

Definición

"*La firma digital es un conjunto o bloque de caracteres que viaja junto a un documento, archivo o mensaje y es capaz de acreditar quién es el autor o emisor del mismo (lo que se denomina autenticación) y que nadie haya manipulado o modificado el mensaje en el transcurso de la comunicación (asegura la integridad del mensaje)*". [Ref.1]

Las firmas digitales son utilizadas para firmar documentos electrónicos, asegurar la integridad de los mismos, certificar código de una aplicación, actualizar topología de redes públicas seguras, cifrar datos, etc.

Se generan mediante la aplicación de algoritmos matemáticos sobre un documento, y producen como resultado una representación única del mismo; la particularidad que presenta dicha representación es que la misma depende tanto del documento como de la persona que lo está firmando. Es decir que se obtienen dos firmas distintas si dos personas diferentes firman el mismo documento, o la misma persona firma dos documentos distintos.

Las firmas digitales se construyen utilizando encriptación asimétrica, con dos códigos: uno para firmar (clave privada) y uno para verificar (clave pública). Estos códigos son comúnmente denominados *par de claves*.

Procesos

La utilización de firmas digitales habitualmente involucra dos procesos:

- *Creación de la Firma Digital*: una vez establecido el mensaje a firmar, una función de hash en el software del firmante computa un resumen único, el digesto, para dicho mensaje (1). Este digesto es posteriormente encriptado, utilizando para ello la clave privada del firmante, obteniendo de este modo la firma digital del mensaje (2). El mensaje original es combinado con la firma obtenida (3) y el resultado – un mensaje autenticado – es enviado al destinatario (4).
- *Verificación de la Firma Digital*: este proceso consiste en validar la firma digital, utilizando para ello el mensaje original y la clave pública dada. Una vez que el mensaje es recibido por el destinatario, se separa el mensaje de la firma digital (5), la cual es desencriptada utilizando la clave pública (6). Por otro lado se aplica la misma función de hash al mensaje recibido y se obtiene el digesto (7) el cual se compara con el digesto recibido (8). Si ambos digestos son iguales, el mensaje no ha sido modificado durante el envío, con lo cual queda autenticado.

De esta forma el receptor puede validar:

1. Que la firma digital fue creada usando el correspondiente valor de clave privada,
2. Que el digesto calculado se corresponde con el recibido, que fue transformado en la firma digital durante el proceso de firmado.

Esto significa:

1. La clave privada del firmante fue la utilizada para firmar digitalmente el mensaje (*autenticación*),
2. El mensaje no fue alterado durante el envío (*integridad*).

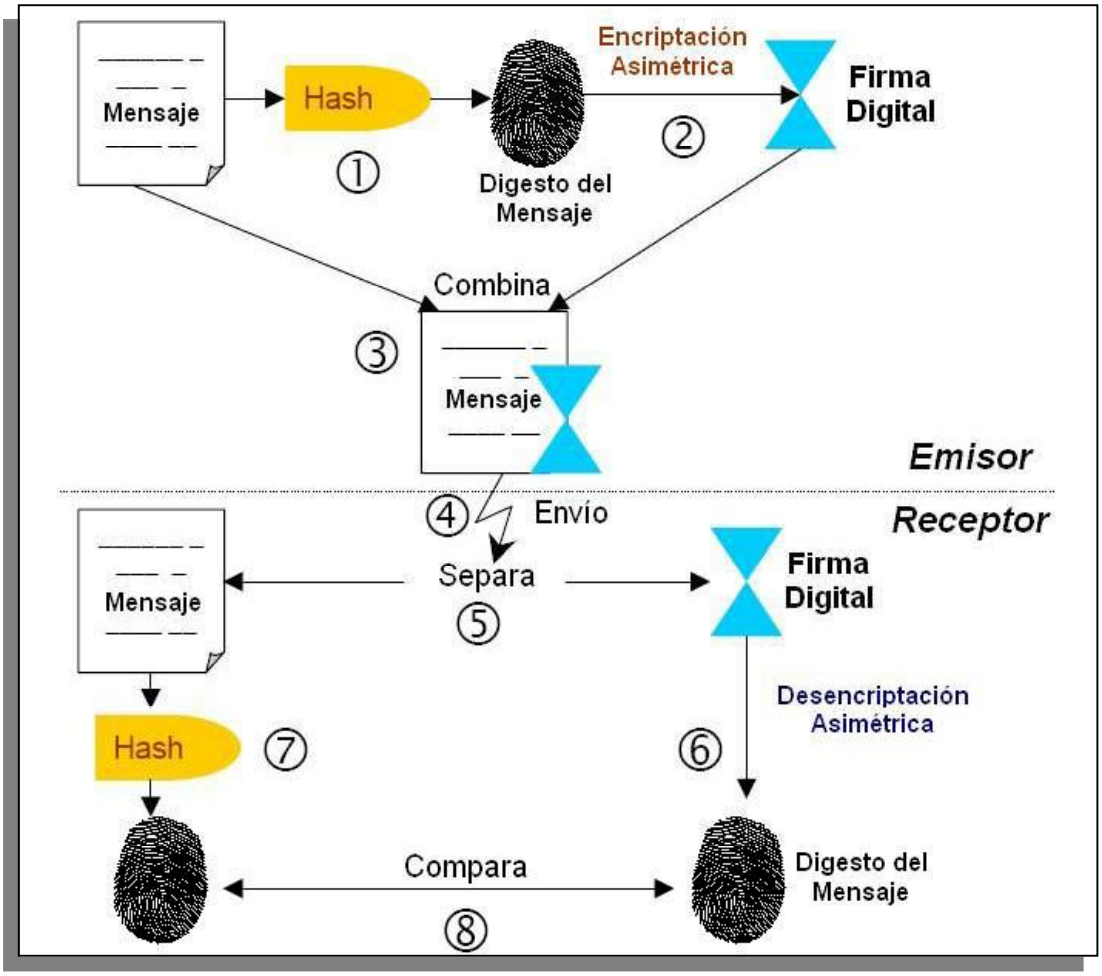


Fig. 4 – Procesos de una Firma Digital

Capítulo 5
J2EE - Java 2 Platform Enterprise Edition

Introducción

La dinámica actual exige a los desarrolladores cada vez más, el diseño, desarrollo y construcción de aplicaciones empresariales distribuidas a menor costo, con menores recursos y mayor velocidad. Para esto, la tecnología J2EE proporciona un enfoque basado en componentes que permite el diseño, desarrollo, ensamblado y deployment de ellas. La plataforma J2EE brinda un modelo de aplicaciones distribuidas en múltiples capas, posibilitando la reutilización de componentes, y proporcionando un modelo de seguridad unificado y un control flexible de las transacciones. No solamente permite disponer de rápidas soluciones hacia el cliente, sino que además estas soluciones basadas en J2EE son independientes de la plataforma, y no están atadas a productos ni APIs de ningún fabricante.

Básicamente, J2EE es una especificación creada por Sun Microsystems, para el desarrollo de soluciones empresariales y de e-commerce basada en Java. En ella se definen estándares para desarrollar aplicaciones en múltiples capas (multitier) y se proveen una serie de servicios y componentes a fin de facilitar la construcción de dichas aplicaciones.

Modelo

La plataforma J2EE, utiliza el modelo de aplicaciones distribuidas en múltiples capas. Esto significa que la lógica de la aplicación está dividida en *componentes* según la función, y los distintos componentes que forman la aplicación J2EE, son instalados en diferentes equipos, dependiendo de la capa a la cual pertenece – dentro del modelo de múltiples capas.

En la Fig. 1 vemos las partes de una aplicación J2EE:

- Los componentes de la capa Cliente corren sobre el equipo cliente
- Los componentes de la capa Web corren sobre el servidor J2EE
- Los componentes de la lógica del negocio corren sobre el servidor J2EE
- La capa de software de Enterprise Information System (EIS) corre sobre el servidor EIS

Mientras una aplicación J2EE puede consistir de tres o cuatro capas, las aplicaciones J2EE en múltiples capas son consideradas generalmente aplicaciones en tres capas porque están distribuidas sobre tres ubicaciones diferentes: equipo cliente, equipo servidor de J2EE y equipos de la base de datos o de los sistemas legacy.

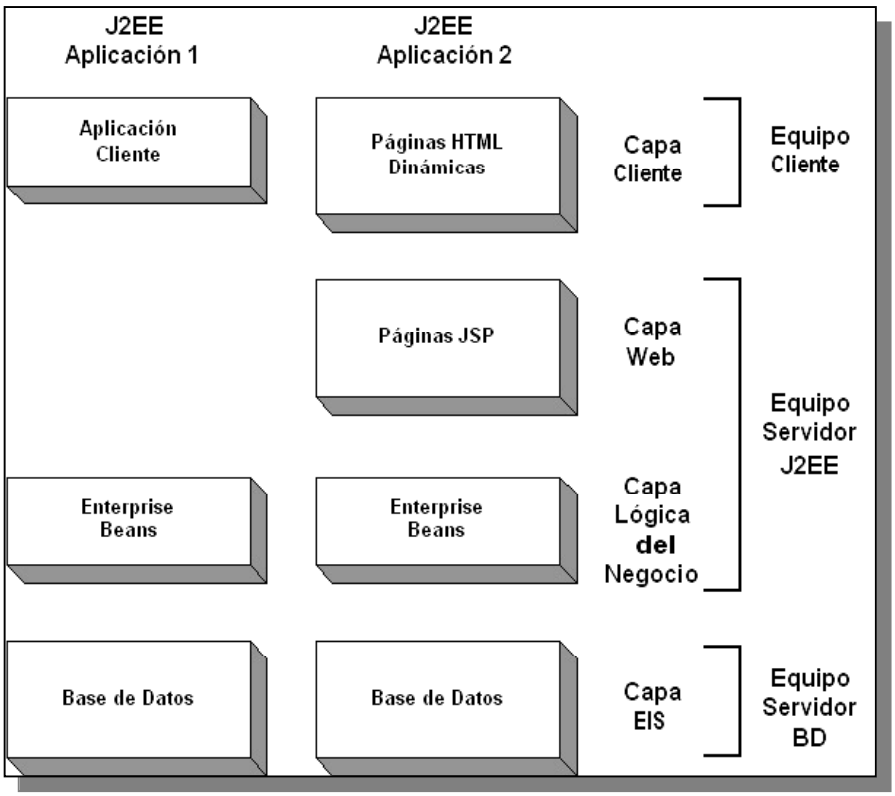


Fig. 1 – Aplicaciones en Múltiples capas

COMPONENTES DE LAS APLICACIONES J2EE

Las aplicaciones J2EE están formadas por *componentes*. Una componente J2EE es una unidad de software funcional independiente, que es ensamblada dentro de una aplicación J2EE, junto con sus clases y archivos relacionados, y que se comunica con otras componentes. La especificación J2EE defines los siguientes componentes:

1. Aplicaciones cliente, clientes web y applets son componentes clientes
2. Java Servlet y JavaServer Pages (JSP) son componentes web
3. Componentes Enterprise JavaBeans (EJB) (también llamados enterprise beans) son componentes de negocio (o componentes empresariales)

Los componentes J2EE están escritos en Java y compilados de igual forma que un programa Java; la diferencia entre los componentes y las clases estándares es que los componentes son ensamblados dentro de la aplicación J2EE y se ejecutan y gerencian por un contenedor.

1.- Componentes Cliente:

Una aplicación J2EE puede ser basada en web o no. Una aplicación cliente se ejecuta en la máquina del cliente si no es basada en web y un browser descarga páginas web y applets a la máquina del cliente para el caso de ser basada en web.

Aplicaciones Cliente

Una aplicación cliente corre sobre la máquina del cliente y típicamente tiene una interfase de usuario gráfica creada con APIs. Las aplicaciones clientes, directamente acceden a los enterprise beans que corren en la capa empresarial. Sin embargo, si los requerimientos de la aplicación J2EE del cliente lo precisan, puede abrir una conexión HTTP para establecer una comunicación con un Servlet que corre en la capa web.

Browsers Web

El browser del usuario descarga páginas web de HTML estático o dinámico, WML (Wireless Markup Lenguage) o XML (Extensible Markup Language) de la capa web. Las páginas web dinámicas son generalmente servlets o páginas JSP que corren en la capa web.

Applets

Una página web, descargada de la capa web, puede incluir applet embebido. Un applet es una aplicación pequeña escrita en java, que requiere de un web browser para ejecutarse (Netscape, Mozilla, Microsoft Explorer, etc.) y que se ejecuta en la Java VM instalada en el browser. No obstante, los sistemas del cliente pueden necesitar un *plugin* [Ref. 1] de Java y posiblemente un archivo de política de seguridad para que el applet se pueda ejecutar en el browser.

Las páginas JSP son las API para crear un programa cliente basado en web, ya que no serán necesarios plug-in ni archivos de política de seguridad sobre el sistema cliente. Además, las páginas JSP permiten diseñar aplicaciones más claras y modulares ya que permiten separar la programación de las aplicaciones del diseño de la página web; esto significa que las personas que diseñan la página no necesitan entender el lenguaje Java para hacer sus tareas.

Tanto las applets como las aplicaciones, requieren de la plataforma Java para ejecutarse. En el caso de las applets, la plataforma Java debe estar embebida en el web browser, mientras que para las aplicaciones requieren que esté disponible a través del sistema operativo o como un programa separado. En términos generales, las applets y las aplicaciones acceden a las mismas APIs.

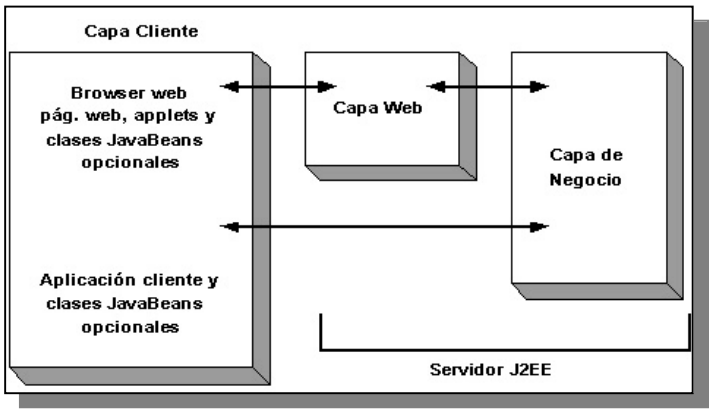
Las aplicaciones tienen acceso ilimitado a los recursos del sistema local: pueden leer y escribir en el File System local y remoto. Pero las applets, al ser descargadas a través de la red, están restringidas a acceder solamente al File System remoto, del servidor. Es la política de seguridad del *sandbox* [Ref. 2].

Arquitectura de componentes JavaBeans

La capa cliente, puede también incluir arquitectura de componentes basados en JavaBeans (componentes JavaBeans) para administrar el flujo de datos entre la aplicación cliente o applet y los componentes que corren sobre el servidor de J2EE. Los JavaBeans no se consideran componentes en la especificación J2EE. Estos JavaBeans escritos para la plataforma J2EE tienen variables de instancia y leen y setean métodos para acceder los datos en las variables de instancia.

COMUNICACIÓN CON EL SERVIDOR J2EE

En la figura siguiente vemos los distintos elementos que pueden componer la capa cliente. Las comunicaciones entre el cliente y la capa de negocios, que corre sobre el servidor J2EE, pueden ser directamente o – en el caso de un cliente corriendo con un browser- a través de páginas JSP o servlets que corren en la capa web.



Clientes Delgados (Thin Clients)

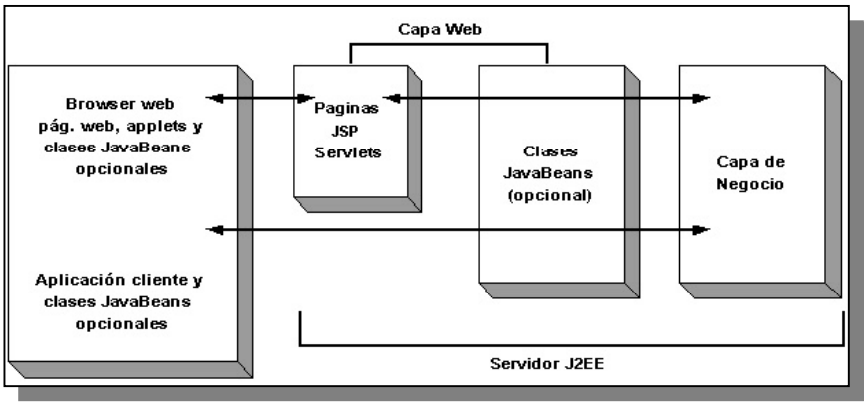
Las aplicaciones J2EE utilizan un cliente fino o delgado. Un cliente fino, es una interfase ligera que no realizan conexiones a bases de datos, tampoco ejecutan reglas de negocios complejas o ni se conectan con otros sistemas de la organización (aplicaciones legacy). Las operaciones más pesadas son realizadas por las componentes que se ejecutan sobre el servidor J2EE, disponiendo de la seguridad, velocidad y servicios de la tecnología J2EE.

2.- Componentes Web:

Los componentes web J2EE pueden ser páginas JSP o servlets. Los *servlets* son clases en lenguaje Java que procesan requerimientos y construyen respuestas de manera dinámica. Las páginas JSP son documentos basados en texto formadas por contenido estático y porciones de código escritos en Java para generar contenido dinámico. Cuando se carga una página JSP, un servlet en background ejecuta la porción de código y retorna la respuesta.

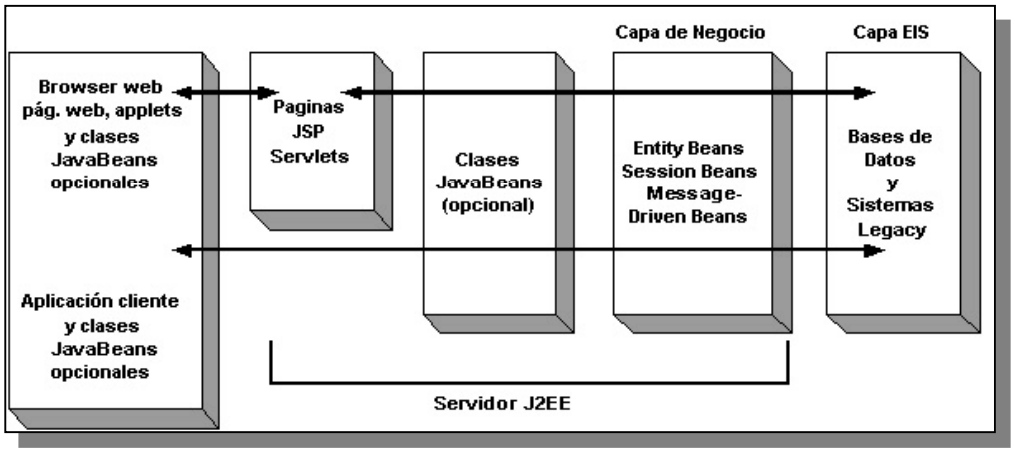
Las páginas HTML estáticas y los applets son ligadas con los componentes web durante el ensamblado de la aplicación, pero no se consideran componentes web de la especificación J2EE.

En la figura siguiente podemos ver, que la capa web puede incluir un objeto JavaBeans para manejar las entradas del usuario y enviar esa entrada al beans enterprise – que corre en la capa de negocios – para su procesamiento.



3.- Componentes Empresariales (o de Negocios):

Las codificación de las reglas de negocios, es decir la lógica que resuelve o satisface las necesidades específicas de un área particular, como pueden ser bancos, retail o finanzas por ejemplo, son manejadas por enterprise beans que corren en la capa de negocios y se ejecutan en una máquina servidor de aplicaciones J2EE. En la figura, podemos observar como un enterprise bean recibe datos de los programas del cliente, los procesa si es necesario, y los envía a la capa EIS para su almacenamiento. Un enterprise bean también recupera los datos almacenados, los procesa si es necesario, y los devuelve al programa del cliente.



Los Enterprise Java Beans (EJB) se ejecutan en un Contenedor EJB, que es un entorno de ejecución provisto por el servidor J2EE. Son componentes portables, posibilitando la construcción de nuevas aplicaciones ensamblando beans existentes. Las aplicaciones se ejecutan en cualquier servidor J2EE compatible.

Existen tres clases de beans de empresa:

- *de sesión (Session Beans)*: representa un cliente dentro del servidor J2EE; no es compartido y hay uno por cliente. Cuando el cliente finaliza la ejecución, el bean aparenta terminar y deja de estar asociado con el cliente. No es persistente, es decir que los datos no se guardan en una base de datos. Modelan procesos de negocio.

- *de entidad (Entity Beans)*: representa datos persistentes y almacenados en una base de datos relacional. Cada instancia de un bean se corresponde con una fila de una tabla de la base de datos; si el cliente finaliza o el servidor es cerrado, los servicios aseguran que los datos son guardados. Modelan datos del negocio (por ej. clientes, productos).

- *de manejo de mensajes (Message-driven Beans)*: permite procesar mensajes asincrónicamente. Combina las características de un bean de sesión y del Java Message Service (JMS), permitiendo a un componente de negocio recibir mensajes JMS de manera asincrónica. Son similares a los de sesión, pero solo pueden invocarse mediante mensajes. Actúa como un *listener de mensajes JMS*.

Capa EIS (Enterprise Information system)

La capa EIS, maneja los sistemas de información de la empresa, tales como los ERP, los sistemas de base de datos, y otros sistemas de información legacy. Los componentes de aplicaciones J2EE necesitan el acceso a los sistemas de información, por ejemplo para la conectividad con la base de datos.

Capítulo 6
Arquitectura J2EE

Introducción

Normalmente, las aplicaciones cliente en múltiples capas son complejas de desarrollar, ya que deben manejar transacciones y administrar estados, pooling de recursos y otros detalles de bajo nivel. La arquitectura basada en componentes J2EE facilita ello ya que la lógica de negocios se organiza en componentes reutilizables y el servidor J2EE provee servicios bajo la forma de contenedores para todos los tipos de componentes. De esta forma los desarrolladores deben concentrarse en resolver los problemas de la lógica de negocios y no de los servicios.

Contenedores y Servicios

En la arquitectura J2EE todas las componentes se ejecutan en el entorno de un Contenedor. Un contenedor es un ambiente de ejecución que provee de servicios a las componentes, como conectividad con la red, gerenciamiento del ciclo de vida de las componentes, seguridad y otros servicios especiales, como ser transaccionalidad, persistencia, pooling, etc.

Los componentes son instalados en sus contenedores y son la interfase entre un componente y la funcionalidad específica de la plataforma de bajo nivel que soporta el componente. Antes que un web, enterprise bean o componente de la aplicación cliente pueda ejecutarse, debe ser ensamblado en la aplicación J2EE y colocado en su contenedor.

A fin de proporcionar soporte para los Enterprise JavaBeans, Java Servlets y JSP, la plataforma J2EE define una cierta cantidad de servicios estándar y configurables, utilizados por los componentes J2EE; servicios tales como seguridad, administración de transacciones, búsquedas en Java Naming and Directory Interfase (JNDI), balanceo de carga y conectividad remota. Podríamos mencionar algunos, como ser:

- El modelo de seguridad de J2EE permite que uno mismo configure un componente web o enterprise bean como recursos del sistema, para que sean accedidos solamente por usuarios autorizados.

- El modelo de transacción de J2EE, permite que uno especifique las relaciones que componen una simple transacción así como que todos los métodos en una transacción sean tratados como una simple unidad.

- Los servicios de búsqueda JNDI proporcionan una interfase unificada para que múltiples nombres y servicios de directorio en la empresa, como componentes de aplicación, puedan acceder a nombres y servicios de directorio.

- El modelo de conectividad remota de J2EE administra las comunicaciones a bajo nivel entre clientes y enterprise beans. Después que un enterprise bean fue creado, el cliente invoca métodos sobre él, como si estuviesen en la misma máquina virtual.

Los contenedores, también manejan servicios no configurables tales como gerenciamiento de ciclos de vida de las componentes (enterprise bean y servlet), pooling de conexiones a las bases de datos, persistencia de datos y accesos a las APIs de la plataforma J2EE.

Tipos de Contenedores

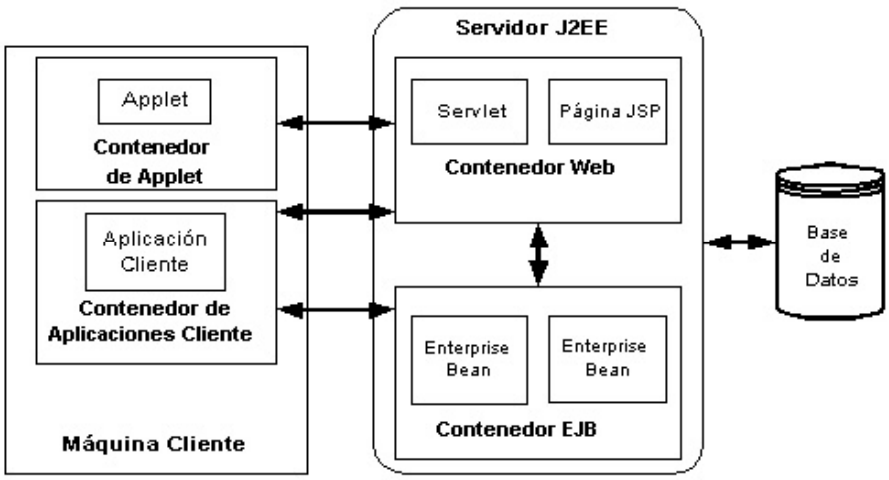
En la figura se muestran los distintos tipos de contenedores que se instalan:

- Un contenedor de Enterprise JavaBeans (EJB), que administra la ejecución de todos los enterprise beans de una aplicación J2EE. Los enterprise beans y sus contenedores se ejecutan sobre el servidor J2EE. Ejemplo de este tipo de contenedor es JBoss.

- Un contenedor web, que administra la ejecución de todas las páginas JSP y los componentes servlet de una aplicación J2EE. Los componentes web y sus contenedores corren sobre el servidor J2EE. Ejemplo: Tomcat.

- Un contenedor de aplicaciones cliente, que administra la ejecución de todos los componentes de las aplicaciones cliente sobre J2EE. Las aplicaciones cliente y sus contenedores corren sobre la máquina del cliente.

- Un contenedor de Applets; es un web browser que se ejecuta en la máquina del cliente y gerencia la ejecución de los applets. Es la combinación del web browser y los Plugin Java corriendo sobre la máquina cliente.



Podemos ver en la figura, que el cliente J2EE se comunica con la capa de negocios que se ejecuta en el servidor J2EE directamente (para el caso de las aplicaciones cliente), o vía servlets o páginas JSP para el caso de clientes que se ejecutan en un web browser (por ej. applets).

APIs de J2EE

API es la abreviatura de Aplication Programming Interface. Un API no es más que una serie de servicios o funciones que el Sistema Operativo ofrece al programador. Se refiere a toda Interface de Programación que un S.O. o aplicación exporta para ser utilizada por ella misma o por otras aplicaciones.

Visto desde la perspectiva de un lenguaje de alto nivel, el API aparece como un conjunto de procedimientos y funciones. Básicamente es una función con su nombre, argumentos y retorno.

Las APIs se agrupan por familias en librerías; en Windows estas librerías se encuentran en forma de DLL's (Dinamic Link Library), que no son más que archivos de código ejecutable que exportan funciones (o sea API's).

Se requiere del Java 2 Platform, Standard Edition (J2SE) SDK para ejecutar el J2EE SDK, el cual provee APIs para escribir componentes J2EE, herramientas de desarrollo y la máquina virtual Java (JVM).

La plataforma J2EE, proporciona las siguientes APIs [Ref. 1] que pueden ser utilizadas para aplicaciones J2EE:

- EJB (Enterprise Java Beans): como ya vimos, permiten implementar la lógica del negocio.
- JDBC: permite invocar comandos SQL desde métodos de lenguaje Java. Puede ser utilizado desde un servlet o JSP para acceder a la base de datos en forma directa, sin necesitar un bean empresarial.
- Java Servlet Techonology: permite definir clases servlet HTTP específicos.
- JSP (Java Server Pages): permite combinar pequeñas porciones de código Java con texto estático en un documento de texto.
- JMS (Java Message Service): es un estándar que permite a los componentes de una aplicación J2EE crear, enviar, recibir y leer mensajes de manera asincrónica.
- JTA (Java Transaction API): provee una interfase estándar para la demarcación de las transacciones.
- Java Mail: posibilita a las componentes de una aplicación J2EE utilizar el envío de mensajes por internet. Tiene dos partes: una interfase a nivel de aplicación utilizada por los componentes de aplicación para enviar mails, y una interfase para proveer servicios.
- JAF (JavaBeans Activation Framework): está incluido porque lo utiliza el Java Mail. Permite determinar el tipo de un dato cualquiera, encapsular el acceso a él, las operaciones disponibles sobre él y crear los componentes JavaBean apropiados para efectuar esas operaciones.

- JAXP (Java API for XML): posibilita a las aplicaciones a parsear y transformar documentos XML independientemente de una implementación particular de XML.

- JCA (Java Conector Architecture): es utilizado por los fabricantes de herramientas e integradores de sistemas para crear adaptadores (componentes de software) que soporten el acceso a sistemas de información que puedan ser introducidos dentro de cualquier producto J2EE.

- JAAS (Java Authentication and Authorization Service): permite a las aplicaciones J2EE autenticar y autorizar a un grupo específico de usuarios a ejecutarlas.

Capítulo 7
Conociendo EJBCA

Introducción

EJBCA es una Autoridad de Certificación construida puramente en Java. Desarrollada siguiendo los lineamientos de J2EE, constituye una CA basada en componentes, flexible e independiente de la plataforma, que puede ser operada en forma standalone o integrada dentro de cualquier aplicación J2EE de la empresa.

El proyecto se inicia en Noviembre de 2001, bajo el nombre *ejbca*. Actualmente se encuentra en estado de Producción / Estable. Es un Open Source con licencia LGPL [Ref. 1], miembro de FSF [Ref. 2]. El software posee certificado *OSI Certified Open Source Software*, esto es una certificación otorgada por el Open Source Initiative [Ref. 3] que se entrega a aquellos productos que cumplen con la definición de Open Source.

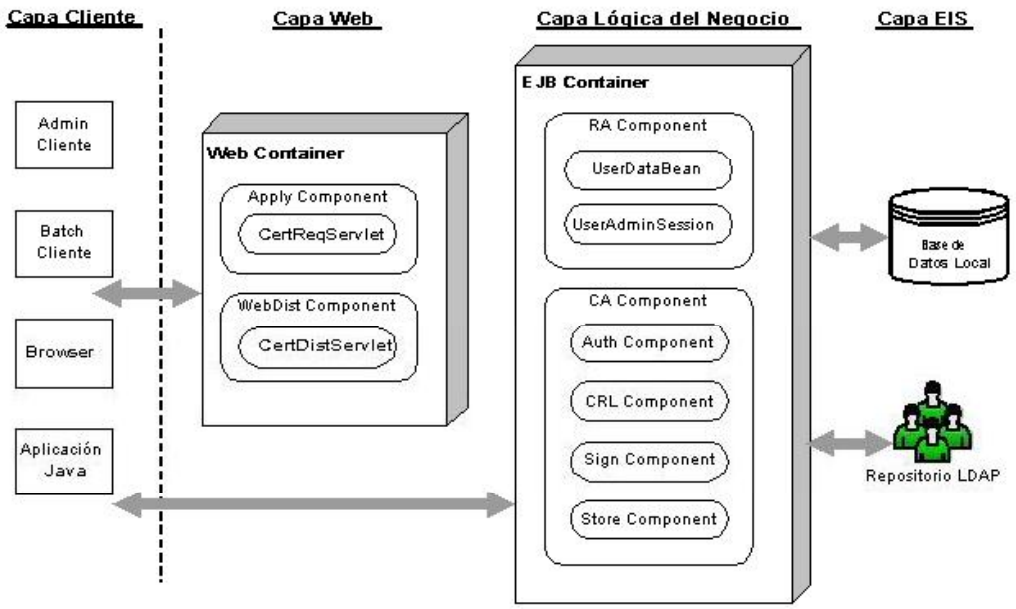
La página web del proyecto puede encontrarse en http://ejbca.sourceforge.net/. El staff de desarrollo comprende aproximadamente a 15 personas, entre fundadores y colaboradores de diversas partes del mundo.

Resumen de características [Ref.4]:

- Estado: Producción / Estable
- Licencia: GNU Library o Lesser General Public License (LGPL)
- Sistemas Operativos: 32-bits MS Windows (95/98/NT/2000/XP), todos POSIX [Ref. 5] (Linux/BSD/UNIX-like), OS independientes (escritos en un lenguaje interpretado).
- Lenguaje de Programación: Java, JavaScript
- Idiomas: Inglés, Sueco
- Interfase de Usuario: No interactiva (daemon), basada en Web
- Bases de Datos: ha sido testeado con Hypersoniq (hsqldb, default en JBoss), PostegreSQL 7.2 y 8.0 , MySQL, Oracle 8i, Sybase Enterprise System 11.

Arquitectura

Esta implementación de PKI posee una arquitectura en múltiples capas, basada en componentes, lo cual le permite una mayor flexibilidad de adaptación a los requerimientos. Está construida con la tecnología de enterprise Java Beans (EJB), permitiendo el reemplazo de los componentes de manera relativamente sencilla. Esta característica, le permite también correr como una CA standalone o integrada dentro de servidores de aplicaciones.



Plataforma

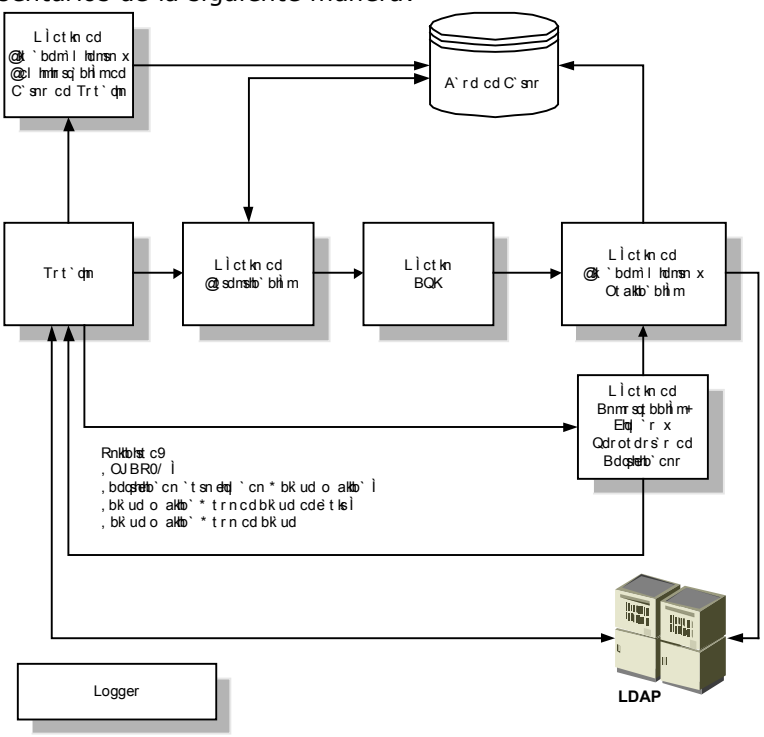
Al estar completamente escrito en Java, EJBCA debe correr en cualquier plataforma donde corra un servidor J2EE . Los desarrollos y testeos fueron efectuados sobre plataformas Linux y Windows 2000. Como servidores de aplicaciones ha sido probado con JBoss (default), con Bea WebLogic 7.x/8.x

Principales Características

- Licencia Open Source (LGPL).
- Construido en base a la especificación J2EE 1.3 (EJB 2.0).
- Arquitectura flexible, basada en componentes.
- Es posible construir dentro de una misma instancia de EJBCA una infraestructura completa (o varias) con múltiples CAs y niveles de CAs.
- Puede correr como una aplicación standalone o integrada a cualquier aplicación J2EE.
- Simple instalación y configuración.
- Posee un GUI Web para administración, utilizando autenticación fuerte.
- Administración por línea de comandos para scripts.
- Suscripción individual o producción batch de certificados.
- Los certificados server o clientes pueden ser exportados como PKCS12, JKS o PEM.
- Suscripción a través de browsers Netscape, Mozilla, Internet Explorer, etc.
- Mediante APIs abiertas y herramientas es posible la suscripción a través de otras aplicaciones.
- Notificación vía e-mail a los nuevos usuarios agregados por la RA
- Password aleatoria o manual para la autenticación inicial de usuario
- Provee un modulo Hard token, para integración con sistemas emisores hard token (smart cards) – Solamente disponible en PrimeCA (versión comercial de EJBCA).
- Soporta el Simple Certificate Enrollment Protocol (SCEP)
- Múltiples niveles de administradores, con privilegios especificados por grupos de usuarios

- Perfiles de certificados configurables para diferentes tipos y contenidos de certificados
- Perfiles de entidades configurables, para diferentes tipos de usuarios
- Cumple con los estándar X.509 y PKIX (RFC 3280)
- Maneja revocación y Listas de Revocación (CRLs)
- Soporta en su totalidad el Online Certificate Status Protocol (OCSP), incluyendo la extensión AIA [Ref. 6].
- Creación de CRL y puntos de distribución de CRL basadas en URL, acorde RFC 3280
- Almacenamiento de Certificados y CRLs en cualquier base de datos SQL (manejados por el servidor de aplicaciones)
- Múltiples editores opcionales para publicación de certificados y CRLs en LDAP y otros medios de almacenamiento
- Modulo de recuperación de claves para almacenar claves privadas para recuperación por usuarios seleccionados y certificados
- Arquitectura basada en componentes para publicación de certificados y CRLs a diferentes orígenes
- Arquitectura basada en componentes para varios métodos de autorización de entidades cuando otorgan certificados
- Fácil de integrar a grandes aplicaciones, permitiendo una óptima integración en los procesos de la organización

EJBCA está compuesto por módulos que interactúan entre ellos; gráficamente podemos representarlos de la siguiente manera:



Módulos de EJBCA

Consideraciones Generales sobre EJBCA

En este capítulo, daremos una serie de consideraciones y aclaraciones sobre terminología y aspectos generales de EJBCA, que nos ayudarán a comprender mejor esta herramienta.

EJBCA es una implementación open source de CA, escrita en ambiente Java/J2EE. El foco del proyecto es crear una solución de CA/RA flexible, independiente de la plataforma y escalable, cumpliendo varios requerimientos que puede tener una gran empresa, concernientes no solamente en aspectos de seguridad, sino también conectividad, delegación administrativa y seguimiento (loggin) de eventos.

PrimeCA es el nombre de la versión comercial del producto, y presenta las siguientes diferencias con EJBCA:

- Soporte para HSM, utilizado para almacenar credenciales secretas de las CAs.
- Disponibilidad para emitir hard tokens, tales como smart cards, mediante la utilización de otro producto – PrimeCard.

PrimeCA tiene es comercializado y tiene soporte y servicios profesionales a través de PrimeKey Solutions AB, empresa con sede en Estocolmo, Suecia.

Seguidamente, daremos una breve explicación de los conceptos de EJBCA tales como perfil de entidad final, perfil de certificados y como se relacionan unos con otros.

EJBCA implementa la parte de la CA de una PKI, de acuerdo a los estándares tales como X.509 y IEFT-PKIX, siguiendo sus lineamientos bien de cerca. La administración de la PKI tiene algunos conceptos específicos de EJBCA, a fin de implementar una gran flexibilidad.

Conceptos Generales

Autoridad de Certificación (CA): una CA emite certificados con el propósito de autenticar entidades. El nivel de confianza que uno le asigne a una CA es individual, por cada CA, y depende de las políticas y prácticas de las CAs.

CA Raíz (RootCA): una CA Raíz tiene un certificado firmado por ella misma. La verificación de otros certificados en la PKI, finaliza con el certificado auto-firmado de la RootCA. Como el certificado de la CA Raíz está auto-firmado debe, de alguna manera, configurarse como una raíz confiable para todos los clientes en la PKI.

CA Subordinada (SubCA): es una CA cuyo certificado está firmado por otra CA, que puede ser una SubCA o una CA Raíz. Al estar su certificado firmado por otra CA, no es necesario configurarla como raíz confiable; forma parte de una cadena de certificados que finaliza en la CA Raíz.

Autoridad de Registración (RA): es una función administrativa que registra entidades en la PKI. La RA es la encargada de identificar y autenticar entidades siguiendo las políticas de las CAs. Puede existir una o más RAs conectadas a cada CA en la PKI.

Entidad Final (End-entity): es un usuario, como por ejemplo cliente de e-mail, servidor web, navegador, gateway VPN, etc. No tiene permitido emitir certificados a otras entidades; se pueden ver como los nodos hojas de la PKI.

Punto de Distribución de CRL (CRL Distribution Point): es una extensión que se provee como información para los clientes que verifican un certificado. El valor es un URI que apunta a la CRL donde se puede verificar si el certificado está revocado. La CRL es emitida por la CA. Existen diferentes casos de CRL Distribution Points, y EJBCA soporte actualmente un URI.

Locador de Servicios OCSP (OCSP Service Locator): es similar en función al CRL Distribution Points, excepto que el OCSP recibe un requerimiento y envía de vuelta una respuesta de estado indicando si un certificado está revocado o no. El OCSP es también un URI. OCSP es utilizado por los clientes de una PKI para verificar certificados en tiempo real.

Extended Key Usage (Uso de Clave Extendida): esta extensión surgió cuando fue necesario un mayor control en el uso del propósito de los certificados. Algunas aplicaciones requieren realmente de esta extensión; Outlook por ejemplo, requiere que la protección de email sea establecida para permitirle al certificado ser utilizado para encriptar correo electrónico.

Microsoft Template Value (Valor de Plantilla Microsoft): es una extensión especial de Microsoft utilizada en certificados de Controladores de Dominio (AD Controller). Solamente puede ser usada en certificados emitidos por Controladores de Dominio.

Conceptos Específicos de EJBCA

Perfil de Certificado: determina contenidos de usos no específicos y el proceder de los certificados. La mayor parte son extensiones y aquí uno decide si una determinada extensión está presente y cuando ella es crítica o no. Algunas extensiones son llenadas con un valor, si ellas tienen el mismo valor para todos los certificados (por ejemplo el CRLDistributionPoint). En otros casos solamente se determina su presencia, cuando el valor está determinado específicamente por el usuario (por ejemplo SubjectAlternativeName). Aquí también se determina si esos certificados serán publicados y mediante cual publicador.

Perfiles de Entidad: determinan que datos pueden o deben ser mostrados a los usuarios conectados con este perfil. Algunos valores puede estar predeterminados, por ejemplo la Organización, o el DN.

Publicadores: Un Publicador almacena certificados en una ubicación centralizada. EJBCA soporta LDAP y Active Directory, pero también es posible crear plug-ins personalizados.

Perfiles de Hard Token: contiene información tal como longitud de la clave utilizada en smart cards, perfiles de certificados que pueden ser utilizados, etc. (*)

Emisores de Hard Token: representa una ubicación física en la cual se encuentra corriendo PrimeCard y donde los hard tokens pueden ser emitidos. (*)

(*) Estos conceptos se brindan solamente para información, ya que los mismos no se pueden utilizar en la versión EJBCA, sino que son exclusivamente de PrimeCA. Sin embargo, en algunas pantallas de la Web de Administración se pueden ver.

Capítulo 8
Construcción y Configuración General

Construcción de EJBCA

La versión instalada y configurada para este trabajo fue EJBCA 3.0.7

Para la construcción y ejecución, son necesarios:

JDK 1.4.x o 1.5.x:	Se utilizó J2SE v 1.4.2_08 SDK
JBoss 3.2.x o 4.0.x:	Se utilizó JBoss 4.0.1 SP1
Ant 1.6.x para la construcción:	Se utilizó Ant 1.6.4

A fin de permitir la utilización de encriptación fuerte y/o passwords de longitud mayor a 7 caracteres en las claves, se instalaron los archivos de políticas JCE *'Unlimited Strength Jurisdiction Policy Files'* del SDK [Ref.1]. Estos archivos de políticas se encuentran en el mismo sitio de Sun donde se descarga JDK 1.4 [Ref.2]

El producto EJBCA, tiene incluidos una serie de scripts (cmd para Windows y sh para UNIX) que permiten efectuar distintas operaciones de instalación, construcción, configuración y testeo.

En primer lugar se descomprime el archivo descargado en un directorio.

Posteriormente para la construcción inicial, utilizamos el script *ant*

Se debe setear la variable de ambiente JBOSS_HOME apuntando al directorio raíz de JBoss (generalmente /JBoss-<versión>).

Esto es para que el script que efectúa el deploy sepa donde deberá copiar los archivos; de esta forma ellos son copiados al directorio
$JBoss_HOME/server/default/deploy

También es posible construir javadoc sobre todas las APIs usadas, mediante:
ant javadoc
Esto genera un directorio con la información de todas las APIs, la cual es posible browsear con un navegador.

API Doc

Consideraciones previas a la Configuración

EJBCA ha sido testeado con distintas bases de datos, pero en la configuración por default que éste trae, se utiliza Hypersonic. Esta base de datos open source, es la que viene predefinida en el servidor de aplicaciones JBoss. Sin embargo existen dos razones fundamentales para no utilizarla en ambientes de producción:

1. Hypersonic, es una base de datos residente en memoria, con lo cual con el tiempo consumirá mucha memoria. Si se otorgan un gran número de certificados, esto seguramente se transformará en un problema.
2. Hypersonic no soporta full SQL, en particular la sentencia ALTER. Cuando se libera una nueva versión de EJBCA, no es posible crear scripts que actualicen la base si algunas tablas cambian, lo cual hará los upgrades mucho más difíciles.

Por estos motivos es altamente recomendable la utilización de alguna de las otras alternativas disponibles. En nuestro caso particular, hemos instalado y configurado MySQL. Optamos por MySQL por ser la base de datos Open Source más popular y desarrollada existente en la actualidad. Esta disponible bajo licencia pública general GNU.

Conexión con la Base de Datos:

Se instaló y configuró MySQL Server Versión 4.1 y para establecer la comunicación de JBoss con MySQL se utilizó el driver JDBC [Ref. 3] *mysql-connector-java-3.0.16-ga-bin.jar* [Ref.4].

Configurar EJBCA:

Para cambiar la configuración por default de la base de datos, se debe cambiar el DataSource en el archivo jbosscmp-jdbc.xml y todas las apariciones de 'java:/DefaultDs' por la de la base de datos seleccionada. Existe un script que ayuda a realizar esta tarea ("ant replaceDS"), en el cual se debe indicar el tipo de base de datos (mysql, oracle, mssql,... etc.) y el datasource a utilizar. Finalmente se debe construir y efectuar el deploy para que EJBCA tome los cambios necesarios y los incluya en el servidor de aplicaciones.

Luego de la instalación y configuración del servidor MySQL, definimos dentro de este una base de datos para almacenar las tablas de EJBCA. Hemos creado una base llamada *ejbca*, definiendo un usuario y password para que JBoss pueda realizar la conexión. Existe un archivo, mysql-ds.xml el cual sirve para configurar el datasource de MySQL; en este se definen usuario, password, nombre jndi [Ref. 5] del datasource, url de conexión, etc.

Una vez copiado el driver de conexión a la base en el servidor JBoss, se debe iniciar JBoss, construyendo así las tablas necesarias de la aplicación en la base de datos *ejbca* creada dentro del servidor MySQL.

TABLAS PRINCIPALES DE EJBCA

Tabla	Descripción
accessrulesdata	almacena las reglas de acceso por cada grupo de administradores y CAs que componen las PKIs
adminentitydata	almacena las entidades que cumplirán funciones de administración de las CAs y Ras.
admingroupdata	almacena los grupos administrativos por cada CA, incluido el ´superadministrador´.
adminpreferencesdata	almacena las preferencias de configuración del admin-GUI (lenguajes, letras, colores, etc.) de los administradores.
cadata	contiene los datos de las CAs (id, nombre, subjectDN, fecha expiración, etc).
certificatedata	almacena los datos de los certificados emitidos (huella digital, subjectDN, firma de la CA emisora, expiración, etc.)
certificateprofiledata	almacena los perfiles de certificados que se emitirán de acuerdo al tipo de certificado. (Ver Nota aclaratoria)
crldata	contiene la CRL.
endentityprofiledata	almacena los perfiles de certificados de las entidades finales. (Ver Nota aclaratoria)
globalconfigurationdata	contiene la configuración global de la GUI, establecida por el ´superadministrador´ (Título, banners, control sobre las RAs, etc.)
keyrecoverydata	contiene datos de los certificados para la recuperación de claves.
logconfigurationdata	almacena las preferencias en la configuración de los logs para las CAs.
Logentrydata	almacena los logs
Publisherdata	almacena los datos de los publicadores (LDAP, AD, etc).

Userdata	Contiene datos de los usuarios (administradores y entidades finales), incluido el ´superadmin´.

Nota sobre los Perfiles de Certificados:
Como hemos observado en las tablas de EJBCA, existen dos tipos de perfiles: de *Certificado* y de *Entidad Final*; la diferencia entre ellos está dada por:

> Un Perfil de Certificado, contiene atributos que son constantes para determinada categoría de certificados
>
> Un Perfil de Entidad Final, especifica atributos que cambian para cada usuario, por ejemplo DN y nombre del sujeto.

Iniciando EJBCA

Una vez lograda la conexión con la base de datos y creadas las tablas se inicia el proceso de construcción de la PKI mediante EJBCA.

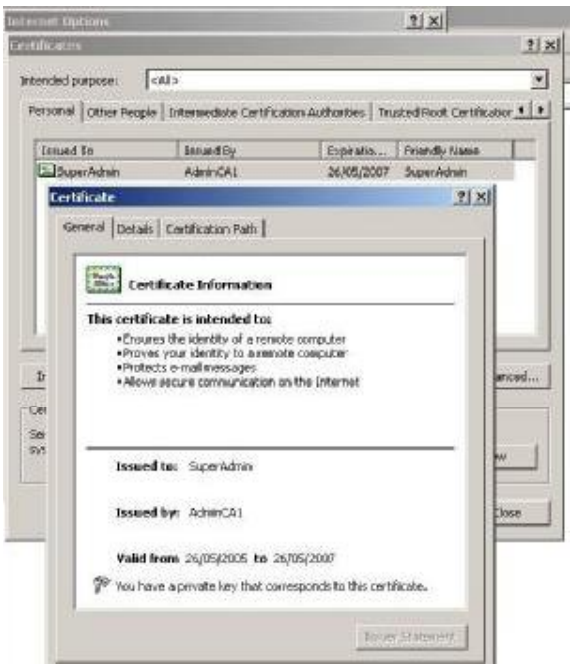
Una CA puede ser una CA Raíz (root CA), una CA subordinada a otra CA en EJBCA o una CA subordinada a una CA externa a EJBCA. Inicialmente, debemos configurar una CA raíz, a fin de poder acceder al *admin-GUI* (GUI Web utilizada para administración de EJBCA).

Para ello, existe un script (*install*) que nos permite establecer la primer CA Raíz. Esta CA, tiene por finalidad crear el primer *superadministrador* y el certificado para el servidor SSL de administración vía web (admin-GUI). Una vez configurado el servidor web administrativo, podremos crear otras CAs y administradores. Los datos necesarios para esta primer CA son los siguientes:

Dato Solicitado	Valor Ingresado	Descripción
CA short name	AdminCA1	Nombre corto con el que se conocerá la CA. Tiene propósitos solamente administrativos
Distinguished Name CA	CN=AdminCA1,O=Tesis EJBCA Info.UNLP,C=AR	DN de la CA. Se utiliza para crear un nombre único de la CA a nivel mundial.
Keysize of the CA	2048	Longitud en bits de la clave
Validity in days for the CA	3650	Cantidad de días en que será válido que la CA otorgue certificados.
Policy id of the CA	2.5.29.32.0	Determina cuales políticas de PKI utilizará la CA
Computer name of CA server	Localhost	Nombre del servidor al que se accede para la interfase administrativa.
Distinguished name of the SSL server certificate	Localhost	Es el DN del servidor SSL utilizado
Password for the super administrators keystores	ejbca	Password para almacenamiento de la clave del super administrador

Una vez completados y confirmados los datos requeridos, se crea e inicializa la CA, se genera el certificado para el superadministrador , se crea y publica la CRL inicial.

Para poder acceder a la interfase administrativa de EJBCA, se deberá importar el certificado almacenado al browser. Durante el proceso de instalación, se creó un token temporario para el administrador, el cual se debe utilizar para comenzar con la configuración inicial. Luego uno deberá crear nuevas claves para los administradores, con privilegios administrativos mejor definidos en base a las propias necesidades.



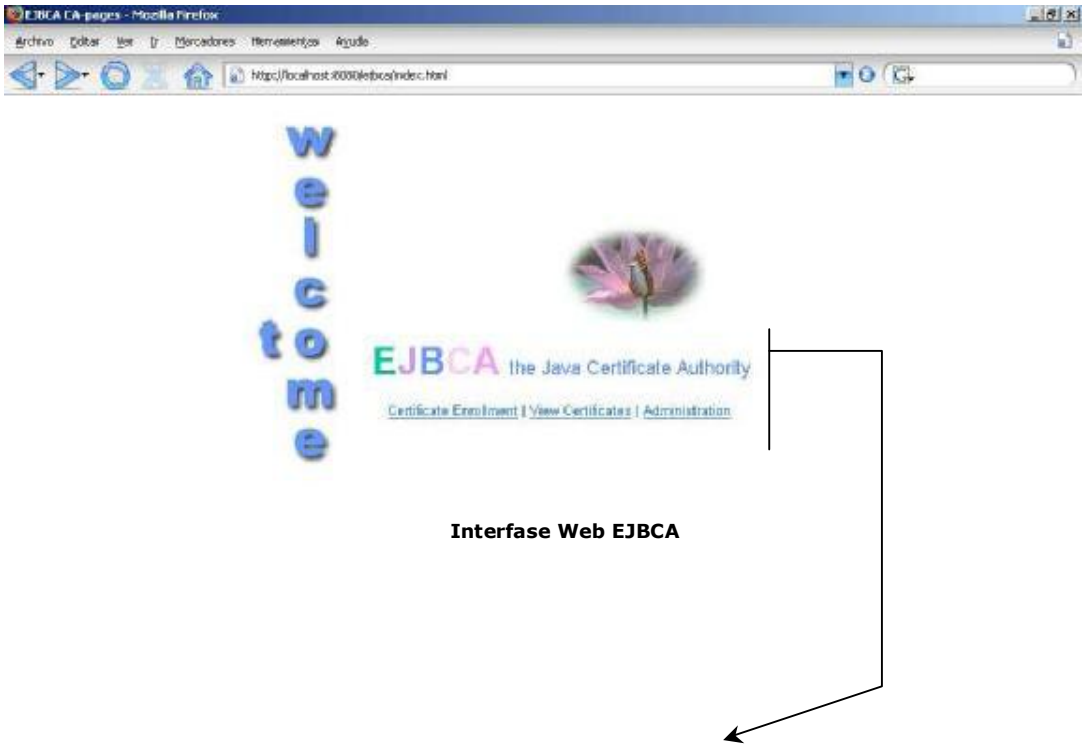
Una vez completada la importación del certificado, se procede a configurar la PKI.

Esta página fue dejada en blanco de manera intencional.

Capítulo 9
Interfase Web de EJBCA

Introducción

EJBCA cuenta con una interfase web pública, mediante la cual el usuario puede hacer uso de los servicios de certificación, como ser requerir un certificado, efectuar búsquedas de certificados, consultar lisas de revocación, etc. [Ref.1]. Esta interfase sirve también para administrar la PKI, debiendo identificarse mediante usuario y contraseña para poder ingresar a esta última opción.



Interfase Web EJBCA

Certificate Enrollment	{ View Certificates	{ Administration
Permite instalar un certificado propio en el browser	Permite ver y buscar certificados, comprobar revocación, traer certificados de CAs y Listas de Revocación	Ingresa a herramienta de Administración

Ver Certificados:

Mediante esta opción es posible buscar y ver los datos de un certificado mediante el Subject DN, instalar listas de revocación (CRLs), verificar si un certificado se encuentra revocado.

Opción View Certificates

Administración:

Se ingresa a la herramienta de administración general de la PKI. Para ello se debe seleccionar el usuario y suministrar la contraseña de acceso correspondiente. Esta opción será tratada en detalle en el Capítulo siguiente.

Capítulo 10
Administración mediante Interfase Web

Introducción

EJBCA dispone de dos interfases de administración: Web basada en GUI o mediante Línea de Comandos.

EJBCA permite definir grupos de administradores y sus roles (Administrador de CA, Administrador de RA, Supervisor, Súper Administrador). Dependiendo del rol que tenga asignada la persona que ingresa a la Web de administración, serán las opciones habilitadas y que entidades de la PKI puede administrar y o ver.

Roles:

- SUPER ADMINISTRADOR: él tiene el acceso general al sistema.
  El rol de "Súper Administrador" es para:
  Editar la configuración del sistema
  Administrar CAs
  Administrar publicadores (LDAP, AD, Custom)
  Crear "Administradores de CA"
  Por supuesto que con este rol es posible realizar todo tipo de configuración (actuar como Administrador de CA, o de RA por ejemplo).

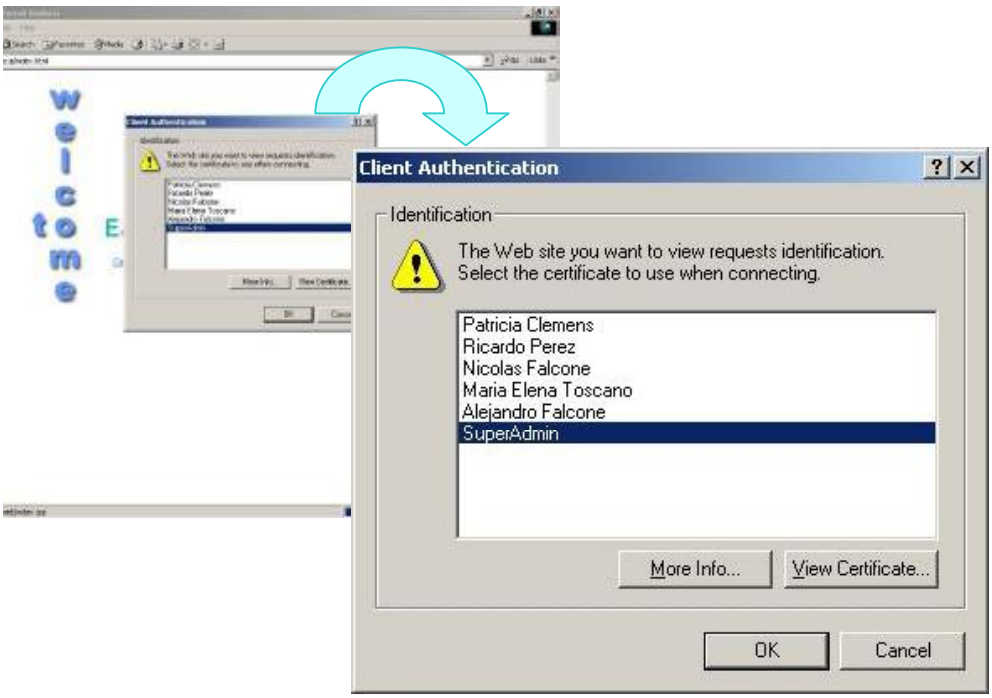
- ADMINISTRADOR de CA: las personas con este rol tienen a su cargo
  Administrar perfiles de certificados
  Administrar perfiles de las entidades finales
  Configuración del log
  Creación de los Administradores de RA
  Todo perteneciente a su propia CA

- ADMINISTRADOR de RA: los administradores de RA tienen a su cargo
  Creación y Edición de Entidades Finales
  Revocación y eliminación de Entidades Finales
  Pueden ver las Entidades Finales existentes y su historia.

- SUPERVISOR: puede
  Ver las Entidades Finales creadas
  Buscar en el log para ver quien hizo algo en particular.

Descripción de la Interfase Web de Administración

En primer lugar se debe seleccionar el usuario con el cual se ingresará a la Administración. [Ref.1]



A la pantalla con todas las opciones disponibles se ingresa mediante un usuario con rol de Súper Administrador.

Desde esta pantalla inicial y como Súper Administradores podemos comenzar a configurar y construir nuestra PKI.

Básicamente la pantalla de administración se encuentra dividida en 5 grupos, según la funcionalidad de los mismos:

Funciones de CA
Funciones de RA
Funciones de Log
Funciones del Sistema
Preferencias

Estos grupos serán visibles según la jerarquía del usuario que ha ingresado, es decir el grupo administrativo al que pertenece. Podemos describir a grandes rasgos las opciones y funcionalidad de cada uno de estos grupos.

Pantalla Inicial para Administración de EJBCA

1. CA Functions: (Funciones de CA)

1.1 Básicas: permite visualizar las CAs creadas, ver información sobre ellas, sus certificados; muestra información sobre la CRL y permite crear una nueva CRL (actualizar la existente). Permite también descargar los certificados.

Funciones de CA – Funciones Básicas

1.2 Edición de Perfiles de Certificados: permite administrar los perfiles de certificados que utilizará la PKI. Puede crear un nuevo perfil, eliminar o modificar un perfil existente.

Funciones de CA – Edición de Perfiles de Certificados

Aquí definiremos los atributos necesarios para cada perfil que necesitemos. Algunos de esos atributos son:

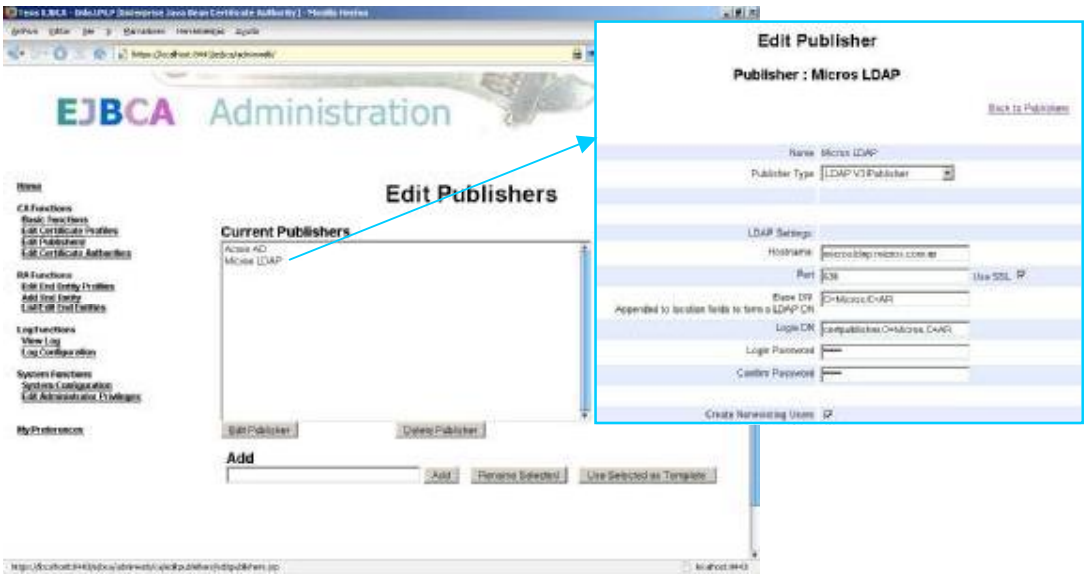
Validez en días del certificado, Uso de Limitaciones básicas y criticidad de éstas,

Uso de CRL y URL en caso afirmativo, Políticas de Uso de Certificados, Uso de OCSP y URL correspondiente, Uso de la Clave (No Repudio, Firma Digital, Cifrado de datos, etc), Uso de Clave Extendida (Autenticación de servidor,

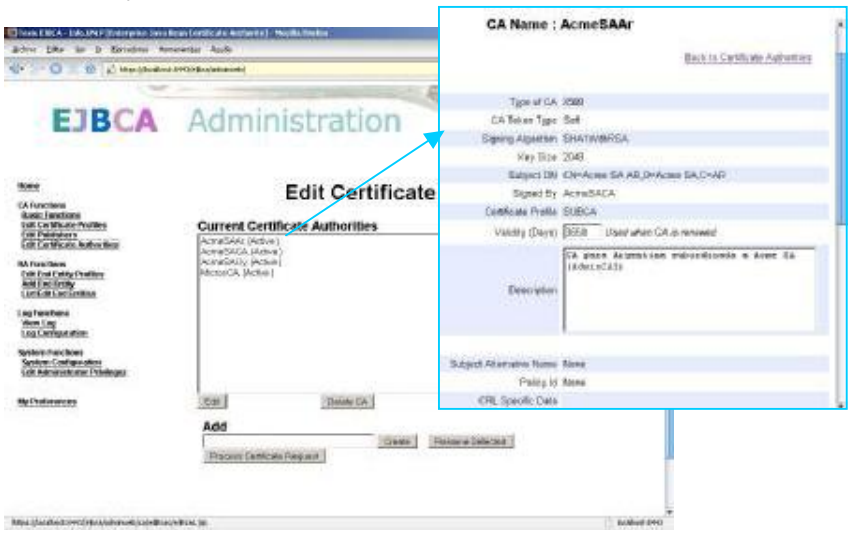
autenticación de clientes, firmado de código, protección de e-mail, MS Smart Card logon, etc), Longitudes disponibles en bits(512,1024,2048,4096), Selección de las CAs que utilizarán este perfil , etc.

En la confección de nuevos certificados es posible utilizar los existentes como plantillas, agilizando así el proceso de creación de los nuevos.

1.3 Edición de Publicadores: permite administrar las diferentes formas de publicar los certificados. EJBCA soporta implementaciones de LDAP y Active Directory, pero es posible crear plug-ins customizados.



1.4 Edición de Autoridades de Certificación: permite administrar las diferentes CAs que componen la PKI. De igual forma que en casos anteriores, es posible crear, modificar o borrar CAs.

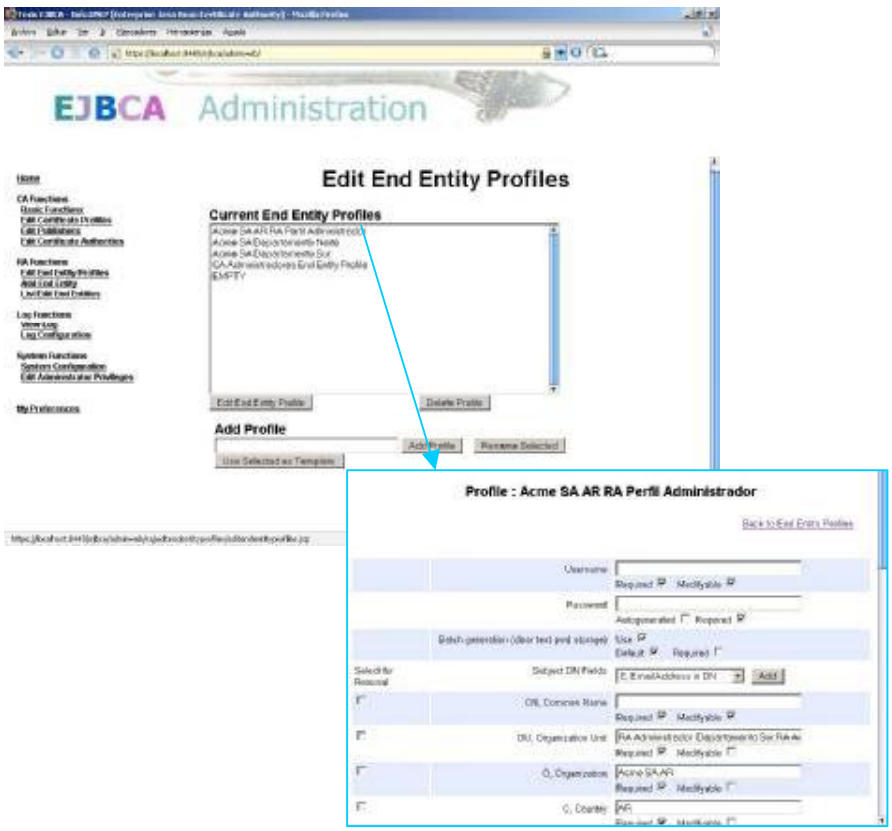


Algunos de los datos a especificar son: Longitud de la clave, Subject DN de la CA, quien firma (auto firmada, CA Externa, otra CA dentro de la PKI que estamos construyendo), lo cual determinará el perfil del certificado de la CA (root o Sub CA), validez en días, Nombre alternativo, Políticas, período de emisión de las CRL en horas, selección del publicador, si utiliza el servicio OCSP, etc. Muchas de estas opciones son dependientes de otras que seleccionamos en pasos anteriores.

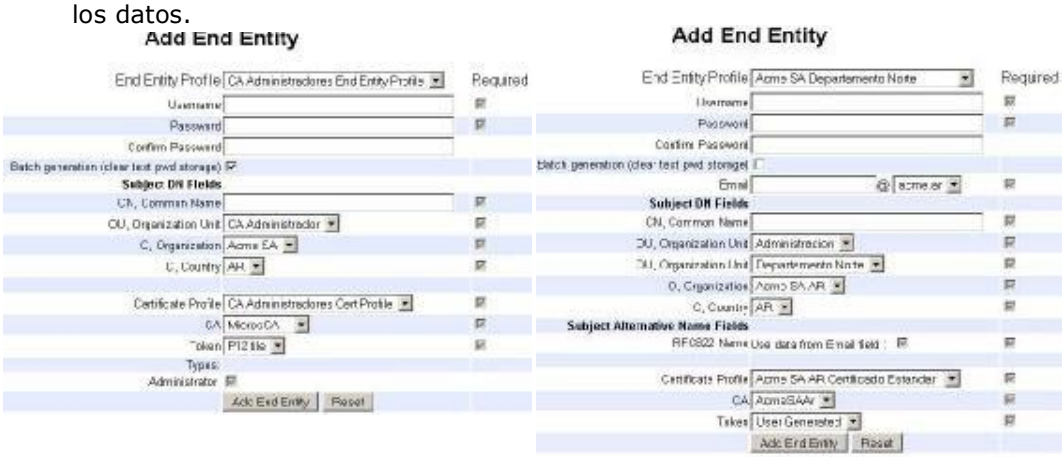
2. RA Functions: (Funciones de RA)

2.1 Edición de Perfiles de Entidades Finales: aquí determinamos que datos serán presentados a los usuarios conectados con cada perfil. Todo usuario que se agrega al PKI debe estar conectado con algún perfil de entidad. Cuando creamos nuevos perfiles, podemos utilizar algunos de los ya existentes como plantilla.

Algunos de los datos significativos de los perfiles son: campos del Subject DN utilizados (CN, OU, C, etc.) cuales de ellos se utilizarán y si serán requeridos y/o modificables, dominio de e-mail, perfil de certificado por default, cuales perfiles de certificado estarán disponibles para este perfil de entidad, CA por default y lista de CAs disponibles, etc.

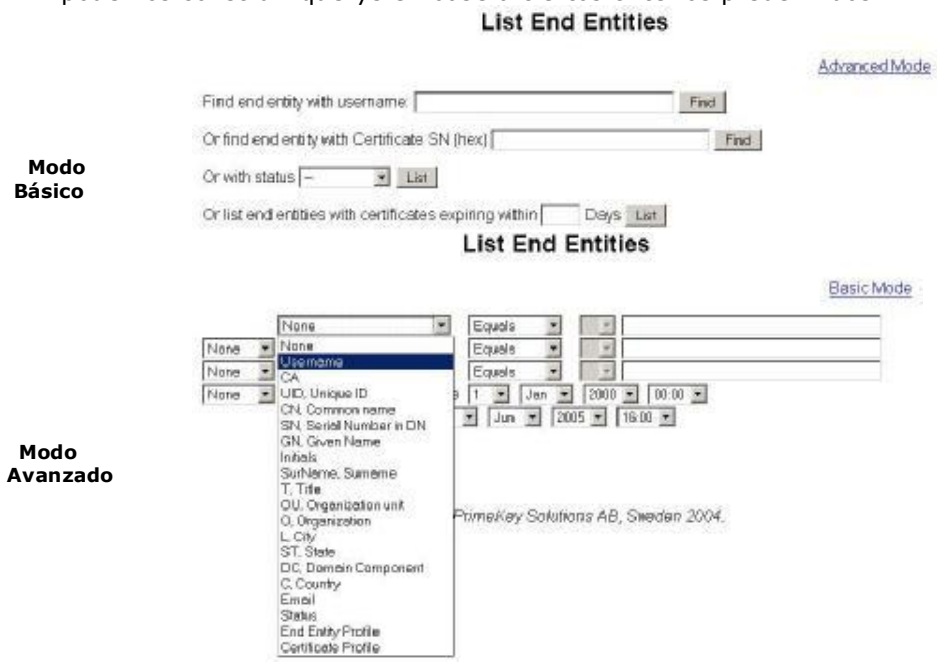


2.2 Agregar Entidad Final: mediante esta opción, agregamos los usuarios participantes en la PKI. Lo primero que deberemos hacer es seleccionar, de la lista de perfiles de entidades creadas en el paso anterior, el perfil de entidad que tendrá este nuevo usuario. Dependiendo del perfil seleccionado, y de las opciones indicadas en la creación de dicho perfil, es que se deberán completar los datos.

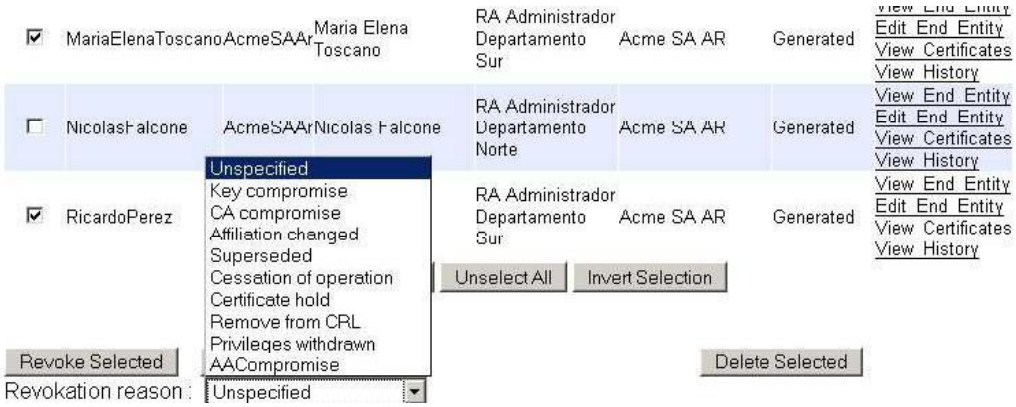


Agregar Entidad Final – Los valores solicitados dependen del Perfil de Entidad

2.3 Listar y Editar Entidad Final: mediante esta opción podemos listar los usuarios deseados, según diversos criterios, por ejemplo: nombre de usuario, Número de Serie del Certificado, por el estado (todos, nuevo, inicializado, generado, revocado, etc.). Existe una modalidad de búsqueda Avanzada, en la cual podemos construir querys en base a ciertos criterios predefinidos.

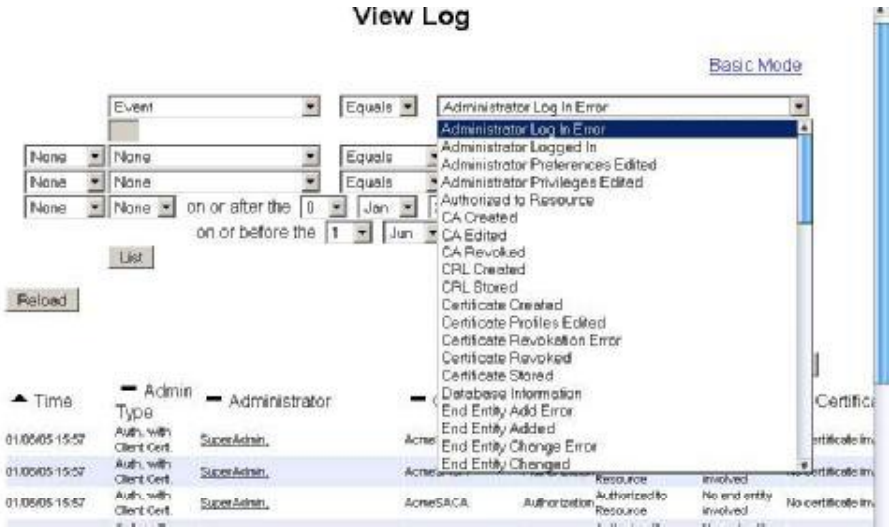


La lista de usuarios que cumplen con el criterio especificado se muestra con una serie de datos tales como: Nombre del Usuario, CA a la que pertenece, Common Name (CN), estado, etc. Disponemos además de una serie de opciones que podemos efectuar con dichos usuarios: ver y/o modificar sus datos, ver sus certificados, ver su historial en los logs. Es posible también revocar sus certificados (especificando si se desea el motivo), e inclusive revocar los certificados y borrar los usuarios.



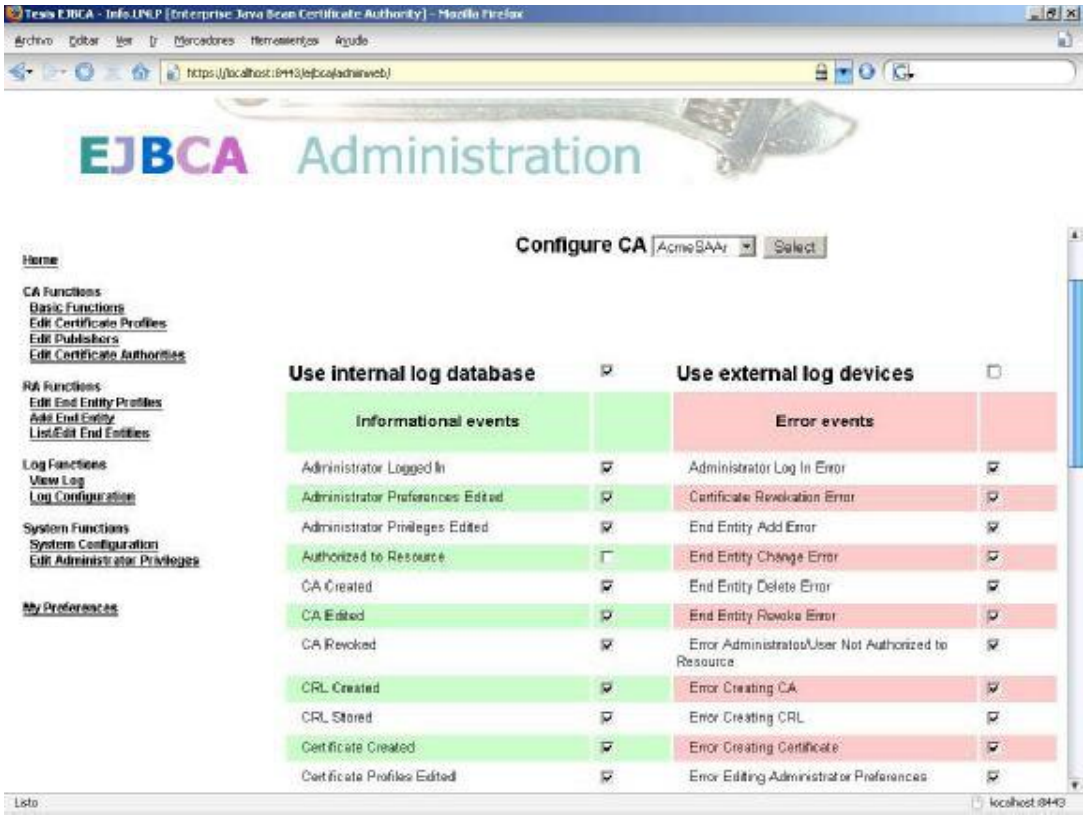
3. Log Functions: (Funciones de Log)

3.1 Ver Log: existen dos modos de buscar datos en el Log. En Modo Básico, se selecciona el intervalo de tiempo a partir de donde se desea consultar (15 minutos, 1 hora, 6 horas, 1 día ó 7 últimos días). En Modo Avanzado, es posible armar querys múltiples en base a criterios predefinidos.



El log muestra una serie de columnas de interés, como ser fecha y hora del suceso, administrador, módulo, evento, nombre de usuario, certificado involucrado, etc. Dependiendo del suceso algunos de las columnas pueden no tener valores.

3.2 Configuración de Log: todos los logs en EJBCA son almacenados por defecto en una tabla interna de la base de datos y pueden también ser enviados a mecanismos de logs externos (Syslog o Eventlog por ejemplo). Los eventos almacenados en la base de datos interna, son visualizados mediante la opción Ver Log, tratada en el punto anterior. Mediante esta opción es posible configurar, para cada CA existente, cuales eventos serán registrados en dichos logs. Por ejemplo: Login del Administrador, Edición de Preferencias del Administrador, Creación de CAs, Edición de CAs, Revocación de CAs, Creación de Certificados, Revocación de Certificados, etc. Esta es una lista predefinida de posibilidades, en las cuales se indica si debe registrarse en el log o no mediante checkboxes. Por default todos los posibles eventos son registrados. Para utilizar mecanismos de log externos se debe escribir una clase java implementando la actual se.anatom.ejbca.log.IlogDevice con 2 métodos. Ambos tipos de logs (en base de datos y externo) pueden ser deshabilitados mediante un check si no deseamos registrar los eventos.



4. System Functions: (Funciones del Sistema)

4.1 Configuración del Sistema: mediante esta opción, podemos efectuar la configuración general de EJBCA. Las principales opciones disponibles son: Nombre de la instalación PKI, banner de cabecera y pié (se puede personalizar la cabecera y pié de la interfase web, creando páginas JSP y luego poner sus nombres en esta opción), Control sobre las RA (es posible habilitar o deshabilitar esta opción; la habilitación implica que los datos de usuarios serán cotejados con los datos de los perfiles de las entidades existentes), habilitación de Key Recovery (recuperación de claves), lenguajes principal y secundario, fuentes y colores, etc.

4.2 Edición de Privilegios de Administradores: esta opción nos permite crear y administrar los distintos grupos de administradores que manejarán nuestra PKI. Desde esta página, podemos crear los administradores de CA, de RA o supervisores por ejemplo. Establecido el nombre del grupo tenemos básicamente dos cosas a configurar:

Personas que integrarán el grupo
Reglas de Acceso del Grupo

Las reglas de acceso las trataremos con mayor detalle al final del capítulo, puesto que son fundamentales para comprender los privilegios de los grupos de administradores.

5. Preferencias: mediante esta opción se configuran las preferencias generales del usuario que está conectado al admin. Web. Es posible configurar el lenguaje preferido y secundario, las fuentes y colores y la cantidad de registros a mostrar por página.
   La configuración de fuentes y colores se efectúa mediante "temas" , contenido en archivos css (hojas de estilo) [Ref. 2].

Reglas de Acceso

Estas fijan los derechos y privilegios que tendrán quienes integren el grupo.

Existen dos formas de configurar las Reglas de Acceso:

Modalidad Básica: es la forma más sencilla de configurar. Simplemente se selecciona el Rol que cumplirá el grupo y éste determina las reglas de acceso, ya que las mismas se encuentran preestablecidas para cada rol. Los posibles Roles son: Administrador de CA, Administrador de RA, Supervisor y SuperAdministrador. Según el rol seleccionado, es posible deshabilitar algunas reglas y/o perfiles de entidades. Finalmente se deberá seleccionar sobre cuales CAs tendrá autoridad este grupo, pudiendo seleccionar una, varias o todas.

Modalidad Avanzada: mediante esta opción cada regla puede ser aceptada o denegada y es posible configurarla con un flag de recursividad. El seteo de las reglas puede verse como una estructura de árbol, similar a un filesystem, donde el flag de recursividad indica que la regla también se aplica para todas las subreglas. A continuación describiremos el significado de cada una de ellas.

Reglas de Acceso – Modalidad Básica

Reglas de Acceso – Modalidad Avanzada

Reglas de Acceso – Modalidad Avanzada

Descripción de las Reglas de Acceso

Reglas de Acceso Basadas en Roles

Regla	Recurso
/public_web_user	Acceso a web público
/administrator	Acceso a páginas de administración
/super_administrator	Acceso a todo; puede editar CAs y Publicadores

Reglas de Acceso Regulares

Regla	Recurso
/ca_functionality	No se usa
/ca_functionality/basic_functions	Acceso a la página de funciones básicas
/ca_functionality/create_crl	Posibilidad de crear CRL
/ca_functionality/edit_certificate_profiles	Acceso a Pág. para editar perfiles de certificados
/ca_functionality/create_certificate	Usuario autorizado a crear certificados
/ca_functionality/store_certificate	Usuario autorizado a almacenar certificados
/ca_functionality/view_certificate	Habilitado a ver certificados

/ra_functionality	No se usa
/ra_functionality/edit_end_entity_profiles	Acceso a Pág. de edición perfiles de entidades finales
/ra_functionality/create_end_entity	Acceso a Pág. para agregar entidades finales
/ra_functionality/delete_end_entity	Aparece botón de borrar en lista de entidades finales
/ra_functionality/edit_end_entity	Acceso a Pág. de edición de entidades finales
/ra_functionality/revoke_end_entity	Aparece botón de Revocar en lista de entidades finales
/ra_functionality/view_end_entity	Acceso a Pág. de vista de entidades finales
/ra_functionality/view_end_entity_history	Acceso a Pág. de ver historial
/ra_functionality/keyrecovery	Acceso a funciones de Keyrecovery
/log_functionality	No se usa
/log_functionality/edit_log_configuration	Acceso a Pág. de configuración de log
/log_functionality/view_log	Acceso a Pág. de ver log
/log_functionality/view_log/adminweb_entries	Posibilidad de visualizar eventos enviados desde módulo admin. Web
/log_functionality/view_log/ca_entries	Posibilidad de ver eventos enviados desde módulo de ca
/log_functionality/view_log/log_entries	Posibilidad de ver eventos enviados desde módulo de log
/log_functionality/view_log/publicweb_entries	Posibilidad de ver eventos enviados desde módulo de web pública
/log_functionality/view_log/ra_entries	Posibilidad de ver eventos enviados desde módulo ra
/log_functionality/view_log/hardtoken_entries	Posibilidad de ver eventos enviados desde módulo de hardtoken
/log_functionality/view_log/keyrecovery_entries	Posibilidad de ver eventos enviados desde módulo de keyrecovery
/log_functionality/view_log/authorization_entries	Posibilidad de ver eventos enviados desde módulo de autorización
/hardtoken_functionality	No se usa
/hardtoken_functionality/issue_hardtokens	Administrador tiene derechos de emitir hard tokens
/hardtoken_functionality/issue_hardtokens_administrator	El Administrador tiene derecho de emitir hard tokens con privilegions de administrador. (Es de versiones anteriores. No se utiliza más)
/hardtoken_functionality/edit_hardtoken_issuers	Acceso a Pág. para editar Emisores de Hard Token.
/hardtoken_functionality/edit_hardtoken_profiles	Acceso a Pág. para editar perfiles de Hard Token.
/system_functionality	No se usa
/system_functionality/edit_administrator_privileges	Acceso a Pág. de Privilegios de Administrador

Reglas de Acceso a Perfiles de Entidades Finales

Regla	Recurso
/endentityprofilerules	Ninguno
/endentityprofilerules/< End Entity Profile Name>/create_end_entity	Administrador tiene derecho de crear usuarios con ese perfil

/endentityprofilerules/< End Entity Profile Name>/delete_end_entity	Administrador tiene derechos de remover usuarios con ese perfil
/endentityprofilerules/< End Entity Profile Name>/edit_end_entity	Administrador tiene derechos de editar usuarios con ese perfil
/endentityprofilerules/< End Entity Profile Name>/revoke_end_entity	Administrador tiene derechos de revocar usuarios con ese perfil
/endentityprofilerules/< End Entity Profile Name>/view_end_entity	Administrador tiene derechos de ver usuarios con ese perfil
/endentityprofilerules/< End Entity Profile Name>/view_end_entity_history	Administrador tiene derechos de ver historial de usuarios con ese perfil

Reglas de Acceso CA

Regla	Recurso
/ca	No se usa
/ca/< CA Name>	El administrador tiene derechos de administrar esa CA

Esta página fue dejada en blanco de manera intencional.

Capítulo 11
Administración mediante Línea de Comandos

Descripción General

Esta interfase consiste básicamente en dos comandos: *ca* y *ra.* Dependiendo de la plataforma sobre la cual esté corriendo (Windows o Linux) será *ca.cmd* ó *ca.sh*, siendo lo mismo para el caso de *ra*. La sintaxis por línea de comando es:

<comando ca o comando ra> <subcomando> <lista de parámetros>

Escribiendo solamente el comando (*ca* ó *ra*) se obtiene una lista de los subcomandos posibles. Ingresando el nombre del comando y del subcomando se dará información sobre argumentos y uso general del mismo.
Por ejemplo: ingresando *ca getrootcert* se obtiene

Save root CA certificate <PEM- or DER-format> to file
Usage: CA getrootcert <caname> <filename> <-der>

A continuación describiremos la lista de posibles comandos.

Comandos para Administración de CA

Nombre del Comando: **CA**

- *init*: Crea una nueva CA Raíz. Almacena certificados de la CA y publica la primer CRL.

- *makeroot*: crea un nuevo keystore de CA Raíz (solo se utiliza para CA Raíz). No se utiliza más a partir de la versión 3 de EJBCA. El DN entre comillas (") se trata como un solo argumento.

- *getrootcert*: exporta en certificado de la CA a un archivo.

- *makereq*: genera un requerimiento de certificado para una subCA que es enviado a una CA Raíz. Se utiliza para crear un keystore para una CA subordinada. El DN entre comillas se trata como un solo argumento. No se utiliza más a partir de la versión 3 de EJBCA.

- *recrep*: utilizado para recibir certificados que se produjeron como resultado de enviar un requerimiento a una CA Raíz. No se utiliza más en EJBCA Ver.3.

- *processreq*: procesa un requerimiento de certificado para una CA subordinada y crea una respuesta de certificación. No se utiliza más en EJBCA Ver.3.

- *createcrl*: emite una CRL.
- *getcrl*: recupera la última CRL.

- *rolloverroot*: se utiliza para generar un nuevo certificado de la CA Raíz a partir de un par de claves existentes.(Solamente usado para CA Raíz). Esto actualiza

el keystore actual de la CA Raíz. Los sub-certificados no necesitan ser refirmados. No se utiliza más en EJBCA Ver.3.

- *rolloversub*: usado para generar un nuevo certificado de SubCA usando un par del claves existente. Actualiza el keystore actual de la subCA y el sub-certificado no necesita ser refirmado. No se utiliza más en EJBCA Ver. 3.

- *listexpired*: lista los certificados que expirarán dentro de una cierta cantidad de días.

- *exportprofiles*: exporta perfiles de certificados y entidades a archivos xml.

- *importprofiles*: importa perfiles de certificados y entidades desde archivos xml. Cuando se exportan perfiles para ser importados en otra CA, debe uno asegurarse que 'CAs Disponibles' esté configurado como 'Cualquier CA' para los perfiles que se están exportando. De no ser así, puede haber problemas de autorización si uno no tiene las mismas CAs con los mismos caids (DNs) donde se importarán los perfiles.

- *importca*: crea una nueva CA, pero importando claves desde un archivo PKCS12 existente. Un archivo PKCS12 puede generarse desde archivos PEM con openssl.

Comandos para Administración de RA

Nombre del Comando: **RA**

- *adduser*: agrega un usuario a la base de datos; una vez agregado puede solicitar un certificado. Si la dirección de email es nula, entonces no se pone email en el certificado. Las comillas en el DN hacen que se trate como un solo argumento. Altnames es un string similar al DN, pero utilizando nombres alternativos de RFC3280. Por ejemplo:

  "rfc822Name=<email>,dNSName=<nombrehost>,uri=http//<cualquiera>,iPAddress=10.56.32.24"

  La lista completa soportada hasta este momento es:
  otherName, rfc822Name, dNSName, x400Address, directoryName, ediPartyName, uniformResourceIdentifier, iPAddress, registeredID Only rfc822Name, dNSName, iPAddress y uniformResourceIdentifier (uri). También son soportados el MS UPN [Ref. 1] y GUID [Ref. 2].

- *deluser*: elimina un usuario de la base de datos; cualquier certificado emitido permanece activo y presente en la base de datos.

- *setpwd*: establece una nueva password para un usuario. La password se almacena como un hash en la base de datos.
- *serclearpwd*: establece una password para un usuario en texto claro; necesaria para generar un certificados de forma batch.

- *setuserstatus*: establece el estado de un usuario; los usuarios solo pueden requerir certificados cuando su estado es 'NUEVO'.

- *finduser*: busca un usuario en la base de datos y lista sus detalles.

- *listnewusers*: lista todos los usuarios con estado NUEVO.
- *listusers*: lista los usuarios con un determinado estado (escribiendo el comando solo, se muestra la lista de códigos de estado posibles). Los posibles estados son: 10= NUEVO; 11= FALLIDO; 20= INICIALIZADO; 30= EN PROCESO; 40= GENERADO; 50= HISTORICO
- *revokeuser*: revoca un usuario y todos los certificados que se le entregaron.
- *keyrecover*: recupera las claves relacionadas con un certificado específico.
- *keyrecovernewest*: recupera las últimas claves de un usuario.

Esta página fue dejada en blanco de manera intencional.

Capítulo 12
Dependencias

La mayoría de las dependencias se distribuyen en el mismo paquete de instalación de EJBCA. Lo mínimo a descargar e instalar son *JBoss* como servidor de aplicaciones y *Ant* para la construcción.

Bouncycastle

EJBCA utiliza para criptografía y creación de certificados el open source Bouncy Castle Crypto package, provistas por el grupo *Legion of the Bouncy Castle*. [Ref.1]

Este paquete es una implementación en Java de algoritmos criptográficos; está organizado para contener un API liviano conveniente para su utilización en cualquier ambiente, con la infraestructura adicional conforme a los algoritmos del framework JCE.

Estas cripto APIs consisten de lo siguiente:

- Un API criptográfico liviano en Java, con soporte para:
    - BlockCipher
    - BufferedBlockCipher
    - AsymmetricBlockCipher
    - BufferedAsymmetricBlockCipher
    - StreamCipher
    - BufferedStreamCipher
    - KeyAgreement
    - IESCipher
    - Digest
    - Mac
    - PBE
- Un proveedor para JCE y JCA
- Una implementación limpia del JCE 1.2.1
- Una librería para leer y escribir objectos codificados ASN.1
- Generador de certificados para X.509 Versiones 1 y 3, CRLs Versión 2 y archivos PKCS12.
- Generador para atributos de certificados X.509 Versión2
- Generadores y Procesadores para S/MIME y CMS (PKCS7)
- Generadores y Procesadores para OCSP (RFC 2560)
- Generadores y Procesadores para TSP (RFC 3161)
- Generadores y Procesadores para OpenPGP (RFC 2440)
- Una versión jar firmada conveniente para JDK 1.4/1.5 y el JCE Sun.

Esta API ligera trabaja con todo desde el J2ME hasta el JDK 1.5.

JBoss

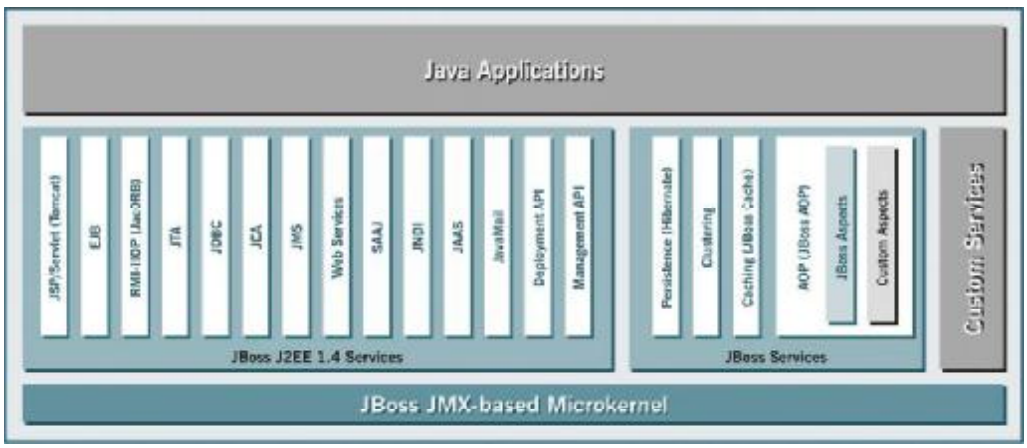
EJBCA está desarrollado sobre el servidor de aplicaciones open source JBoss [Ref.2]. Lo utiliza principalmente como servlet engine (web container).

Durante el transcurso del año 2004, JBoss obtuvo la certificación de Sun habiendo superado los tests de compatibilidad para J2EE 1.4. Se convierte así en el primer producto libre (licencia LGPL) en obtener este tipo de certificación.

Arquitectura de JBoss

La arquitectura del servidor JBoss se puede dividir en cuatro capas principales:

- Capa del Microkernel: su corazón es un servidor basado en el microkernel. Utilizando las Java Management Extensions (JMX), el microkernel un modelo de componentes ligero que ofrece una administración completa del ciclo de vida, y avanzadas características de carga de clases.
- Capa de Servicios: arriba del microkernel reside una Arquitectura Orientada a Servicios (SOA, Services Oriented Architecture), consistente en una serie de servicios empaquetados. Los servicios van en un amplio rango, desde servicios de transacciones y mensajería a servicios de correo, seguridad o administración de conexiones. Es posible añadir nuevos servicios o remover los innecesarios para reducir el tamaño y mejorar la performance. Puede construir servicios propios y extenderlos como SARs dentro del servidor JBoss. Cada servicio se empaqueta como un Archivo de Servicio o SAR, y cada SAR son *hot-deployables*, haciendo muy fácil extender JBoss.
- Capa de Aspecto: esta capa está basada sobre el modelo Aspect-Oriented Programming (AOP). Una larga tradición en JBoss es el concepto de *interceptores*. Los *interceptores* permiten al sistema agregar de manera transparente la conducta provista por los servicios a cualquier objeto. El JBoss EJB container por ejemplo, se construye como un conjunto preempaquetado de interceptores, y uno puede fácilmente agregar o quitar interceptores en base a las necesidades propias.



Arquitectura JBoss: Microkernels, Servicios y Aspectos

- Capa de Aplicación: aquí es donde viven las aplicaciones propias. Estas aplicaciones pueden aprovechar las capacidades de JBoss, utilizando los contenedores de servicios en forma directa o bien utilizando la capa AOP y agregar conductas a los objetos propios.

Ant

Es una herramienta open source de la *Apache Software Foundation* – es parte del proyecto Apache Jakarta - , utilizada para la construcción de programas escritos en Java [Ref.3]. A diferencia de otras herramientas de este tipo, Ant es independiente tanto del ambiente como de la plataforma de desarrollo.

Make, gnumake, jam y muchas otras alternativas, están basadas sobre un shell particular o interfase de comandos, por lo que está limitado a los sistemas operativos que utilizan ese shell. Ant utiliza clases Java en lugar de los comandos basados en shell; está escrito en XML y Java, lo que permite ofrecer una solución interoperable a nivel del sistema operativo (debido a Java) y configuraciones descriptivas (debido a XML). Los desarrolladores utilizan XML para describir los módulos en los programas que construyen, que deben hacer esos módulos y cualquier dependencia entre ellos y otras partes del programa. Ant determina que partes del programa han sido modificadas desde la última compilación, como también cualquier parte del programa que depende de esos componentes, compilando entonces solo esas partes requeridas en el orden apropiado.

Un IDE [Ref. 4] ofrece interfases gráficas (GUI) para llevar a cabo compilaciones, debugers elaborados, generación de código automático y otras funcionalidades (algunos IDES que podríamos mencionar son: NetBeans, Eclipse, Jbuilder, Jdeveloper, etc). Aunque Ant es herramienta de compilación y construcción muy flexible, que permite acelerar el tiempo de desarrollo, no es un IDE; inclusive en muchas ocasiones es utilizada en conjunto con un IDE. En el caso de EJBCA, se utilizó NetBeans (es open source).

LOG4J

Es un API para manejar el registro (log) de operaciones en los programas, y es utilizado para depuración [Ref.5]. Log4j es uno de los componentes del proyecto Apache Jakarta.

El API ofrece cinco prioridades diferentes, de mayor a menor a saber: DEBUG, INFO, WARN, ERROR, FATAL; de ser necesario es posible crear prioridades propias.

De igual forma que la estructuración en paquetes de Java, Log4j divide los tipos de registro por categorías, cuya prioridad es hereditaria. Por ejemplo, si no se define una prioridad para la categoría "se.anatom.ejbca.admin", heredaría la prioridad de "se.anatom.ejbca". Así, una vez definida una categoría, podemos enviar peticiones de registro (log requests) utilizando los métodos que ésta ofrece.

Log4j proporciona *Layouts*, que permite determinar la información que acompaña a los mensajes. Un ejemplo es el SimpleLayout, que muestra simplemente la prioridad y el mensaje, o un poco más elaborado, PatterLayout que permite formatear libremente la información a mostrar.

Existen tres clases que nos permiten configurar el registro: BasicConfigurator, PropertyConfigurator y DOMConfigurator. La primera utiliza un Layout predefinido para mostrar el registro por pantalla; con las dos restantes es posible utilizar un archivo de configuración, normal para PropertyConfigurator y uno XML parseado para la última.

Por último, para determinar que debe hacer Log4j con la información de registro, existen los *Appenders*. Por ejemplo, es posible escribir en un OutputStream con WriteAppender, en archivos con FileAppender, en archivos de registro de un tamaño máximo con RollingFileAppender; también es posible enviar por correo la información con SMTPAppender, o registrarla en un servidor especial para Log4j (cuyas clases se incluyen) con SocketAppender, etc. Adicionalmente es posible enviar a varios sitios a la vez, con lo cual las posibilidades que brinda son muy grandes.

JUnit

Es un framework Java open source, utilizado para escribir y correr test repetibles, para automatizar las pruebas de clases Java [Ref.6]. JUnit fue escrito originalmente por Erich Gamma y Kent Beck.

La forma típica de funcionamiento de un test con JUnit es: definir unos datos, darles algún tipo de tratamiento, obtener unos resultados y compararlos con los resultados que se esperaban. La idea principal es que sea el propio JUnit el que indique si el test ha sido exitoso o si el resultado no coincide con lo esperado.

JUnit incluye las siguientes características:

- Afirmaciones respecto de los resultados esperados
- Fixtures de test para compartir datos comunes de pruebas
- Suites de test para organizar y correr test fácilmente
- Corridas de pruebas gráficas y de texto

JUnit se ofrece bajo la Licencia pública de IBM [Ref.7].

HttpJUnit

Es una extensión de Junit, utilizada para testear páginas web y forms.

OpenLDAP

OpenLDAP es un servidor LDAP que se distribuye bajo licencia GNU (Open Source). Este archivo jar es utilizado por EJBCA para almacenar certificados y CRLs en directorios LDAP. El jar se distribuye con EJBCA.

El proyecto OpenLDAP [Ref.8] nació como la continuación de la versión 3.3 del servidor LDAP de la Universidad de Michigan, cuando éstos dejaron de desarrollarlo.

Las versiones actuales de OpenLDAP, funcionan con la Versión 3 de LDAP (RFC 3377); LDAPv3 es el estándar actual para todos los servidores LDAP.

Los paquetes que incluyen las distribuciones de OpenLDAP son:

- slapd: (Servidor LDAP), es un servidor para recibir las peticiones LDAP, servirlas, gestionarlas, etc. Es el núcleo del proyecto y lo que implementa las funciones básicas de LDAP.
- slurpd: (Servidor de replicación LDAP), un daemon que sirve para replicar un directorio LDAP. Por el momento solo permite la replicación de un árbol entero, sin replicar ninguna de sus partes.
- librerías: (Software Development Kit) provee una serie de librerías (en C) y utilidades para el acceso al directorio, que también forma parte de su proyecto.
- utilidades, herramientas y ejemplos de clientes.

Esta página fue dejada en blanco de manera intencional.

Capítulo 13
APIs de EJBCA

Introducción

EJBCA dispone de diversos paquetes Java que pueden ser también utilizados para operarla de manera integrada a una aplicación J2EE existente en la empresa. [Ref.1]

Información detallada y completa de cada uno de ellos, con sus clases, interfases y excepciones se obtiene corriendo el script:

ant javadoc

el cual genera páginas web que permiten navegar a través de cada uno de estos paquetes, pudiendo visualizar las clases e interfases y buscar dentro de ellas los métodos, constructores, campos, etc. como está construido y la documentación necesaria para su utilización.

En este capítulo, haremos una breve reseña de estos paquetes, describiendo algunas de las funcionalidades implementadas en ellos.

Documentación

Paquete	Descripción
se.anatom.ejbca	Contiene las clases bases e interfases para los beans de entidad que implementan métodos y ayudas requeridos. Algunas de las clases que contiene son BaseEntityBean, BasePropertyEntityBean, BaseSessionBean, PropertyEntityPk
se.anatom.ejbca.admin	Este paquete contiene las clases e interfases que implementan la administración de EJBCA. Algunas de las clases son: BaseCaAdminCommand (para los comandos de CA, contiene las funciones comunes para las operaciones de la CA), Install (utilizado para el script de instalación de EJBCA), ca (implementa la interfase por línea de comandos de la CA).
se.anatom.ejbca.apply	Contiene clases y servlets para crear usuarios, recuperar certificados, instalar clave privada de un certificado en el browser, etc.
se.anatom.ejbca.appserver.jboss	Contiene clases y una interfase que administra la creación de CRL de manera automática.
se.anatom.ejbca.authorization	Clases e interfases para el manejo de autorización, grupos de usuarios y reglas de acceso.
se.anatom.ejbca.batch	Contiene solo una clase (BatcjMakeP12) que genera claves y requerimiento de certificados para todos los usuarios con estado NEW.
se.anatom.ejbca.batch.junit	Contiene clases para testeo de proceso batch

se.anatom.ejbca.ca.auth	Contiene clases e interfases para autenticar usuarios hacia la base de datos.
se.anatom.ejbca.auth.junit	Clases para testeo de autenticación.
se.anatom.ejbca.ca.crl	Contiene clases e interfases para la creación de CRL e información de certificados revocados.
se.anatom.ejbca.ca.crl.junit	Para testeo de sesiones CRL.
se.anatom.ejbca.ca.caadmin	Contiene clases e interfases para la administración y manejo de CAs en EJBCA y generación de certificados y CRLs de las CAs en base al estándar X509.
se.anatom.ejbca.ca.caadmin.extendedcaservices	Contiene las clases que implementan servicios extendidos de las Cas (OCSP, Recuperación de Claves, etc.)
se.anatom.ejbca.ca.caadmin.hardcatokens	Clases que manejan los Hard Tokens de CAs.
se.anatom.ejbca.ca.caadmin.junit	Clases para testeo del entity bean ca.
se.anatom.ejbca.ca.exception	Para el manejo de excepciones (errores de login, de generación de certificados, de publicadores, etc.)
se.anatom.ejbca.ca.publisher	Contiene clases e interfases para implementar publicadores personalizados.
se.anatom.ejbca.ca.publisher.junit	Clases para testeo de publicadores.
se.anatom.ejbca.ca.sign	Contiene las clases e interfases para la creación y firma de certificados.
se.anatom.ejbca.sign.junit	Para testeo de generación de números de serie de certificados.
se.anatom.ejbca.ca.store	Contiene clases e interfases para implementar el almacenamiento de certificados y CRL en la base de datos local.
se.anatom.ejbca.ca.store.certificateprofiles	Contiene las clases para almacenar los perfiles de certificados (de las CA, Usuarios Finales, Hard Token, etc.)
se.anatom.ejbca.ca.store.junit	Contiene las clases para el testeo de almacenamiento de certificados.
se.anatom.ejbca.exception	Clase base para todas las excepciones específicas de la aplicación arrojadas por EJBCA.
se.anatom.ejbca.hardtoken	Contiene las clases, interfases y excapciones para el manejo de dispositivos de Hard Token.
se.anatom.ejbca.hardtoken.hardtokenprofiles	Contiene las clases e interfases que implementan es manejo de perfiles de hard token en el sistema.
se.anatom.ejbca.hardtoken.hardtonetypes	Contiene clases para majejar los distintos tipos de hard token (EID Token, Swedish EID Token)
se.anatom.ejbca.hardtoken	Contiene clases para testear el manejo de hard token.
se.anatom.ejbca.keyrecovery	Contiene las clases e interfases necesarias para implementar la recuperación de claves.
se.anatom.ejbca.keyrecovery.junit	Contiene las clases para testear los módulos de recuperación de claves.
se.anatom.ejbca.log	Este paquete contiene las clases e interfases para almacenar logs en la base de datos interna, implementar dispositivos de log utilizando Log4j, configuración de logs, etc.
se.anatom.ejbca.log.junit	Implementa los testeo de los módulos de log.

se.anatom.ejbca.protocol	Contiene interfases y clases para el manejo de mensajes en EJBCA. Existen clases para manejo de mensajes de OCSP, respuestas de estado de SCEP, requerimientos de PKCS7 y PKCS10, X509, etc.
se.anatom.ejbca.protocol.exception	Clases para el tratamiento de excepciones en el manejo de mensajes (errores en la forma de requerimiento ó tipos de requerimientos no spoportados).
se.anatom.ejbca.protocol.junit	Contiene las clases que implementan el testeo de páginas http para ocsp y scep.
se.anatom.ejbca.ra	Contiene las interfases y clases para las RA.
Se.anatom.ejbca.ra.exception	Contiene solo la excepción NotFoundException, que se produce cuando un objeto no se encuentra en la base de datos y el error no es crítico y se desea informar al usuario de una manera elegante.
se.anatom.ejbca.ra.junit	Contiene clases que implementan el testeo del bean UserData y algunas partes del bean UserAdminSession.
se.anatom.ejbca.ra.raadmin	Contiene clases, interfases y excepciones para la administración de RAs.(preferencias personales del administrador, recuperación de diferentes campos de un DN, parámetros de configuración global, errores que se producen cuando se intenta agregar una entidad final o un perfil que ya existe, etc.)
se.anatom.ejbca.ra.raadmin.junit	Contiene clases para el testeo de los beans de preferencias del administrador, perfiles de entidades finales, y configuración global.
se.anatom.ejbca.samples	Contiene una clase con el resultado completo de un pedido de autenticación de usuario, un servlet para autenticar un usuario y una clase para traer un certificado mediante http.
se.anatom.ejbca.upgrade	Contiene una clase e interfases para la actualización de la base de datos cuando se producen nuevos releases de ejbca.
se.anatom.ejbca.util	Contiene clases para implementación de herramientas utilizadas en ejbca. (manejo de operaciones comunes de claves y certificados, para exportar cert. De formato P12 a PEM, manejo de strings, ejecución de sentencias sql, etc.)
se.anatom.ejbca.util.junit	Contiene algunas clases para el testeo de clases del paquete anterior.
se.anatom.ejbca.util.passgen	Contiene interfases y clases para la implementación del generador de passwords de manera aleatoria y en base a los distintos tipos que pueden existir (solo dígitos, letras y dígitos, etc.)
se.anatom.ejbca.util.query	Contiene clases y excepciones para la construcción de los distintos querys en ejbca (búsquedas en log, tablas de datos de usuarios, etc.). También contiene un manejador de excepciones para cuando el query es illegal (incorrecta la sintaxis).

se.anatom.ejbca.webdist	Contiene un servlet que implementa la distribución de certificados y CRL.
se.anatom.ejbca.webdist.cainterface	Contiene las clases que implementan la distribución de certificados y CRL mediante la herramienta administrativa web.
se.anatom.ejbca.webdist.hardtokeninterface	Contiene clases que implementan las vistas, perfiles de hard token, autorizaciones de los adiministradores para ver o editar hard tokens, etc.
se.anatom.ejbca.webdist.junit	Contiene una clase que implementa un test de página http para webdist público.
se.anatom.ejbca.webdist.loginterface	Contiene clases que implementan el módulo de log, que partes está autorizado a ver el administrador, etc.
se.anatom.ejbca.webdist.rainterface	Contiene clases para las vistas de usuarios de la ra existentes en la base de datos por medio de la interfase web, java beans para el manejo de la interfase entre el módulo ra de EJBCA y las páginas JSP, etc.
se.anatom.ejbca.webdist.webconfiguration	Contiene clases y excepciones para el manejo de la configuración mediante la interfase web. Funciones básicas de la interfase web, lenguajes, almacenamiento de preferencias del administrador, etc. También maneja lo errores como consecuencia de tratar de agregar un usuario a la base de datos que ya existe o borrar o modificar uno que no existe.

Capítulo 14
Construyendo una PKI paso a paso con EJBCA

Una vez aprendidos los conceptos generales de la infraestructura PKI y habiendo estudiado la funcionalidad de EJBCA, procedimos a la configuración del mismo para crear una jerarquía medianamente compleja de PKI que utiliza EJBCA.

La PKI que construimos consiste en:

Dos empresas distintas, Acme S.A. y MICROS, cada una con su propia root CA.

MICROS, con sede en Argentina, es una PKI con una CA y una RA solamente.

Acme S.A. tiene sus sedes y negocios en Argentina y Uruguay, requiriendo por lo tanto una sub CA en cada país.

Acme S.A. también necesita contar con tres RA distintas por cuestiones administrativas.

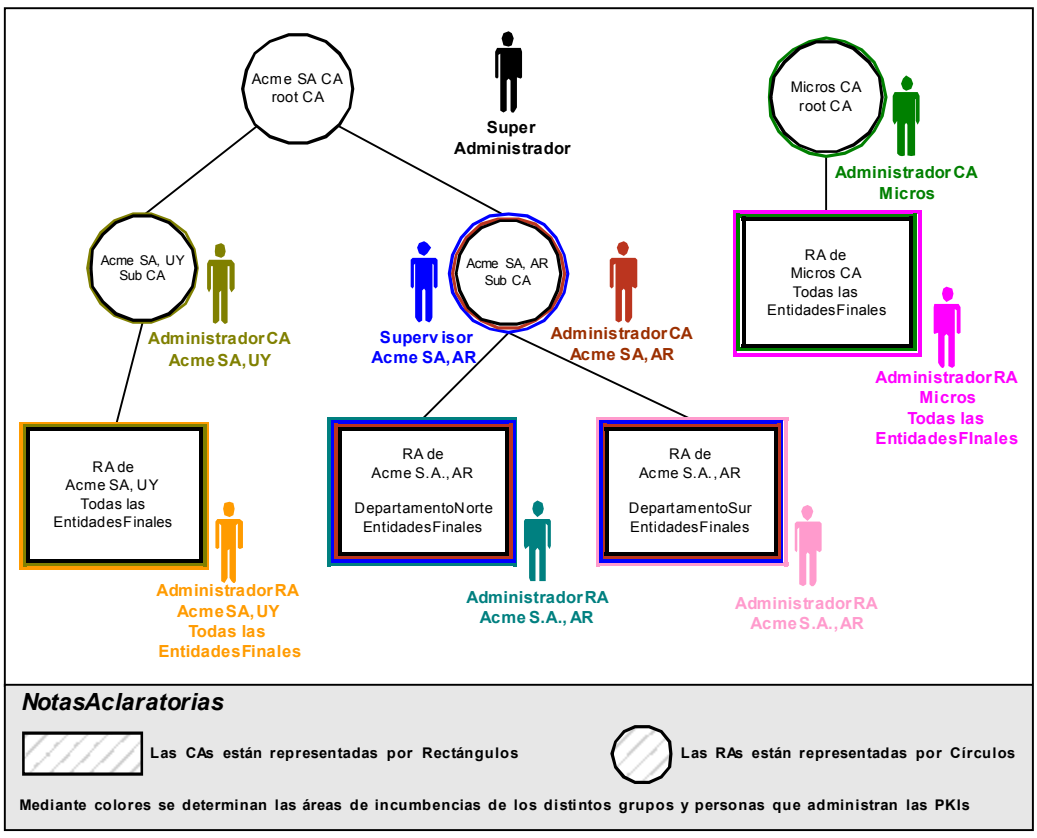


Diagrama de la PKI construida

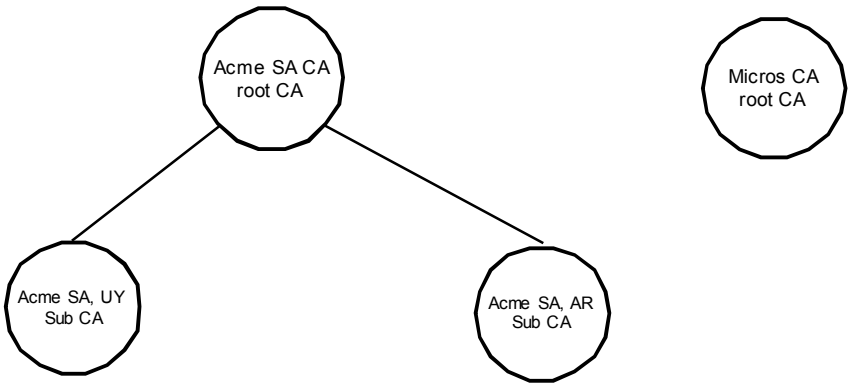
1. Creación de Publicadores:

   En Primero lugar hemos creado los publicadores de certificados, suponiendo tener dos tipos:

   a. Acme AD: de tipo Active Directory, conteniendo los certificados de Acme SA. DN: CN=Users,DC=acme,C=AR

   b. Micros LDAP: de tipo LDAP, que contendrá los certificados de Micros. Definimos su DN: O=Micros,C=AR

2. Luego construimos la jerarquía de CA necesarias:



Para todas las CAs hemos definido una clave RSA de longitud 2.048 bits y 10 años de validez.

Acme SA
- Nombre: AcmeSACA
- Tipo: root CA
- DN: "CN=Acme SA,O=Acme SA"
- Publisher: Acme AD

Acme SA Argentina
- Nombre: AcmeSAAR
- Tipo: SubCA de AcmeSACA
- DN: "CN=Acme SA ARG CA,O=Acme SA,C=AR"
- Publisher: Acme AD

Acme SA Uruguay
- Nombre: AcmeSAUY
- Tipo: SubCA de AcmeSACA
- DN: "CN=Acme SA UY CA,O=Acme SA,C=UY"
- Publisher: Acme AD

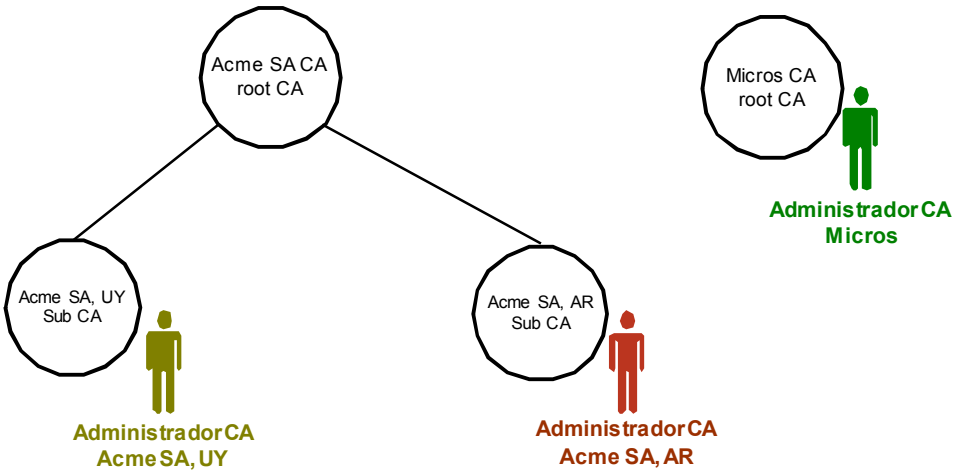
Micros
- Nombre: MicrosCA
- Tipo: root CA

DN: "CN=Micros,O=Micros,C=AR"
Publisher: Micros LDAP

3. Definición de los privilegios de los Administradores: creamos los grupos de administradores necesarios, considerando tener administradores en tres lugares:

   Acme en Argentina
   Acme en Uruguay
   Micros en Argentina

   Aquí es importante tener en cuenta que los administradores no puedan ver datos de los otros sitios tal como usuarios, logs, etc.



Nombre Grupo: CA Administradores Acme AR
Nombre Administrador: Alejandro Falcone
Reglas de Acceso: Rol: Administrador de CA
CAs Autorizadas: AcmeSAAR

Nombre Grupo: CA Administradores Acme UY
Nombre Administrador: Patricia Clemens
Reglas de Acceso: Rol: Administrador de CA
CAs Autorizadas: AcmeSAUY

Nombre Grupo: CA Administrador Micros
Nombre Administrador: Jorge González
Reglas de Acceso: Rol: Administrador de CA
CAs Autorizadas: Micros

4. El paso siguiente es crear los key-stores para los nuevos administradores, pero en primer lugar debemos definir el contenido de los certificados que estamos por emitir. Para ello creamos un Perfil de Certificado que utilizará el superadministrador para emitir a los Administradores de CA.

Nombre del Perfil de Certificado: CA Administrators Cert Profile
Validez: 1.825 días (5 años)
CAs disponibles: SubCA Argentina Acme, SubCA Uruguay Acme, RootCA Micros
Longitud: Todas las posibles (512,1.024,2.048,4.096 bits)
Publicadores: Como no necesitamos publicar estos certificados, no seleccionamos ninguno.
Tipo de Certificado: Entidad Final

5. EL siguiente paso es Agregar los Perfiles de Entidades Finales, en el cual definiremos los campos usados en el DN.

   Nombre del Perfil: CA Administrators End Entity Profiles
   Subject DN Fields:

CN= lo dejamos en blanco.	Requerido - Modificable
OU= CA Administrator.	Requerido – No Modificable
O= Acme SA, Micros.	Requerido – No Modificable
C= AR;UY	Requerido – No modificable

   Perfil de Certificado por Defecto: CA Administrators Cert. Profile
   Perfiles de Certificado Disponibles: CA Administrators Cert. Profile
   CAs Disponibles: Indicamos las 3 (MicrosCA, AcmeSAAR, AcmeSAUY)
   Token Disponibles: P12 file
   Flag Administrador: habilitado. Este flag debe estar habilitado para que el administrador pueda logearse, además de lo que indicamos en los privilegios administrativos.

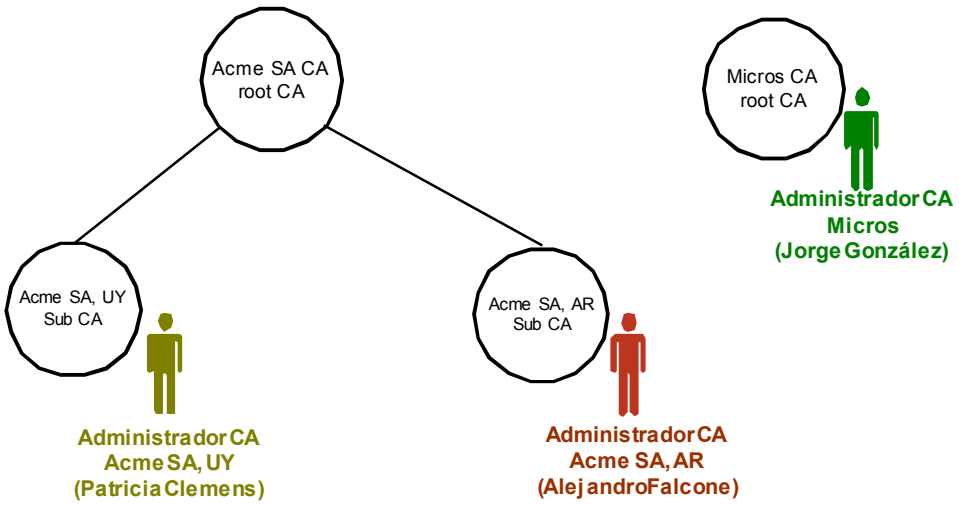
6. El próximo paso será agregar los administradores de CA al sistema, esto es agregar las Entidades Finales, con el perfil de entidad recientemente creado. Las entidades finales que agregaremos serán:

   - Alejandro Falcone (Administrador de Acme SA AR),
     Perfil de Entidad: CA Administrators End Entity Profiles
     Nombre de Usuario: AlejandroFalcone
     CN: Alejandro Falcone
     OU: CA Administrator
     O: Acme SA
     C: AR
     Perfil Certificado: CA Administrators Cert Profile
     CA: AcmeSAAR

   - Patricia Clemens (Administrador de Acme SA UY) y
     Perfil de Entidad: CA Administrators End Entity Profiles
     Nombre de Usuario: PatriciaClemens
     CN: Patricia Clemens
     OU: CA Administrator
     O: Acme SA
     C: UY
     Perfil Certificado: CA Administrators Cert Profile
     CA: AcmeSAUY

   - Jorge González (Administrador de Micros CA)

Perfil de Entidad: CA Administrators End Entity Profiles
Nombre de Usuario: JorgeGonzalez
CN: Jorge González
OU: CA Administrator
O: Micros
C: AR
Perfil Certificado: CA Administrators Cert Profile
CA: MicrosCA



7. El paso siguiente es la creación de los perfiles de Certificados que utilizarán cada una de las CAs. Para ello podemos ingresar al admin. Web con el usuario administrador de cada una de las CAs que hemos definido; también podemos hacerlo con el usuario superadministrador. En el caso de utilizar los administradores de cada CA, éstos solamente podrán definir perfiles de certificados y administrar entidades finales de su propia CA, puesto que han sido definidos con esas limitaciones.

   Creamos 2 tipos de certificados: uno estándar con 5 años de validez (Acme SA AR Certificado Estandar) y uno temporal con tan solo 10 días de validez (Acme SA AR Certificado Temporal).

8. Seguidamente debemos crear los Perfiles de Entidades Finales. Necesitaremos tres perfiles distintos: uno para emitir los Certificados de Administrador de RA y otros dos para los usuarios finales de cada una de las dos RA que tenemos.

   - Nombre del Perfil: Acme SA AR RA Perfil Administrador
   - Subject DN Files:
     CN: en blanco. Requerido y Modificable
     OU: RA Administrador Departamento Norte;
     RA Administrador Departamento Sur
     Requerido y No modificable

O: Acme SA AR Requerido y No modificable
C: AR Requerido y No modificable

- Perfil de Certificado por Defecto: Acme SA AR Certificado Estandar
- Perfiles de Certificados Disponibles: Acme SA AR Certificado Estándar y Acme SA AR Certificado Temporal
- CA por Defecto y Disponible: AcmeSAAR
- Tokens Disponibles: User Generated, P12 file
- Usar Administrador

Luego los dos perfiles que serán utilizados por los administradores de RA cuando registren entidades finales:

- Nombre del Perfil: Acme SA Departamento Norte
- Subject DN Files:
  CN: en blanco. Requerido y Modificable
  OU: Administración;Personal Requerido y No Modificable
  OU: Departamento Norte Requerido y No Modificable
  O: Acme SA AR Requerido y No modificable
  C: AR Requerido y No modificable
- Perfil de Certificado por Defecto: Acme SA AR Certificado Estandar
- Perfiles de Certificados Disponibles: Acme SA AR Certificado Estándar y Acme SA AR Certificado Temporal
- CA por Defecto y Disponible: AcmeSAAR
- Tokens Disponibles: User Generated, P12 file
- NO Usar Administrador

- Nombre del Perfil: Acme SA Departamento Sur
- Subject DN Files:
  CN: en blanco. Requerido y Modificable
  OU: Administración;Personal Requerido y No Modificable
  OU: Departamento Sur Requerido y No Modificable
  O: Acme SA AR Requerido y No modificable
  C: AR Requerido y No modificable
- Perfil de Certificado por Defecto: Acme SA AR Certificado Estandar
- Perfiles de Certificados Disponibles: Acme SA AR Certificado Estándar y Acme SA AR Certificado Temporal
- CA por Defecto y Disponible: AcmeSAAR
- Tokens Disponibles: User Generated, P12 file
- NO Usar Administrador

En estos perfiles, hemos creado dos Unidades Organizacionales como parte de los campos del DN. Uno es para separar el sector donde se desempañará y otro es para separar los Departamentos Norte y Sur.

Además hemos habilitado utilizar el e-mail en el Subject Alt Name (nombre RFC822) y hemos fijado que el dominio del mail deberá ser acme.com.ar para ambos casos.

9. El paso final será crear los privilegios para los nuevos administradores y agregarlos a ellos como usuarios del sistema.

   En primer lugar, definimos los grupos de administradores; necesitaremos tres nuevos grupos de administración:

   Uno para los Administradores de RA del Departamento Norte,
   Uno para los Administradores de RA del Departamento Sur y
   Uno para los Supervisores

   Grupo: RA Administradores Acme AR Dep. Norte

   Administrador: Nicolás Falcone
   Reglas de Acceso:
   - Rol: Administrador RA
   - CA Autorizada: AcmeSAAR (única disponible)
   - Reglas: aquí es posible seleccionar que está autorizada a hacer la RA con las Entidades Finales (Ver Entidades Finales, Ver Historia, Crear Entidad Final, Editar Entidad Final, Borrar Entidad Final, Revocar Entidad Final). Seleccionamos todas.
   - Perfil de Entidad Final: mediante esta opción, podemos indicar que requerimientos de perfil deben cumplir las entidades finales que se agregarán. En nuestro caso indicaremos que la RA está solamente autorizada a agregar entidades finales que cumplan los requerimientos del perfil de entidad final del Departamento Norte. De esta forma, al agregar una entidad final, le serán solicitados los datos definidos oportunamente en el perfil de este departamento.
     Seleccionamos entonces como Perfil de entidad final: Acme SA Departamento Norte.

   Grupo: RA Administradores Acme AR Dep. Sur

   Administrador: María Elena Toscano
   Reglas de Acceso:
   - Rol: Administrador RA
   - CA Autorizada: AcmeSAAR
   - Reglas: Seleccionamos todas las disponibles.
   - Perfil de Entidad Final: Acme SA Departamento Sur. En este caso las nuevas entidades deberán cumplir con el perfil definido para el Departamento Sur.

   Grupo: Supervisores Acme AR

   Administrador: Ricardo Pérez
   Reglas de Acceso:
   - Rol: Supervisor
   - CA Autorizada: AcmeSAAR
   - Reglas: al seleccionar el rol de supervisor, automáticamente se seleccionan las opciones: Ver Entidades Finales y Ver Historia

solamente. Esto es porque la función del supervisor no debería incluir el agregad, modificación o revocación de entidades finales, sino solamente efectuar tareas de control.
    - Perfil de Entidad Final: seleccionamos Todas, ya que el grupo debería tener acceso a todas las entidades finales creadas en esta CA.

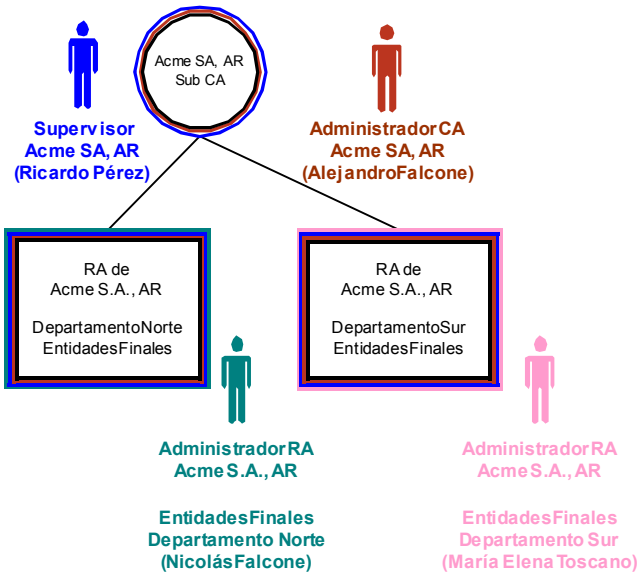
10. Ahora agregamos los administradores al sistema, con la opción de Agregar Entidades Finales.

- Perfil de Entidad Final: Acme SA AR RA Perfil Administrador
- Nombre Usuario: NicolasFalcone
- CN: Nicolás Falcone
- OU: RA Administrador Departamento Norte
- O: Acme SA AR
- C: AR
- Perfil de Certificado: Acme SA AR Certificado Estandar
- CA: AcmeSAAR
- Token: Archivo p12

- Perfil de Entidad Final: Acme SA AR RA Perfil Administrador
- Nombre Usuario: MariaElenaToscano
- CN: María Elena Toscano
- OU: RA Administrador Departamento Sur
- O: Acme SA AR
- C: AR
- Perfil de Certificado: Acme SA AR Certificado Estandar
- CA: AcmeSAAR
- Token: Archivo p12

- Perfil de Entidad Final: Acme SA AR RA Perfil Administrador
- Nombre Usuario: RicardoPerez
- CN: Ricardo Pérez
- OU: RA Administrador Departamento Norte
- O: Acme SA AR
- C: AR
- Perfil de Certificado: Acme SA AR Certificado Estandar
- CA: AcmeSAAR
- Token: Archivo p12

De esta manera tenemos construida una de las ramas de la PKI. El gráfico sería de la forma:



Los otros grupos, privilegios y entidades se crean de igual forma que la descripta.

Finalmente ejemplificaremos la creación de un usuario de la PKI construida.

Para ello podemos ingresar al admin. Web con el usuario administrador de RA del Departamento Sur (María Elena Toscano) y agregar una Entidad Final con las siguientes características:

- Perfil de Entidad Final: Acme SA Departamento Sur
- Nombre Usuario: GracielaVaena
- E-mail: graciela.vaena@acme.com.ar
- CN: Graciela Vaena
- OU: Administración
- OU: Departamento Sur
- O: Acme SA AR
- C: AR
- Perfil de Certificado: Acme SA AR Certificado Estandar
- CA: AcmeSAAR
- Token: Como buscamos que el usuario traiga su certificado mediante la interfase de web pública, indicamos: User Generated

Esta página fue dejada en blanco de manera intencional.

Conclusiones

Actualmente numerosas organizaciones se encuentran operando en todo el mundo con sistemas PKI. En nuestro país en particular, se encuentra tomando impulso la utilización de Firma Digital a partir de la ley N° 25.506 que reglamenta el licenciamiento de certificados digitales y de los proyectos implementados en la Administración Pública Nacional.

Nuestro objetivo inicial, proponer una arquitectura para montar Firma Digital con J2EE, tiene mayor sentido aún en estos momentos, cuando está tomando auge también la implementación de aplicaciones Open Source, las cuales han demostrado ser tan seguras y confiables como cualquier implementación comercial existente.

De acuerdo a la experiencia adquirida a partir de nuestro trabajo, podemos asegurar que existe una posibilidad concreta de implementar y construir una infraestructura PKI a partir de las especificaciones de J2EE. La arquitectura investigada, EJBCA, puede ser empleada de forma standalone o integrarse con cualquier aplicación J2EE existente; cumple con los estándares X.509 y IETF-PKI. Provee capacidad de administración mediante interfases de comandos o basadas en web y toda su construcción, configuración e implementación se efectúa con herramientas y aplicaciones Open Source.

Nuestro trabajó finalizó con la implementación de una infraestructura PKI de mediana complejidad, incluyendo la configuración de todas las componentes de dicha infraestructura que la hacen totalmente operativa.

Pretendemos que este informe pueda servir de guía para todo aquel que requiera montar una infraestructura PKI basada en las especificaciones J2EE, iniciándolo en los conceptos básicos necesarios para ello y presentándole una propuesta de arquitectura acompañada de un ejemplo de implementación paso a paso. Adicionalmente hemos migrado la aplicación al lenguaje español, ya que originalmente se encuentra en idioma inglés y no existen al momento traducción a nuestra lengua. La misma ha sido enviada a los desarrolladores para que puedan incluirla en futuras distribuciones del producto.

Esta página fue dejada en blanco de manera intencional.

Terminología y Abreviaciones

A continuación presentamos los términos y abreviaciones utilizados en el desarrollo del informe.

- **Active Directory (AD)**

Es un servicio de directorio que almacena información sobre objetos de una red y los pone a disposición de los usuarios. Fue desarrollado por Microsoft.

- **Algoritmo RSA**

Es el nombre del primer algoritmo de clave pública (o asimétrica) publicado. Su nombre lo debe a sus inventores Rivest, Shamir y Edelman. Maneja longitud de claves de 512, 1024 y 2048 bits.

- **Autenticidad – Authenticity**

La propiedad de ser genuino, capaz de ser verificado y, por lo tanto, confiable.

- **Autoridad de Certificación - Certification Authority (CA)**

Entidad que emite certificados digitales con el propósito de autenticar identidades. Debe gozar de la confianza de los usuarios. Es responsable de administrar el ciclo de vida de los certificados (determinar períodos de validez y revocación) y, frecuentemente, el de las claves asociadas a ellos.

- **Autoridad de Registración - Registration Authority (RA)**

Es una entidad opcional en una PKI, cuya principal misión es registrar y verificar la información necesaria para que la CA emita certificados. Su funcionamiento está regido por las políticas de la CA y puede existir una o más RAs conectadas a una CA.

- **Camino de certificación- + certification path - ruta de certificación**

Es una secuencia ordenada de certificados que le permite al usuario verificar la firma del último certificado de la secuencia, y así obtener la clave pública certificada de la entidad sujeto de ese último certificado.

- **Cifrado – encrypt – cipher**

Transformación criptográfica de datos (de texto plano) a una forma que oculta el significado original (llamado texto cifrado) para prevenir su conocimiento o uso; si la transformación es reversible, el proceso inverso que restaura los datos a su estado original se denomina descifrado.

- **Certificado digital - digital certificate**

Un certificado bajo la forma de objeto digital de datos usado por computadoras, al que se le adiciona un valor calculado de firma digital que depende del objeto.

- **Certificado raíz - root certificate**

En un sistema de certificados y claves públicas es el certificado que identifica una autoridad confiable, desde la cual se derivan otras relaciones de confianza. En la teoría de PKI, la confianza en una identidad, está basada en la confianza que uno tenga en la autoridad que valida esa identidad.

- **Clave privada - private key**

Una de las dos claves de la criptografía asimétrica. Es el componente secreto del par, y es conocido solamente por el usuario titular.

- **Clave pública - public key**

Es el otro componente del par de claves utilizadas en criptografía asimétrica; esta es la parte publicable y conocida.

- **Confidencialidad - data confidentiality - confidencialidad de los datos**

La propiedad de que la información no esté disponible ni sea divulgada hacia entidades del sistema no autorizadas.

- **Criptografía - cryptography**

Rama de la matemática que trata sobre la transformación de datos para hacer ininteligible su significado, prevenir su alteración no detectada o prevenir su uso no autorizado; si la transformación es reversible, también trata acerca de la restauración de los datos cifrados a su forma original.

- **Criptografía asimétrica - asymmetric cryptography**

Una de las ramas de la criptografía, conocida también como "criptografía de clave pública", cuyos algoritmos emplean un par de claves (una pública, otra privada) y usan diferentes componentes del par para distintas etapas de los algoritmos.

- **Criptografía simétrica - symmetric cryptography**

Rama de la criptografía en la que los algoritmos usan la misma clave para dos pasos diferentes, como el cifrado y el descifrado, o la firma y la verificación; se ha empleado por miles de años. Se la suele denominar de "clave secreta" (en oposición a la de "clave pública"), pues las entidades que comparten la clave necesitan mantenerla en secreto; esto incrementa el costo y los riesgos de la distribución segura de la clave, sobre todo en grandes organizaciones, asunto en el que la criptografía asimétrica tiene ventaja.

- **Criptosistema - cryptographic system**

Un criptosistema, o sistema criptográfico, se puede definir como los fundamentos y procedimientos de operación – algoritmo – que participan en el cifrado y descifrado de un mensaje.

- **CRL**

Véase Lista de Revocación de Certificados.

- **CSS (Cascading Style Sheets)**

Se trata de una especificación sobre los estilos físicos aplicables a un documento HTML, y trata de dar la separación definitiva de la lógica (estructura) y el físico (presentación) del documento.

- **Descifrado - decrypt**

Proceso de restablecer criptográficamente el texto cifrado a la forma de texto plano que tenía antes del cifrado.

- **Documento digital - digital document**

Un objeto electrónico de datos que representa información originalmente escrita en un medio no electrónico y no magnético (comúnmente tinta sobre papel), o es análogo a un documento de ese tipo.

- **EJBCA (Enterprise Java Bean Certificate Authority)**

Nombre elegido para una Autoridad de Certificación Open Source, construida puramente en JAVA, basada en componentes, con una arquitectura flexible e independiente de la plataforma. Puede ser operada en forma standalone o integrada con cualquier aplicación J2EE existente.

- **Encriptación**

Es el proceso de transformar datos a una forma no inteligible, de manera tal que los datos originales son obtenidos usando el proceso inverso de desencriptado.

- **Entidad Final (End Entity)**

Es un usuario dentro de una PKI, por ejemplo un cliente de e-mail, un servidor web, etc. Se podría ver como el nodo hoja dentro de la estructura PKI.

- **Firma digital - digital signature**

Valor calculado mediante un algoritmo criptográfico y añadido a un objeto de datos (es el llamado digesto del mensaje) de forma tal que cualquier receptor de los datos pueda usar la firma para verificar el origen de esos datos y su integridad.

- **Función de hash - hash function**

Algoritmo que calcula un valor basado en un objeto de datos (como un mensaje o archivo); este valor calculado tiene la propiedad de ser único, es decir, aplicando la función de hash a diferentes objetos, se obtienen diferentes valores resultantes.

- **Hard Token**

Dispositivo de hardware para proveer seguridad. Véase Smart Card.

- **GUID (Global Unique IDentifier)**

Es un conjunto de números que constituyen un identificador virtualmente único. Si bien no se puede garantizar que un número de GUID generado sea único, la cantidad total de claves únicas (2^{128}) es tan grande que la posibilidad de generar el mismo número dos veces es virtualmente cero.

- **IDE (Integrated Development Environment)**

Un Ambiente de Desarrollo Integrado (también conocido como *ambiente de diseño integrado* y *ambiente de debuggin integrado*) es un software para ayudar a los programadores a desarrollar software.

- **Identificación - identification**

Acto o proceso que presenta un identificador ante un sistema, de modo que el sistema pueda reconocer la entidad y distinguirla de otras.

- **Infraestructura de Clave Pública - public-key infrastructure (PKI)**

Sistema de CAs (y servidores y agentes auxiliares) que desempeña algún conjunto de funciones de administración (de certificados, de repositorios, de claves, de dispositivos, etc.) para una comunidad de usuarios que participan de ella, en aplicaciones de criptografía asimétrica.

- **Integridad - data integrity - integridad de los datos**

Propiedad que garantiza que los datos no hayan sido cambiados, destruidos o perdidos, en forma no autorizada o accidental, sin que uno no se haya enterado de ello; no se refiere al significado de los datos ni a la certeza de la fuente que los originó.

- **JBoss**

Servidor de aplicaciones J2EE, ampliamente utilizado y Open Source. También es el nombre de la empresa que provee el soporte comercial de este producto.

- **JDBC (Java Database Connectivity)**

Es una API de Java que define como un cliente puede conectarse a una base de datos. Provee métodos para ejecutar instrucciones SQL sobre ellas.

- **JNDI (Java Naming Directory Interface)**

Es una especificación que permite localizar información en distintos directorios distribuidos, directorios LDAP o servicios CORBA. Es una API para servicios de directorios.

- **KeyStore**

Son archivos que los productos utilizan para almacenar claves y certificados utilizados para proteger y verificar sitios web, contenidos de sitios web, y contenidos de archivos y mensajes. Contiene también notas de usuario sobre cualquier información almacenada en él. Son protegidos de ataques mediante criptografía.

- **Lista de certificados revocados - Certificate Revocation List (CRL)**

Estructura de datos que enumera certificados digitales que su emisor ha invalidado por diversos motivos, con anticipación su la fecha de vencimiento.

- **Longitud de Clave – Key length**

La longitud de una clave es el número de bits que el valor de la clave ocupa. Esta longitud se considera como una medida de seguridad de la clave, a mayor longitud mejor seguridad.

- **MS UPN (MS User Principal Name)**

Nombre de entidad de usuario. Utilizado en Windows 2000 Active Directory, basado en el estándar de Internet RFC 822.

- **Nombre común - common name (CN)**

Cadena de caracteres, posiblemente ambigua, por la cual se conoce a un objeto dentro de un campo limitado (por ejemplo, dentro de una organización), y que está construida según las convenciones culturales asociadas; ejemplos: "Jorge González", "AdminCA1", "Centro de Cómputos".

- **Normas de Criptografía de Clave Pública - Public-Key Cryptography Standards (PKCS)**

Serie de especificaciones publicadas por los Laboratorios RSA para las estructuras de datos y uso de algoritmos en aplicaciones básicas de criptografía asimétrica; si bien se emplean ampliamente, no están respaldadas por ningún organismo normalizador oficial.

- **Nombre distinguido - distinguished name (DN)**

Identificador que representa un objeto de forma única, en el árbol de información de directorio X.500; es un conjunto de valores de atributos que identifican la ruta desde la raíz del árbol hasta el objeto que es nominado. Por ejemplo, los certificados digitales X.509 contienen el DN del sujeto y el DN del emisor (la autoridad de certificación).

- **PKIX**

Grupo de trabajo que está especificando la arquitectura y el conjunto de protocolos necesarios para que Internet soporte una PKI basada en X.509; su objetivo es facilitar el uso de certificados de clave pública en múltiples aplicaciones de Internet y promover la interoperabilidad entre diferentes implementaciones. Es la abreviatura utilizada para "Infraestructura de Clave Pública X.509".

- **PKCS12**

Abreviación de Public Key Criptography Standard #12. Fue desarrollado por RSA y posteriormente adoptado por la industria. Es un método de almacenamiento de una clave pública y privada en un certificado. A menudo se utiliza como medio seguro para transferir claves a usuarios, y es encriptado utilizando una clave o password secreta.

- **Plugin (Plug-in)**

Son módulos de software o hardware que agregan una característica o servicio a un sistema.

- **Política del certificado - certificate policy (CPS=certification policy statements)**

Conjunto de reglas que indica la aplicabilidad de un certificado a una comunidad particular y/o a una clase de aplicaciones con requerimientos de seguridad compartidos; ayuda al usuario del certificado a decidir si puede confiar en el mismo para una situación particular.

- **Privacidad - privacy**

El derecho que una entidad (normalmente una persona) tiene, actuando por sí misma, de determinar su grado de interacción con el entorno, incluyendo el grado en que desea compartir con otros información sobre sí misma; es un motivo para proveer seguridad y no un tipo de servicio de seguridad.

- **Protocolo ON Line de Estado de Certificados (OCSP= OnLine Certificate Status Protocol)**

Servicio que recibe un requerimiento de validez de un certificado digital y retorna una respuesta sobre el estado del mismo, es decir si se encuentra revocado o no.

- **Repudio - repudiation**

Negativa de una entidad que ha estado involucrada en una asociación (especialmente para transferir información), de haber participado en la relación.

- **Sandbox**

Es una medida de seguridad en el ambiente de Java. Conjunto de reglas que limita ciertas funciones cuando el applet se envía como parte de una página web.

- **Simple Certificate Enrollment Protocol (SCEP)**

Protocolo de comunicación para interoperabilidad en la infraestructura PKI desarrollado por Cisco. Es una de las primeras especificaciones de su tipo en ser adoptadas por una gran cantidad de fabricantes, ya que ofrece un método común y consistente para solicitar y recibir certificados digitales por parte de diversas autoridades de certificación.

- **Smart Card**

Dispositivo del tamaño de una tarjeta de crédito que contiene un microprocesador, memoria (para almacenar programas y datos) y contactos eléctricos usados como interfase con el lector de tarjetas. Una Smart Card usualmente contiene un valor secreto, tal como un número secreto o una clave privada usada en el proceso de creación de la firma digital.

- **Texto Plano – plain text**

Datos de entrada que serán transformados por un proceso de cifrado, o datos de salida de un proceso descifrador.

- **Texto cifrado – cipher text**

Resultado de aplicar un proceso de cifrado a un texto. Usualmente, la entrada al cifrador es texto plano pero, en algunos casos, puede ser texto cifrado proveniente de la salida de otra operación de cifrado.

- **URI**

Abreviatura de Uniform Resource Identifier. Es el término genérico para todo tipo de nombres y direcciones de objetos en el WWW. Un URL es una clase de URI.

- **X.509**

Se refiere a un estándar desarrollado por la ITU (Interantional Telecommunications Union) y posteriormente modificado por la IETF. El estándar, concierne a la definición de un registro en una base de datos que almacena certificados de clave pública para accederlos a través de una PKI.

Esta página fue dejada en blanco de manera intencional.

Referencias

Capitulo 1 - Introducción y Conceptos Generales

[1] http://web.mit.edu/kerberos/www/

[2] http://www.itl.nist.gov/fipspubs/fip46-2.htm

[3] http://csrc.nist.gov/publications/nistpubs/800-20/800-20.pdf

[4] http://www.cs.nps.navy.mil/curricula/tracks/security/notes/chap04_43.html

Capítulo 2 – Infraestructura PKI

[1] ftp://ftp.rfc-editor.org/in-notes/rfc2560.txt

[2] IETF: Internet Engineering Task Force's - http://www.ietf.org

[3] http://www.ietf.org/rfc/rfc2527.txt

[4] http://www.ietf.org/html.charters/pkix-charter.html

[5] CMP (Certificate Management Protocols)
http://www.ietf.org/internet-drafts/draft-ietf-pkix-rfc2510bis-09.txt
CMRF (Certificate Management Request Format
http://www.ietf.org/internet-drafts/draft-ietf-pkix-rfc2511bis-09.txt)

[6] http://www.entrust.com/resources/docs/protocols_pki.htm

Capítulo 3 – Certificados Digitales

[1] ITU: International Telecommunication Union
http://www.itu.int/home/

[2] ISO/IEC: International Organization for Standardization /
International Electrotechnical Commission
http://www.iso.org

[3] http://www.ietf.org/rfc/rfc2560.txt

[4] http://www.w3.org/TR/xkms/

[5] ITU-T: Telecommunication Standardization Sector del ITU
http://en.wikipedia.org/wiki/ITU-T

Capítulo 4 – Firma Digital

[1] "Utilizando Firma Digital" - Trabajo de Grado de las Alumnas
Verónica Fredes y Paula Venosa – 2001 – Facultad de Informática – UNLP

Capítulo 5 – J2EE – Java 2 Platform Enterprise Edition

[1] http://en.wikipedia.org/wiki/Plugin
[2] http://en.wikipedia.org/wiki/Sandbox_%28computer_security%29

Capítulo 6 – Arquitectura J2EE

[1] http://java.sun.com/developer/technicalArticles/J2EE/Intro/index.html#apis

Capítulo 7 – Introducción EJBCA

[1] http://www.opensource.org/licenses/lgpl-license.html
[2] FSF: Free Software Foundation (Fundación para el Software Libre)
http://www.fsf.org
[3] http://www.opensource.org
[4] http://sourceforge.net/projects/ejbca/
[5] POSIX: Portable Operating System Interface for Unix
http://en.wikipedia.org/wiki/POSIX
[6] AIA: Authority Information Access
http://www.ietf.org/proceedings/00jul/SLIDES/pkix-tsp/tsld007.htm

Capítulo 8 – Construcción y Configuración de EJBCA

[1] http://java.sun.com/products/jce/
[2] http://java.sun.com/j2se/1.4.2/download.html
[3] http://en.wikipedia.org/wiki/JDBC
[4] http://dev.mysql.com/downloads/connector/j/3.0.html
[5] http://en.wikipedia.org/wiki/JNDI

Capítulo 9 – Interfase Web de EJBCA

[1] http://localhost:8442/ejbca/

Capítulo 10 – Administración mediante Interfase Web

[1] https://localhost:8443/ejbca/adminweb
[1] http://en.wikipedia.org/wiki/Cascading_Style_Sheets

Capítulo 11 – Administración mediante Línea de Comandos

[1] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/adschema/adschema/a_userprincipalname.asp

[2] http://en.wikipedia.org/wiki/GUID

Capítulo 12 – Dependencias

[1] http://www.bouncycastle.org/
[2] http://www.jboss.org/
[3] http://ant.apache.org/
[4] http://en.wikipedia.org/wiki/Integrated_development_environment
[5] http://logging.apache.org/
[6] http://www.junit.org/
[7] http://oss.software.ibm.com/developerworks/opensource/license10.html
[8] http://www.openLDAP.org/

Capítulo 13 – APIs de EJBCA

[1] http://ejbca.sourceforge.net/docs/api/index.html

Esta página fue dejada en blanco de manera intencional.

Bibliografía

- **Firma Digital – Infraestructura PKI - Criptografía**

Legal aspects of PKI - White Paper – TrustWeaver AB

Cryptography - http://world.std.com/~franl/crypto.html

Introduction to PKI-Public Key Infrastructure
European Master in Multimedia Projects – Prof.:M. Van Droogenbroeck – Mayo 2002

X.509 (2000): 4th edition: Overview of PKI & PMI Frameworks
Sharon Boeyen Principal, Advanced Security Entrust, Inc.
http://www.entrust.com/resources/download.cfm/21140/509_overview.pdf

PKI Interoperability Framework
PKI Forum's Technology Working Group (TWG). Marzo 2001

PKI Basics - A Technical Perspective
PKI Forum's Business Working Group (BWG). Nov. 2002

PKI Basics A technology tutorial - TrustWeaver AB.

Architecture for Public-Key Infrastructure (APKI)
Open Group - Mayo 1997

Usando Infraestructura PKI para implementar FIRMA DIGITAL
Lic. Javier F.Díaz, A.C Paula Venosa.
Presentado en VI Congreso Internacional de Ingeniería Informática ICIE Y2K, Abril 26-28, 2000

Utilizando Firma Digital
Trabajo de Grado de las Alumnas Verónica Fredes y Paula Venosa
Facultad de Informática - UNLP - Agosto 2001

Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure
C. Ellison and B. Schneier - Computer Security Journal, v 16, n 1, 2000, pp. 1-7.

My response to Carl Ellison's and Bruce Schneier's "Ten Risks of PKI."
Aram Pérez
http://home.pacbell.net/aram/responsetenrisks.html

The Open-source PKI Book – A guide to PKIs and Open-source Implementations
Symeon (Simos) Xenitellis - The Open–source PKI Book Version 2.4.7
Edition 23 July 2000

SPKI/SDSI Certificates – Carl M.Ellison -17 November 2002
http://world.std.com/~cme/html/spki.html

Digital Signature Guidelines
Information Security Committee, Section of Science & Technology
American Bar Association – August 1996

Johnson & Johnson - Technology Consultants, LLC – Home Page
http://www.jjtc.com/Security/crypto.htm

Infraestructura de Firma Digital de la República Argentina
http://www.pki.gov.ar

The Internet Engineering Task Force
http://www.ietf.org/

- **J2EE**

Arquitectura empresarial y software libre, J2EE
Martín Pérez Mariñán y Alberto Molpeceres Touris - Agosto 2002

Java 2 Platform, Enterprise Edition (J2EE)
http://java.sun.com/j2ee/
http://java.sun.com/j2ee/compatibility.html

Home Page de Jboss - Servidor de aplicaciones J2EE Open Source.
http://www.jboss.org

Java 2 Enterprise Edition Technology Center
Monica Pawlan - March 23, 2001

Java™ 2 Platform Enterprise Edition Specification, v1.4
Final Release - 11/24/03 Bill Shannon - Sun Microsystems, Inc.

The J2EETM Tutorial
Stephanie Bodoff, Dale Green, Eric Jendrock, Monica Pawlan, Beth Stearns
2001 Sun Microsystems, Inc

High Availability for J2EE Platform-Based Applications
Damian Guy, Allan Packer, and Tom Daly - January 2002

J2EE Architecture
Dr. Michelle Lee - 2001 – (J2EEArchitecture.ppt)

- ## Open Source

Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!
David A. Wheeler – Enero 2001

Competing on Open Source: Strategies and Practise.
Pal, N. & Madanmohan, T. - Indian Institute of Management-Bangalore.
MIT Free/Open Source Software Research Project - (2002)

Is Open Source software development an evolution or a revolution?
Sharon Parker - DISS 790 E-Commerce - Sep 30 2002

The Open Source Reader
Compilation By: Felipe Csaszar - Version 1.2 – Nov 22 2001

Open Sources: Voices from the Open Source Revolution
Libro OnLine (y versión impresa) con diversos ensayos acerca de Open Source
O'Reilly - 1st Edition - January 1999
http://www.oreilly.com/catalog/opensources/book/toc.html

Open-source – Pki Book
http://ospkibook.sourceforge.net/

Open Source Initiative (OSI)
http://www.opensource.org/

OpenCA Labs
http://www.openca.org/

Our Open Source / Free Software Future: It's Just a Matter of Time
Micah Yoder - Version 1.2, 5/15/2002
http://www.yoderdev.com/oss-future.html

Free/open source software
http://opensource.mit.edu/

The Free Software Foundation.
Richard Stallman. The originators of the GPL and original Open Source and Free
http://www.fsf.org

- ## EJBCA

EJBCA - Enterprise Java Beans Certificate Authority
http://ejbca.sourceforge.net/

PrimeKey Solutions
http://www.primekey.se/primekey/en.html
http://docs.primekey.se/documentation/en.html

EJBCA: An Open Source, Java-based Certificate Authority
Todd Sundsted - 28 May, 2002
http://www.computerworld.com.au/index.php?id=1857930161

- ## Informes Generales

Home Page de Netcraft Ltd.
http://www.netcraft.com

Home Page de IDC
http://www.idc.com

Home Page del Gartner Group
http://www.gartner.com

Home Page de SPEC Consortium (Standard Performance Evaluation Corporation)
http://www.spec.org

Home Page de Bloor Research
http://www.bloor-research.com