Indice

Motivación

Dentro del amplísimo marco de ventajas y oportunidades que Internet ofrece, queda al descubierto la imperiosa necesidad de seguridad y confidencialidad. Amenazas de fraude, espionaje y robo de información han limitado que los usuarios disfruten en su totalidad los beneficios del mundo electrónico.

Dado que el intercambio electrónico de información constituye un tema de creciente importancia, el hecho de disponer de una infraestructura que permita garantizar la identidad de un ciudadano a través de Internet permite trasladar a la red un amplio abanico de posibilidades que, en la actualidad, en la mayoría de los casos, sólo pueden llevarse a cabo haciendo coincidir en el tiempo la presencia física de las partes.

A efectos de que los sistemas abiertos puedan ser utilizados para transportar todo tipo de información sensible, es indispensable poder identificar a las personas que participan en las comunicaciones, independientemente del lugar físico utilizado. Son muchas las cuestiones que se plantean en relación a ello: ¿cómo podemos reconocer y confiar en gente a la que no vemos, escuchamos, o ni siquiera conocemos su firma? ¿Cómo mantenemos en secreto nuestras transacciones comerciales sin sobres sellados o llamadas telefónicas privadas? ¿Cómo sabemos que el mensaje ha llegado intacto a la persona a la que iba destinada, y que ésta ha dado su aprobación al contrato?

La tecnología requerida para lograr la seguridad a la que se ha hecho referencia ya existe en la forma de **Firma Digital**. La arquitectura que sustenta esta tecnología se conoce como **Infraestructura de Clave Pública** o **PKI.**

Es necesario tener en cuenta lo expresado para comprender que la firma digital no ha sido ideada en base al concepto tradicional de firma, y mucho menos teniendo en cuenta las particularidades de un ordenamiento jurídico determinado.
Nos encontramos frente a un mecanismo técnico que presenta ciertos atributos que se asemejan a los de la firma tradicional que puede ser utilizado eficientemente y sin mayores complicaciones o exigencias para cierto tipo de operaciones, pero que necesita de un estudio profundo y seguramente complementado con otro tipo de sistema, para extender su utilización a actos de mayor trascendencia y envergadura. En este caso, la solución a adoptar en cada país o región seguramente diferirá según las características del sistema de derecho imperante en ellos.

El objetivo planteado para la realización de este trabajo fue la construcción de una aplicación práctica que ilustre las ventajas de la firma digital con el fin de acelerar y transparentar procesos administrativos.

Para ello se eligió trabajar usando entornos abiertos y extensibles. Fue necesario desarrollar e instalar todas las componentes necesarias para contar con una Infraestructura de Clave Pública segura y confiable.

Estructura del trabajo

La presente tesis desarrolla todos los conceptos involucrados en la implementación de la firma digital. Describe en profundidad el funcionamiento de la PKI y de cada uno de sus componentes. Al mismo tiempo detalla las herramientas utilizadas en la arquitectura del modelo construido así como los puntos más relevantes respecto a la implementación del prototipo de nuestro sistema.

A continuación damos un breve resumen de los tópicos que cubre cada capítulo de este trabajo:

Motivación

Presenta el contexto y plantea el objetivo de la tesis.

Capítulo 1 - Introducción

Describe la problemática que constituye el marco de nuestro trabajo.

Capítulo 2 - Criptografía

Introduce las características de la criptografía, incluyendo sus variantes (sistemas criptográficos simétricos, asimétricos e híbridos). Detalla también algunos de los algoritmos criptográficos más importantes (DES, RSA, etc).

Capítulo 3 - Estándares PKC

Explica los estándares de criptografía de clave pública indicando el propósito de cada uno de ellos.

Capítulo 4 – PKI

Revela las características de la Infraestructura PKI dando a conocer sus usos, sus ventajas y desventajas, así como sus componentes y los objetos en ella involucrados. De cada uno de estos componentes se realiza una descripción en las secciones:

- *Certificados*
- *Autoridades de Certificación*
- *Repositorios de datos*
- *Autoridades de Registración*
- *Listas de Revocación*
- *PKIX*

Capítulo 5 – APuN-CA

Presenta el producto desarrollado por la Administración Pública Nacional como ejemplo de una herramienta para construir una Infraestructura PKI.

Capítulo 6 – Firma Digital

Detalla todos los conceptos relacionados con la firma digital y su utilización, sin dejar de contemplar el entorno legal en el cual la misma se desarrolla.

Capítulo 7 – Servicios de Seguridad en Internet

Describe algunos de los mecanismos existentes para proveer seguridad a los servicios de correo electrónico y de WEB. Muestra también el comportamiento de algunos protocolos específicos disponibles para tal fin como son S/MIME, PEM, PGP, MOSS y SSL.

Capítulo 8 – Implementación

Explica todos aspectos relacionados con la implementación del sistema. Este comienza por el objetivo del mismo, pasando luego por la definición de la arquitectura, la descripción de las herramientas utilizadas, la funcionalidad e interfaz de la aplicación y el modelo de datos elegido. En última instancia, explica minuciosamente los mecanismos usados para firmar digitalmente las transacciones realizadas a través del sistema así como los documentos y los mensajes de correo electrónico enviados.
Este contenido se encuentra en las siguientes secciones:

- *Arquitectura del Sistema*
- *Descripción de herramientas utilizadas*
- *Funcionalidad del Sistema*
- *Interfaz del Sistema*
- *Modelo de datos*
- *Firma digital en el Sistema*

Experiencias

Describe la experiencia obtenida a partir de la realización de este trabajo.

Conclusiones

Menciona las conclusiones de este informe.

Glosario

Define los términos técnicos frecuentemente utilizados en este informe.

Referencias

Da las referencias presentes en cada uno de los capítulos de este informe.

Bibliografía

Lista la bibliografía utilizada para la realización de este informe.

Introducción

Todos los días, tanto las empresas y como las personas, usan la Internet para realizar miles de transacciones en línea. Los empleados comparten archivos e información confidencial vía correo electrónico o a través de la red; los clientes de un banco actualizan sus cuentas o pagan impuestos desde la computadora en sus hogares y una gran variedad de productos son adquiridos a través de formularios en línea (comúnmente conocidos como formularios on-line).

Como una herramienta de negocios, la Internet tiene el potencial de reemplazar el teléfono y el fax en una transacción diaria (Internet es un canal de flujo hecho a la medida: rápido, barato y cada vez más extendido y eficiente).

Por otro lado, la tecnología de Internet también puede ser usada para otros fines como ser: interceptar y falsificar mensajes, robar información importante y espiar a organizaciones e individuos.

Lógicamente, conforme más información hay disponible en Internet, más importancia cobra la protección de esa información y el control del acceso a la misma. Dentro de este panorama, nos enfrentamos a una creciente realidad, la necesidad de seguridad en los datos, los servicios, las transacciones, y las partes involucradas.

Generalmente, en las transacciones basadas en documentación escrita existen mecanismos que garantizan:

- ✓ *Confidencialidad* (asegura que el contenido del mensaje es privado[1]).
- ✓ *Autenticidad* (garantiza que el mensaje proviene de la persona que efectivamente lo está enviando).
- ✓ *Integridad* (asegura que el contenido del documento no ha sido modificado)
- ✓ *No repudio* (implica que no sea posible que, finalizada la transacción, el emisor niegue haber enviado el documento o el receptor niegue haberlo recibido).

El objetivo de la seguridad es garantizar la privacidad de la información y la continuidad del servicio, tratando de minimizar la vulnerabilidad de los sistemas ó de la información contenida en ellos, así como tratando de proteger las redes privadas y sus recursos mientras que se mantienen los beneficios de la conexión a una red pública.

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. Las técnicas criptográficas, tales como encriptación y firma digital, constituyen piezas fundamentales para la implementación de los mismos.

[1] Se entiende como privado un mensaje que no puede ser visto por ninguna persona excepto el destinatario.

Criptografía

El crecimiento exponencial de los usuarios y organizaciones conectadas a Internet ha originado el tránsito de información de todo tipo a través de ella, desde noticias hasta transacciones complejas que requieren medidas específicas de seguridad que garanticen la confidencialidad, la integridad y la constatación del origen de los datos.

Cuando un emisor y un receptor quieren intercambiar mensajes, es posible que un espía quiera intervenir de algún modo en la comunicación, a éste se lo conoce comúnmente como intruso. Un intruso puede ser pasivo, si sólo escucha la comunicación, o activo si trata de alterar los mensajes.

Es aquí donde aparece el concepto de criptografía, la cual tiene como objeto proporcionar comunicaciones seguras sobre canales inseguros.

La criptografía[1], del griego "escritura oculta", ha evolucionado desde las técnicas de transformaciones y sustituciones de símbolos ya utilizadas en las antiguas civilizaciones, a los métodos basados en algoritmos matemáticos.

El objetivo de la criptografía es proteger la información de forma tal que la misma tenga significado únicamente para el destinatario. Para ello la criptografía lleva a cabo el proceso de cifrado, transformando el texto plano[2] en texto cifrado.

Al proceso inverso, el cual transforma un mensaje cifrado en el texto plano correspondiente, se lo llama descifrado.

Idealmente los procesos de cifrado y descifrado no requieren demasiado esfuerzo, siempre que las claves involucradas sean conocidas.

En caso de que un intruso trate de descubrir el contenido del mensaje cifrado, el proceso de descifrado debería ser lo suficientemente costoso en lo que se refiere a tiempo y recursos invertidos en dicho intento, de modo tal de desanimar al espía.

Una comunicación, protegida o no mediante sistemas criptográficos, está sujeta a una gran variedad de ataques de los cuales es imposible dar una taxonomía completa. A continuación de detallan algunos de los más habituales:

- ✓ *Ataque sólo al criptograma:* es el más complicado para el intruso. En este caso sólo tiene acceso al texto cifrado. Su trabajo consiste en recuperar el texto plano de tantos mensajes como le sea posible. En tales condiciones, sólo puede intentar vulnerar dicho algoritmo, realizar un análisis estadístico de los criptogramas o probar todas las claves posibles del algoritmo. Este último caso, se conoce como búsqueda exhaustiva o también como ataque basado en fuerza bruta.
- ✓ *Ataque mediante texto plano conocido:* en este ataque, más ventajoso para el atacante dado que el mismo puede poseer pares de texto cifrado y su correspondiente texto plano (este último puede haber sido adivinado por el atacante de algún modo, a partir de lo cual el contenido del mensaje es conocido). Estos pares pueden ser usados para llevar a cabo el criptoanálisis y averiguar la clave, lo cuál será útil si se usa la misma clave para posteriores comunicaciones.
- ✓ *Ataque mediante texto plano conocido:* este ataque es mucho más eficaz que el anterior, en este caso el intruso es capaz, de algún modo, de conseguir que un texto elegido por él sea cifrado con la clave desconocida. Por lo tanto hay que diseñar el sistema criptográfico de modo tal que un intruso nunca pueda introducir mensajes propios.
- ✓ *Ataque adaptable mediante texto plano escogido:* es un caso especial del anterior en el que el intruso no sólo puede elegir el texto que quiere cifrar, sino que puede tomar decisiones sobre el texto a ser cifrado basándose en resultados anteriores.

[1] Criptografía: La criptografía es la ciencia que se encarga de mantener los mensajes en forma secreta.
[2] Texto plano: La representación del mensaje original es conocida como texto plano.

- ✓ *Ataque mediante criptogramas escogidos:* el atacante puede obtener el descifrado de diversos mensajes cifrados escogidos por él.

Los ataques que se detallan a continuación pueden servir para implementar alguno de los anteriormente descriptos:

- ✓ *Escucha pasiva (passive eavesdropping):* el intruso simplemente escucha el tráfico que circula por el canal
- ✓ *Tercero en el medio (man-in-the-middle):* el intruso, de alguna forma, se coloca entre los dos interlocutores y hace creer a cada uno de ellos que es el otro.
- ✓ *Retransmisión ciega (replay):* el intruso intercepta un mensaje legítimo, lo almacena (sin eliminarlo) y lo reenvía un tiempo después.
- ✓ *Cortado y Pegado (cut-and-paste):* dados dos mensajes cifrados con la misma clave, a veces es posible combinar partes de los dos para producir uno nuevo. El intruso no sabe lo que dice este nuevo mensaje, pero puede utilizarlo para confundir a los interlocutores legítimos e inducir a algunos de ellos a hacer algo beneficioso para él.
- ✓ *Puesta a cero del reloj (time-resetting):* en protocolos que utilizan de alguna forma la hora actual, el intruso puede tratar de confundir a las partes que se están comunicando acerca de cuál es la verdadera hora.

Podemos concluir entonces que un criptosistema[3] debe ser lo suficientemente robusto como para no poder deducir la clave a partir del conocimiento del texto plano, el texto cifrado y el algoritmo utilizado.

La seguridad no debe residir en el algoritmo, el cual debe ser absolutamente conocido por todos, sino en la clave. Por esta razón es de vital importancia el tamaño de la clave utilizada, de forma tal que sea muy poco práctico aplicar técnicas de ataque de fuerza bruta a los criptosistemas que estén expuestos.

Los algoritmos de cifrado tradicionales utilizan una única clave, la cual es compartida por el emisor y el receptor. El emisor usa dicha clave para cifrar el mensaje y el receptor realiza el proceso de descifrado utilizando también la clave en cuestión. Este mecanismo se conoce como **criptografía simétrica**.

La criptografía simétrica tiene el beneficio de realizar un rápido procesamiento de los datos. Sin embargo posee importantes desventajas, dado que la clave debe ser compartida por ambas partes involucradas en la comunicación, por lo cual es necesario contar con un mecanismo seguro de distribución de claves. Por otro lado, si alguien desea comunicarse con varias personas de manera segura, debe contar con una clave por cada comunicación que desea establecer, lo que resulta poco práctico.

En este tipo de criptografía la seguridad de la clave depende de ambas partes involucradas, es decir que tanto el receptor como el emisor pueden comprometer la clave. Dichas desventajas son resueltas a partir del uso de la **criptografía asimétrica o de clave pública**. Este mecanismo utiliza dos claves diferentes pero matemáticamente relacionadas. Estas claves complementarias se denominan “clave privada” y “clave pública” respectivamente.

La finalidad de este criptosistema es proveer a cada usuario de un único par de claves (una pública y una privada) independientemente del número de usuarios con los que desee comunicarse. Este par de claves tiene la propiedad que cada una de las claves invierte la acción de la otra pero, y aquí está el punto más relevante: a partir de una no se puede obtener la otra. Un usuario da a conocer su clave pública a otros usuarios y guarda en secreto su clave privada. Ambas claves pueden ser usadas para cifrar y descifrar datos.

[3] Criptosistema: define un par de transformaciones de datos llamadas encriptado y desencriptado, respectivamente.

Criptografía simétrica

En este tipo de criptosistema, los mecanismos de cifrado, también conocidos como cifradores pueden dividirse en dos grupos:

- ✓ Cifradores de bloque
- ✓ Cifradores de flujo (stream)

En un *cifrador de bloque*, la operación de cifrado o encripción funciona con un tamaño fijo de bloque o texto plano, de n bits (generalmente los bloques son de 64 bits). La función de descifrado opera con bloques de n bits de texto cifrado y genera bloques o texto plano de n bits.

Por otro lado, un *cifrador de flujo* puede operar sobre texto plano de cualquier tamaño, generando texto cifrado del mismo tamaño. Un cifrado de flujo generalmente procesa los datos como una secuencia de caracteres, donde cada carácter puede ser considerado como un bit. Usualmente, los cifradores de flujo son construidos usando el cifrador de bloques como base para el armado de bloques. Esto da como resultado la definición de varios *modos de operación* de cifrado de bloques. Los dos modos de operación más usados en aplicaciones de procesamientos de datos son: CBC (Cadena de cifrado de bloque) y CFB (Regeneración de cifrado)

Algunos ejemplos de algoritmos de clave simétrica son: DES (Data Encryption Standard) [ref. 1], IDEA (International Data Encryption Algorithm) [ref. 2], RC2 [ref. 3],TRIPLE DES [ref. 4].

DES

DES es el nombre del Estandar de Procesamiento de Datos. Creado en 1973, fue el primer criptosistema simétrico desarrollado para uso comercial. Describe el Algoritmo de Encripción de Datos (DEA) que es también definido en el Estandar ANSI [ref. 5].

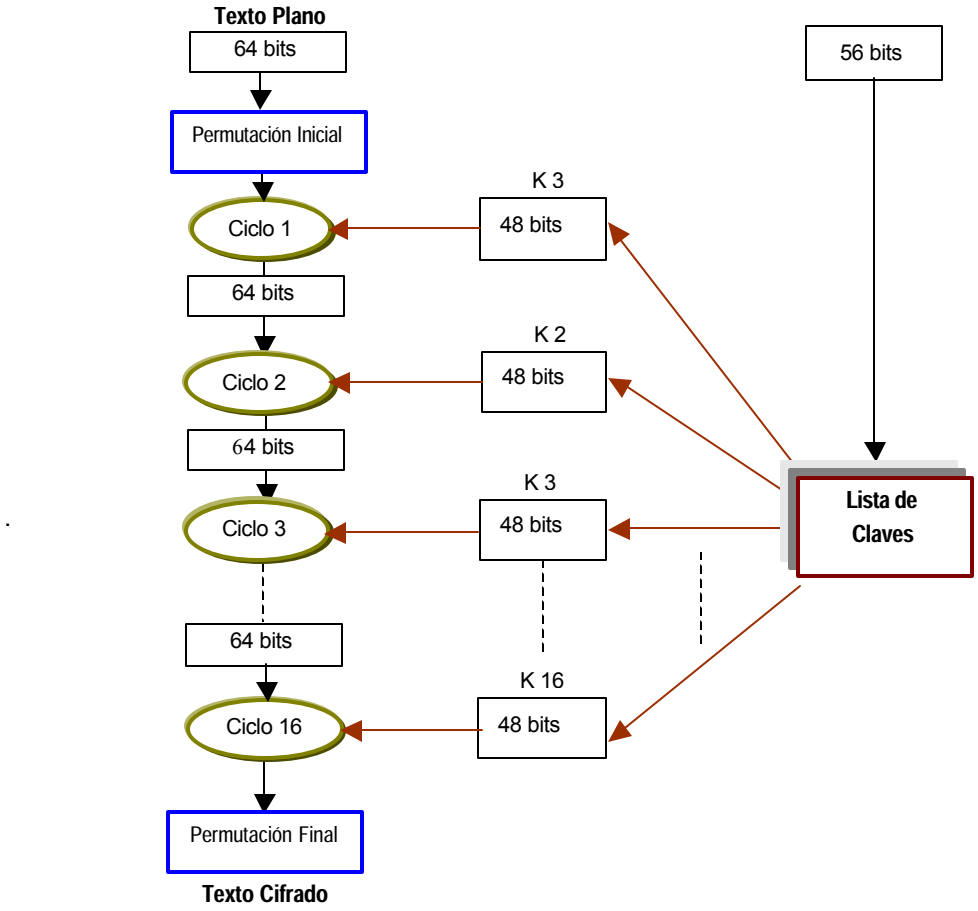
Originalmente fue desarrollado por IBM y conocido como Lucifer. La Agencia de Seguridad Nacional (NSA) y el NBS (National Bureau of Standards) ahora conocido como NIST, jugaron un rol sustancial en la finalización de su desarrollo.

Como en cualquier otro esquema de cifrado, la función de encripción DES recibe dos entradas: el texto plano a ser cifrado y la clave. Con DES el texto plano se divide en bloques de 64 bits y la clave es de 56 bits. El proceso de cifrado se inicia con una permutación de bits del texto plano (bloque de 64 bits). Luego, el resultado pasa por 16 iteraciones de cálculos dependientes de una subclave, y finalmente, se aplica una permutación para obtener el texto cifrado.

El proceso de descifrado toma el texto cifrado como entrada, pero usa las subclaves generadas en cada iteración en orden inverso.

Cualquier cifrador de bloque puede ser atacado usando una técnica de búsqueda exhaustiva. Un sistema atacante que no conoce la clave privada, sistemáticamente trata de desencriptar el bloque cifrado, usando todos los posibles valores de claves hasta que se encuentra el valor que desencripte el bloque. Si suponemos un sistema de encripción donde todas las claves son igualmente factibles, la probabilidad de éxito de una búsqueda exhaustiva depende del tamaño de la clave. Con DES, este valor es 2^{56}.

Actualmente, dado el rápido avance de la tecnología, se ha encontrado un punto donde cualquier criptosistema que maneje claves de 56 bits es cuestionable porque es susceptible a ataques de búsqueda exhaustiva o fuerza bruta.



Algoritmo DES – Proceso de Cifrado

Cuando este algoritmo es usado para comunicaciones, tanto el emisor como el receptor deben conocer la misma clave secreta, la cual puede ser usada para cifrar y descifrar el mensaje, así como para generar y verificar un código de autenticación de mensaje(MAC). También puede ser usado para cifrar archivos que luego serán almacenados en el disco

TRIPLE DES

El tamaño de la clave utilizada por DES puede ser incrementado usando múltiples métodos. Triple DES incluye: un cifrado inicial de un bloque de 64 bits usando una clave **A**, seguido por el descifrado del resultado usando una clave **B**, y finalmente seguido de la encripción del resultado último usando una clave **C**.

Aquí se utilizan 2 o 3 claves (a veces **A** y **C** pueden ser la misma clave). El algoritmo resultante es considerado más confiable y más robusto que DES.

SKIPJACK

En 1993, el gobierno norteamericano propuso la creación de un sistema de cifrado para que las agencias del gobierno pudieran espiar comunicaciones cifradas cuando la ley así lo requiriera. Este propósito fue conocido como el sistema Clipper, que utilizaba un criptosistema simétrico llamado SKIPJACK. Este utiliza cifrado de bloques de 64 bits junto con claves de 80 bits

agregando mayor robustez a la clave, a diferencia de DES. Sin embargo, la especificación de SKIPJACK es confidencial y por lo tanto no está publicada.

Este algoritmo puede ser usado solamente en dispositivos de hardware autorizados. Por ejemplo: chips Clipper.

Algoritmos Propietarios

Se han definido una gran cantidad de algoritmos propietarios usados en productos comerciales tales como :

- ✓ IDEA(Algoritmo de Encripción de Datos Internacional) creado por Ascom-Tech en Suecia.
- ✓ RC2,RC4 y RC5 algoritmos creado por la empresa de Seguridad de Datos RSA.
- ✓ CAST creado por Norteen Telecom.

Algunos de estos algoritmos propietarios pueden ser buenos, pero es difícil para la comunidad que los utiliza poder llegar a coincidir en la robustez del algoritmo, dada su limitada exposición a una crítica por parte del público.

Criptografía asimétrica (Clave pública – Clave Privada)

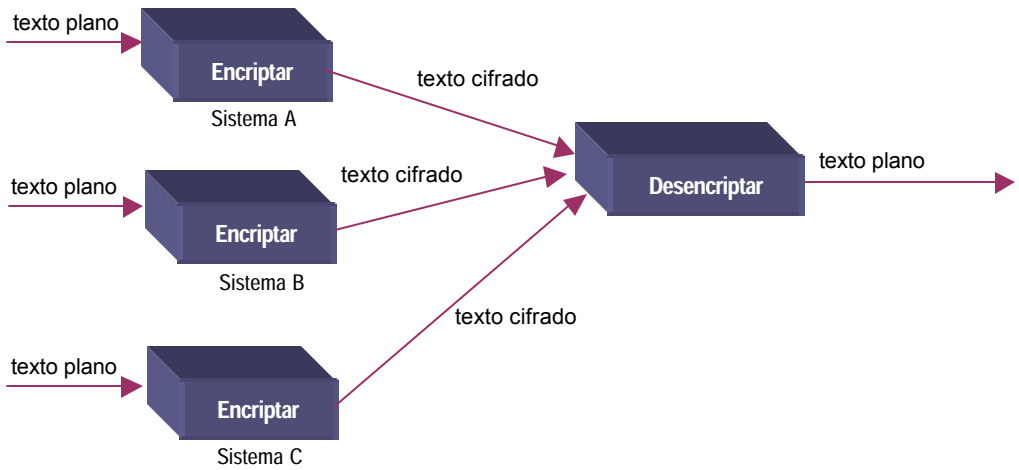
Como ya se ha mencionado anteriormente, el principal problema que presenta el uso práctico de la criptografía de clave simétrica es la distribución de las claves. La criptografía de clave asimétrica, sin embargo, usa claves diferentes para cifrar y descifrar un mensaje. Lo único que se transmite de un usuario a otro es el mensaje cifrado.

En 1976, Whitfield Diffie y Martin Hellman publican la idea de que cada usuario tenga dos claves: una pública (P_i, conocida por todos) y otra privada (S_i, sólo conocida por su dueño).

Existen dos modos en el cuál este tipo de criptosistema puede ser utilizado. Estos son: *modo encripción* y *modo autenticación*.

Cada parte involucrada en una transacción tiene una clave privada que sólo ella conoce y una clave pública asociada conocida por todos.

Si lo que se requiere es dotar de confidencialidad al mensaje, el autor utiliza la clave pública del destinatario para cifrar con ella los datos y luego el receptor utilizará su clave privada para recuperar el mensaje original. Dado que el mensaje cifrado solamente puede desencriptarse utilizando la clave privada asociada, queda garantizado que sólo el receptor podrá descifrar el mensaje. Este modo de uso de la criptografía asimétrica es conocido como **Modo Encripción.**

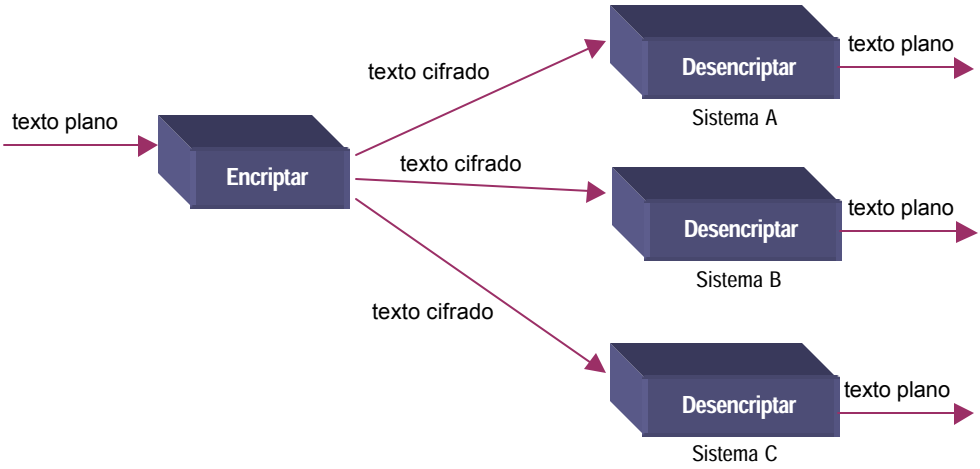


Criptosistema de Clave Pública – Modo Encripción

Supongamos un algoritmo de cifrado **E** y otro de descifrado **D** se aplican a un mensaje **M**, entonces debe cumplirse que :

$$\mathbf{D\ (\ E(M,P_i)\ ,\ S_i\ )\ =\ M}$$

Por otro lado, si lo que se requiere es autenticación de datos e integridad del mensaje, entonces el autor utiliza su clave privada para encriptar dicho mensaje. Luego, el receptor utilizará la clave pública asociada para recuperar el mensaje original. Aquí no se garantiza privacidad dado que la clave pública es conocida por todos y entonces cualquiera puede obtener el mensaje original, lo que se está garantizando aquí es que el mensaje realmente fue enviado por el remitente y la información contenida en el mensaje no fue adulterada por espías. Esto se cumple porque la clave privada es conocida únicamente por el emisor. Este modo de uso de la criptografía asimétrica es conocido como *Modo Autenticación*, el mismo proveen la base para la construcción de sistemas de firma digital.



Criptosistema de Clave Pública – Modo Autenticación

Supongamos un algoritmo de cifrado **E** y otro de descifrado **D**, aplicados a un mensaje **M**. Debe cumplirse que :

$$\mathbf{D\ (\ E(M,S_i)\ ,\ P_i\ )\ =\ M}$$

De acuerdo a lo expresado anteriormente, estamos en condiciones de afirmar que en la criptografía de clave pública existen dos puntos que son de suma importancia: el secreto de la clave privada y la disponibilidad de la clave pública.

Algunos ejemplos de este tipo de criptosistema son:

- ✓ *McElliece* (1978): basado en el problema de decodificación por códigos lineales generales
- ✓ *ElGamal* (1985): basado en el problema de log discreto.
- ✓ *DSS* (1992): basado en el problema de log discreto.

La ventaja entonces, cuando se usa alguna técnica de protección basada en criptografía de clave pública, es que no se requiere un canal seguro para que la clave sea transmitida, es decir que pueden trasmitirse datos sensibles a través de canales inseguros.

✳ RSA

El *algoritmo de RSA* fue inventado por Rivest, Shamir y Adleman en 1977. Este algoritmo trabaja de la siguiente forma:

Se escogen dos números primos grandes, **p** y **q**, y se calcula su producto **n** = **p** * **q**. A **n** se lo conoce como módulo.

También se elige un número **e** menor que **n** y primo a (**p**-1)(**q**-1) lo cual significa que e y (**p**-1)(**q**-1) no tienen factores comunes excepto el 1.

Luego se obtiene el número **d** tal que (**e.d** –1) es divisible por (**p**-1)(**q**-1).

Entonces **e** y **d** son conocidos como el exponente público y el exponente privado respectivamente.

La *clave pública* esta formada por el par **(n,e)** y la clave privada por el par **(n,d)**. Los factores p y q pueden guardarse con la clave privada o ser destruidos.

Resulta complejo obtener la clave privada **d** conociendo la clave pública **(n,e)**. Sin embargo, si alguien puede factorear **n** en **p** y **q**, entonces puede obtener la clave privada **d** a partir de la clave pública. La seguridad de RSA esta basada en la suposición de que dicha factorización es difícil. La técnica básica que se utiliza en los sistemas de clave pública es la factorización. Factorizar un número significa descomponerlo en producto de números primos y esta operación, cuando tratamos de números grandes, resulta un problema muy complicado.

Proceso de Cifrado y Descifrado utilizado por RSA

Suponiendo la clave pública **(n,e)** y la clave privada **(n,d)**. Generalmente se selecciona un exponente público pequeño, por ejemplo $\mathbf{e}=2^{16}+1$.

Entonces el *proceso de cifrado o encripción* es llevado a cabo haciendo el cálculo :

$$\mathbf{C} = \mathbf{M}^{\mathbf{e}} \pmod{\mathbf{n}}$$

Donde **M** es el texto plano tal que 0<=**M**<**n**.

El *proceso de descifrado o desencriptado* es realizado haciendo el cálculo:

$$\mathbf{M} = \mathbf{C}^{\mathbf{d}} \pmod{\mathbf{n}}$$

Luego, en caso de que el emisor quiera enviar un mensaje cifrado, calcula **C** y lo envía al destinatario. Dado que el receptor es el único que conoce **d** entonces solamente él podrá desencriptar el mensaje.

✳ El Gamal

Este algoritmo se basa en la dificultad de calcular algoritmos discretos.

Para generar un par de claves, se elige un número primo **p** y dos números aleatorios **g** y **x**, de modo que ambos sean más pequeños que p. Luego, se calcula:

$$\mathbf{y} = \mathbf{g}^{\mathbf{x}} \bmod \mathbf{p}$$

A partir de ello se obtiene la clave pública formada por **y**, **q** y **p**, y la clave privada representada por **x**.

✳ Algoritmo de curva elíptica (CCE)

Otro tipo de criptografía de clave pública es el que usa curvas elípticas definidas en un campo finito. La diferencia que existe entre este sistema y RSA es el problema en el cual basan su seguridad. RSA razona de la siguiente manera: te da el número 15 y el desafío es encontrar los factores primos. El problema en el cual están basados los sistemas que usan curvas elípticas que denotaremos como CCE es el problema del logaritmo discreto elíptico. En este caso su razonamiento con números sería algo como: te dá el número 15 y el 3 y el desafío es encontrar cuántas veces hay que sumar el mismo 3 para obtener el 15.

La principal ventaja que ofrece el CCE en comparación con RSA, es la longitud de la clave secreta. Se puede demostrar que mientras en RSA se tiene que usar una clave de 1024 para ofrecer una considerable seguridad, el algoritmo de curva elíptica sólo usa163 bits para ofrecer la misma seguridad. Así también, las claves RSA de 2048 bits son equivalentes en seguridad a 210 bits de CCE. Esto se debe a que para resolver el Problema del Logaritmo Discreto Elíptico el único algoritmo conocido toma tiempo de ejecución totalmente exponencial, mientras que el algoritmo que resuelve el Problema de Factorización Entera incluso también el Problema del Logaritmo Discreto en un campo finito toma tiempo subexponencial.

Los CCE están reemplazando a RSA en las aplicaciones que lo implementan, éstos definen también esquemas de firma digital, de intercambio de claves simétricas y otros.

Sistemas Híbridos (Intercambio de claves simétricas)

Dado que los algoritmos criptográficos de clave pública (como ser RSA) son excesivamente lentos, se han desarrollado sistemas híbridos que utilizan criptografía de clave simétrica y de clave pública.

El mecanismo, a grandes rasgos, es el siguiente:

1. **A** genera una clave aleatoria que servirá de clave a un algoritmo simétrico (por ejemplo, DES).
2. Esta clave es cifrada con la clave pública de **B** y luego enviada a **B** (criptografía de clave asimétrica)
3. **B** recibe el mensaje y con su clave privada procede a descifrar el mensaje, obteniéndose la clave para el algoritmo simétrico.
4. El resto de comunicaciones entre **B** y **A** se lleva a cabo usando algoritmos simétricos con la clave transmitida previamente.

Con este método híbrido se eliminan los problemas que origina la distribución de claves. Ahora bien, aparecen problemas nuevos, debidos fundamentalmente a la posibilidad de suplantar algunos de las partes involucradas en el intercambio. Este problema solamente puede evitarse intercambiando certificados en lugar de únicamente claves.

Algoritmo de intercambio de claves Diffie-Hellman

El acuerdo de claves es el método donde dos partes, sin previo conocimiento uno del otro, intercambian mensajes habiendo previamente acordado el uso de una clave conocida por ambos a través de un medio inseguro (como puede ser Internet). Esta clave puede ser almacenada usando un algoritmo de clave pública u otro método.

Se utilizan dos parámetros públicos, **p** y **g**.
p es primo y **g** es conocido como "generador". Este último se define como:

$$\forall \mathbf{n} \; \exists \; [1...(n-1)] \; \forall \mathbf{x} \; / \; \mathbf{n} = \mathbf{x}^{*}\mathbf{g} \pmod{\mathbf{p}}$$

Dada 2 partes **A** y **B** que intervienen en una operación, el funcionamiento del protocolo es:

1. **A** genera aleatoriamente un valor privado **a**.
2. **B** genera aleatoriamente un valor privado **b**.
3. **A** genera un valor público **g*****a** mod **p**.
4. **B** genera un valor público **g*****b** mod **p**.
5. **A** y **B** intercambian estos valores públicos.
6. Luego, **A** calcula $\mathbf{K_a} = (\mathbf{g}^*\mathbf{b} \bmod \mathbf{p})^*\mathbf{a} = \mathbf{g}^*\mathbf{a}^*\mathbf{b} \bmod \mathbf{p}$
7. **B** hace un cálculo similar $\mathbf{K_b} = (\mathbf{g}^*\mathbf{a} \bmod \mathbf{p})^*\mathbf{b} = \mathbf{g}^*\mathbf{a}^*\mathbf{b} \bmod \mathbf{p}$

En este momento ambos poseen una clave común **K**, tal que:

$\mathbf{K} = \mathbf{K_a} = \mathbf{K_b} = \mathbf{g}^*\mathbf{a}^*\mathbf{b} \bmod \mathbf{p}$

El fundamento de este algoritmo consiste en que es difícil calcular **K**, aún conociendo los valores públicos **p**, **g**, **g*****b** (mod **p**) y **g*****a** (mod **p**).

Como se ha mencionado anteriormente, el inconveniente principal de que adolece este algoritmo es la posibilidad de que un intruso intercepte las comunicaciones entre los dos interlocutores haciéndose pasar por uno de ellos. Para evitarlo, será necesario utilizar certificados.

En resumen

Haciendo una comparación, DES y otros cifradores de bloques son mucho más rápidos que RSA.

En software, DES es generalmente 100 veces más rápido que RSA. En Hardware, DES es entre 1.000 y 10.000 veces más rápido, dependiendo de la implementación.

En la práctica, el exponente público usado por RSA es más pequeño que el exponente privado; esto quiere decir que el proceso de verificación de la firma es más rápido que el proceso de firma. Este efecto es deseado ya que un mensaje puede ser firmado por una persona una única vez, pero la firma puede ser verificada muchas veces.

Actualmente, RSA es usado en una gran variedad de productos, plataformas e industrias en todo el mundo. Se la puede ver en muchos sistemas comerciales y esta planeado usarse en muchos más. Esta siendo incorporado en Sistemas Operativos como ser Microsoft, Apple, Sun y Novell. En lo que concierne a hardware, se puede encontrar RSA en seguridad telefónica, en placas de red y Smart Cards. Además, RSA está incorporado dentro de la mayoría de los protocolos para comunicaciones seguras sobre Internet incluyendo S/MIME [ver sección “S/MIME” del Capítulo 7] SSL [ver sección “SSL” del Capítulo 7] y S/WAN [ver sección “SSL” del Capítulo 7].

Estandares PKC

Dada la gran aceptación y uso de la criptografía de clave pública en Internet, se pensó en que los vendedores de productos que utilizan esta tecnología se pudieran poner de acuerdo con respecto a técnicas básicas de clave pública y compatibilidad entre implementaciones. La interoperabilidad requiere la adhesión estricta a un formato estándar con el que se hayan puesto de acuerdo. Estos estándares son conocidos como los *Estándares de Criptografía de Clave Pública (PKCS)*. Son un conjunto de documentos que contienen especificaciones sobre seguridad. Estos forman la base para la interoperabilidad de distintos productos en Internet.

Los *PKCS* describen la sintaxis de los mensajes de una manera abstracta y dan detalles completos sobre los algoritmos criptográficos. Entre dichos algoritmos se encuentra RSA. Estos estándares usados por los desarrolladores de sistemas que utilizan tecnologías de clave pública, son provistos por los Laboratorios RSA.

En el área de seguridad, la Notación de Sintaxis Abstracta (ASN.1) ha sido muy utilizada para describir protocolos de seguridad, interfaces y definiciones de servicios.

Los estándares PKCS son escritos en forma abstracta con un mínimo de notación, por ejemplo la sintaxis de los mensajes está escrita en ASN.1. Como ejemplo, aquí se muestra la definición del mensaje SignedData:

```
SignedData ::= SEQUENCE {
        version             Version,
        digestAlgoritm      DigestAlgoritmIdentifiers,
        contentInfo         ContentInfo,
        certificates        [0] IMPLICIT ExtendedCertificatesAndCertificates
                            OPTIONAL,
        crls                [1]IMPLICIT CertificateRevocationLists
                            OPTIONAL,
        signerInfos         SignerInfos
}
```

Actualmente, hay 12 miembros de la familia PKCS completamente definidos. Sin embargo, algunos de ellos fueron combinados con otros. Por ejemplo, el PKCS#2 y PKCS#4 han sido incorporados al PKCS#1.

PKCS #1: *RSA Encryption Standard*

Describe un método llamado *rsaEncription* para cifrar datos usando el criptosistema de clave pública RSA. Se usa en la construcción de la firma digital y el ensobrado digital.

También describe la sintaxis de las claves pública y privada. La sintaxis de la clave pública es usada en certificados mientras que la sintaxis de la clave privada es generalmente usada en claves privadas encriptadas (ver PKCS#8).

La sintaxis de la clave pública es idéntica tanto para X.509 como para PEM.

PKCS#1 también define tres algoritmos de firma llamados md2WithRSAEncryption, md4WithRSAEncryption y md5WithRSAEncryption. Estos algoritmos pueden ser usados para firmar certificados X.509/PEM, listas de revocación de certificados y extensiones de certificado PKCS#6.

PKCS #3: *Diffie-Hellman Key Agreement Standard*

Describe un método para la implementación de acuerdo de claves Diffie-Hellman, donde dos partes pueden ponerse de acuerdo acerca de una clave secreta, la cual es conocida sólo

por ellos. Entonces, esta clave privada puede ser usada, por ejemplo, para cifrar comunicaciones entre las partes.

La aplicación de este estándar está orientada a protocolos para el establecimiento de comunicaciones seguras, tales como los propuestos por OSI para los niveles de transporte y de red (por ejemplo: ISO90a, ISO90b).

PKCS #5: *Password-Based Encryption Standard*

Describe un método para la encripción de un octeto con una clave privada derivada de una palabra clave. El resultado de este método es una cadena de ocho caracteres. Sin embargo, este estándar puede ser usado para encriptar cualquier cadena de ocho caracteres. Su aplicación principal para la criptografía de clave pública es en la encripción de las claves privadas cuando son transferidas desde un sistema a otro como es explicado por PKCS #8. Este standard define dos algoritmos de encripción de claves: pbeWithMD2AndDES-CBC y pbeWithMD5AndDES-CBC. Estos algoritmos usan encripción de clave privada DES en modo de cadenas de bloques cifrados, donde la clave secreta es derivada de una palabra clave con el algoritmo de hash MD2 o MD5 [ver sección "Mecanismos de uso de la Firma Digital" del Capítulo 5].

PKCS #6: *Extended-Certificate Syntax Standard*

Describe la sintaxis usada en las extensiones de certificados. Una extensión consiste de un conjunto de atributos pertenecientes a un certificado de clave pública X509.

La intención de incluir atributos a un certificado es para extender el proceso de certificación además de incluir otra información acerca de la entidad poseedora del certificado, tal como la dirección de correo electrónico. El estándar PKCS#9 provee una lista exhaustiva de los posibles atributos que puede tener un certificado [ver sección "Certificados" del Capítulo 4].

PKCS #7: *Cryptographic Message Syntax Standard*

Describe una sintaxis general para los datos a los cuales se les pudo haber aplicado un proceso criptográfico, tal como las firmas y ensobrados digitales. La sintaxis admite recursión, por lo que, por ejemplo un sobre puede ser puesto dentro de otro, o una parte puede firmar algo previamente ensobrado. Esto permite atributos arbitrarios tales como la hora en que se firmó para ser autenticado con el contenido del mensaje, y provee otros atributos tales como la cantidad de firmantes para ser asociado con una firma.

Este estándar es compatible con PEM, esto significa que el contenido del dato firmado y el dato encriptado y ensobrado, construido de un modo PEM-compatible, puede ser convertido en mensaje PEM sin necesidad de aplicarle ninguna operación criptográfica. Los mensajes PEM también puede convertirse en datos firmados o datos firmados y encriptados de una manera similar.

PKCS #8: *Private-Key Information Syntax Standard*

Describe una sintaxis para la información de la clave privada. Esta información incluye una clave privada para algunos algoritmos de clave pública y un conjunto de atributos. Este estándar también describe la sintaxis para las claves privadas encriptadas.

Un algoritmo de encripción basado en "palabra clave" puede ser usado para encriptar la información de la clave privada.

La intención de incluir un conjunto de atributos es la de proveer una forma simple de que un usuario pueda confiar en la información, tal como un nombre distinguido (DN) [ref. 1] o la clave pública de la Autoridad de Certificación (CA) [ver sección "Autoridades de Certificación" del Capítulo 4]. Esta confianza también puede ser provista a través de una firma digital.

PKCS #9*: *Selected Attribute Types

Define los atributos a ser usados por las extensiones de los certificados del PKCS #6, por los mensajes firmados del PKCS #7 y por la información de clave privada del PKCS #8.

PKCS #10*: *Certification Request Syntax Standard

Describe la sintaxis para el requerimiento de un certificado. Dicho requerimiento está formado por un nombre único (DN), una clave pública y, opcionalmente, un conjunto de atributos, los cuales son firmados por la entidad que requiere la certificación. Cada requerimiento es enviado a una CA, la cual transforma el mismo en un certificado de clave pública X509 o en un certificado extendido PKCS #6.

La intención de incluir un conjunto de atributos es por dos razones: para proveer otra información acerca de dicha entidad y para proveer atributos para un certificado extendido.

PKCS #12:

Especifica un intercambio de formatos diseñado para transferir certificados y claves privadas almacenadas, entre computadoras. Por ejemplo para transferir claves desde Netscape Communicator a Microsoft Internet Explorer.

El ***PKCS#10*** y el ***PKCS#7*** son muy usados en el mercado de la seguridad, se encuentran embebidos en aplicaciones tales como Netscape Communicator y Microsoft Internet Explorer. También son muy usados en la administración de claves y en el punto de inicialización de confianza de SSL [ver sección "SSL" del Capítulo 7], correo electrónico seguro S/MIME [ver sección "S/MIME" del Capítulo 7] y en las tecnologías de codificación de firma como Autenticode[1].

La notación ASN.1 para el requerimiento de un certificado es la siguiente:

```
CertificationRequest ::= SEQUENSE {
        certificationRequestInfo        CertificationRequestInfo,
        signatureAlgorithm              SignatureAlgorithmIdentifier,
        signature                       Signature
}
CertificationRequestInfo ::= SEQUENCE {
        version                 Version,
        subject                 Name,
        subjectPublicKeyInfo    SubjectPubicKeyInfo,
        attributes              [0] IMPLICIT Attributes
}

Version :: INTEGER
Attributes ::= SET OF Attribute
SignatureAlgorithmIdentifier ::= AlgorithmIdentifier
Signature ::= BIT STRING
```

El ***PKCS #7*** es considerado más complejo que ***PKCS#10***, sin embargo el objetivo de ambos es muy simple. Consiste en especificar la generación del tipo *signed-data* como un mecanismo standard de seguridad para la creación de la firma digital:

```
SignedData ::= SEQUENCE {
    version             Version,
    digestAlgoritm      DigestAlgoritmIdentifiers,
    contentInfo         ContentInfo,
```

[1] Authenticode: es un tipo de firma digital que permite a los desarrolladores firmar digitalmente su código de software.

```
        certificates            [0]IMPLICIT ExtendedCertificatesAndCertificates
                                OPTIONAL,
        crls                    [1]IMPLICIT CertificateRevocationLists
                                OPTIONAL,
        signerInfos             SignerInfos
    }

    SignerInfo ::= SEQUENCE {
        version                 Version,
        issuerAndSerialNumber   IssuerAndSerialNumber,
        digestAlgorithm         DigestAlgorithmIdentifier,
        authenticatedAttributes [0] IMPLICIT Attributes OPTIONAL,
        digestEncryptionAlgorithm DigestEncryptionAlgorithmIdentifier,
        encryptedDigest         EncryptedDigest,   -- la firma
        unauthenticatedAttributes [1] IMPLICIT Attributes OPTIONAL,
    }

EncryptedDigest ::= OCTET STRING   -- la firma RSA del resumen


    EnvelopedData ::= SEQUENCE {
        version                 Version,
        recipientInfos          RecipientInfos,
        encryptedContentInfo    EncryptedContentInfo,
}

    RecipientInfo ::= SEQUENCE {
        Version                 Version,
        issuerAndSerialNumber   IssuerAndSerialNumber,
        keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
        encryptedKey            EncryptedKey
}
```

Infraestructura de clave pública

El amplio crecimiento de la tecnología de clave pública requiere una infraestructura de clave pública que la soporte. La misma consiste en un conjunto de protocolos, servicios y estándares que otorgan soporte a aplicaciones de clave pública.

El término *PKI (Public Key Infraestructure)* se refiere, a veces, simplemente, a la jerarquía de confianza basada en el uso de certificados de clave pública [ver sección "Certificados" del Capítulo 4]. Una visión intermedia, en cambio, es que la PKI incluye servicios y protocolos de manejo de clave pública a través de autoridades de certificación (Certification Authority o CA) y autoridades de registración (Registration Authority o RA), pero no provee necesariamente operaciones criptográficas con las claves. La definición más completa entiende a la PKI como los servicios de firma digital y criptografía provistos a las aplicaciones de usuario final.

En el presente trabajo nos referiremos a la PKI en su más amplio concepto.

Además de definir los protocolos, servicios y estándares, haremos incapié en el rol de las autoridades de certificación, las estructuras entre múltiples autoridades de certificación, los métodos para descubrir y validar caminos de certificación, las herramientas interoperables y la legislación subyacente.

El diseño de una infraestructura de clave pública tiene en cuenta determinados requerimientos:

1. Necesita ser escalable. Dicha escalabilidad es indispensable para permitir economizar el desarrollo y la operación de las mismas, así como para brindar seguridad en las comunicaciones con subsistemas remotos.
2. Debe proveer soporte para múltiples aplicaciones.
3. Debe ser posible la interoperabilidad entre infraestructuras administrativamente separadas, dado que no resulta adecuado contar con una Infraestructura operada por una única administración universal.
4. Debe contar con soporte para múltiples políticas, ya que no todas las aplicaciones tienen las mismas necesidades, por ello las cadenas (o caminos) de certificación válidos para unas pueden no serlo para otras. Es conveniente entonces que cada camino de certificación reconozca diferentes políticas como aceptables.
5. Debe tener la capacidad de manejar los riesgos asociados a sus funciones y los riesgos existentes entre las partes involucradas.
6. Es necesario que las responsabilidades de cada CA estén claramente limitadas.
7. Deben establecerse estándares en el campo de dichas infraestructuras. Los mismos deben referirse tanto a aspectos legales como técnicos.

Su función:

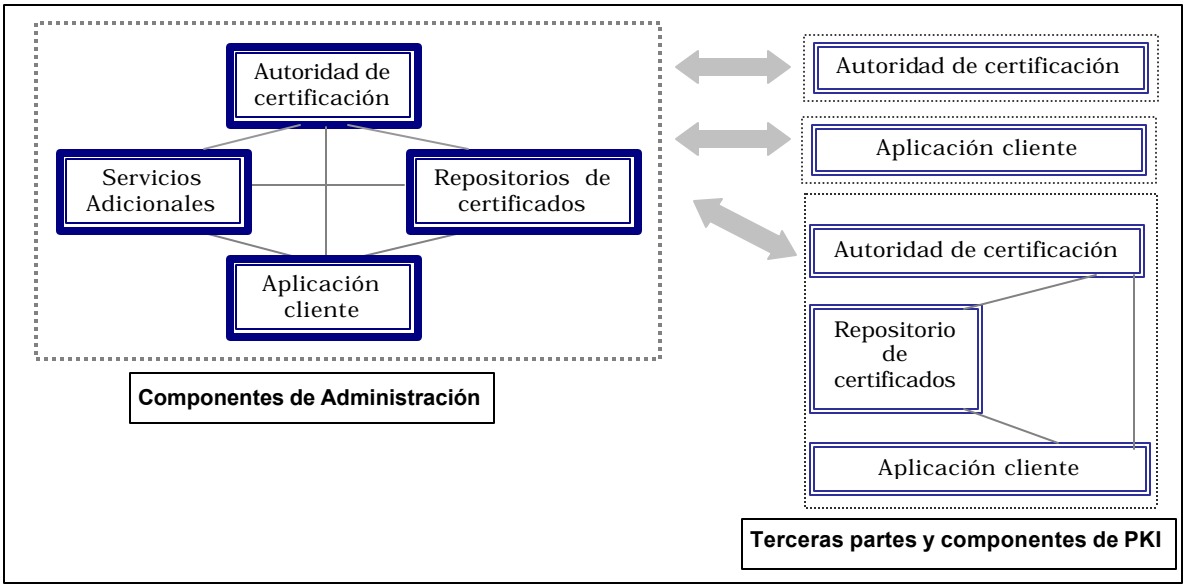
El *propósito de una infraestructura PKI* es proveer claves y manejo de certificados confiables y eficientes, para lograr la habilitación de la autenticación, la no repudiación y la confidencialidad.

Los sistemas basados en clave pública deben estar seguros de que cada vez que ellos confían en una clave pública, la clave privada asociada pertenece al sujeto con el cual ellos se están comunicando. Esta confianza se basa en el uso de certificados de clave pública los cuales son estructuras de datos que ligan valores de clave pública a los sujetos. La ligadura se realiza a través de una autoridad de certificación, la cual verifica la identidad del sujeto y firma digitalmente cada certificado.

Su Arquitectura:

Una *PKI* consta de cinco componentes fundamentales:

1. *Autoridades de certificación*, que emiten y revocan certificados.
2. *Autoridades de registración*, que atestiguan la asociación entre la clave pública y la entidad propietaria del certificado.
3. *Poseedores de los certificados emitidos* que pueden firmar documentos digitales.
4. *Entidad final*, usuarios del certificado PKI y/o usuarios del sistema que son sujeto de un certificado.
5. *Repositorios* que guardan y hacen disponibles tanto certificados como listas de revocación de certificados.



Interoperabilidad PKI

En cuanto a los protocolos definidos por esta infraestructura, podemos agrupar los mismos en dos grandes conjuntos:

- ✓ Los protocolos operacionales, los cuales se relacionan con el requerimiento y la emisión de certificados y CRLs [ver sección "Listas de revocación" en el Capítulo 4].
- ✓ Los protocolos de manejo, que tienen incidencia sobre las interacciones (en línea y fuera de línea) dadas entre los diferentes componentes PKI. Incluyen reglas para los procesos de registración, incialización, certificación, revocación y recuperación de claves.

Algunas ventajas de la Infraestructura PKI

- ✓ Las claves no viajan a través de la red desde el cliente al servidor, dado que los certificados constituyen información pública.
- ✓ Ofrece mejores medios para identificar al usuario ya que bs certificados contienen información verificable relacionada con la identidad del usuario, lo cual no ocurre en la autenticación basada en dirección IP del equipo del usuario, en nombre de dominio o en dirección de mail, dado que las direcciones IP pueden ser dinámicas, y los nombres de dominio y direcciones de mail pueden ser espiadas...

- ✓ Los usuarios pueden conectarse a diferentes servidores de una Intranet logueandose una sola vez.
- ✓ Los certificados basados en tecnología de clave pública proveen un mecanismo de autenticación más fuerte. Sólo el usuario conoce la forma de acceder a su clave privada.
- ✓ Simplificación en la administración y disminución de costos.

Algunas debilidades de la Infraestructura PKI

Existen algunos cuestionamientos que se realizan a esta infraestructura:

- ✓ La seguridad de un sistema basado en CA consiste en varios eslabones, algunos de los cuales no son criptográficos: la gente también forma parte de esta infraestructura.
- ✓ Un conflicto que permanece sin resolver es la seguridad de la clave privada con respecto al almacenamiento de la misma. Que un empleado guarde su clave privada en una PC resulta un hecho riesgoso partiendo de la base de que esa PC puede ser utilizada por varios miembros de la oficina. Como solución surge la aparición de las Smart Cards[1] y entonces ahora la seguridad de la clave privada pasará a depender de la seguridad de estos dispositivos. No puede garantizarse el hecho de que una clave privada será únicamente utilizada por su dueño
- ✓ La CA puede ser una autoridad en cuanto a la creación de certificados pero no estamos en condiciones de afirmar que sea una autoridad con respecto a lo que los certificados contienen.

[1] Smart Card: Es un 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.

Certificados

Dado que el comercio electrónico a través de Internet crece día a día, es necesario que exista una arquitectura que garantice la autenticación y la no repudiación en forma eficiente y real, ante la ocurrencia de transacciones entre extraños, los cuales no se relacionaron nunca antes de dicha transacción.

Con el fin de que dichos servicios de seguridad sean garantizados, cada parte involucrada cuenta con un par de claves pública y privada. Dado que dicho par tiene una asociación no intrínseca con una persona entidad (es simplemente un par de números) es necesario contar con un mecanismo que los vincule.

El hecho de que un usuario utilice una clave pública equivale a decir que el mismo emite la declaración: "Las firmas que se verifican con esta clave pública me pertenecen". Sin embargo, otras partes que estén negociando con dicho usuario pueden tener alguna buena razón para no aceptar dicha declaración.

Una parte que confía en una declaración como la anteriormente mencionada en un sistema abierto corre el riesgo de estar confiando en un impostor, o de intentar refutar una negación de una firma digital que no ha sido realizada por el verdadero firmante.

Para solucionar estos problemas existe una entidad llamada autoridad de certificación, la cual emite un certificado, que es un registro electrónico que contiene la clave pública del dueño del certificado y confirma que el mismo es poseedor de la correspondiente clave privada.

Una de las principales funciones de un certificado es ligar un par de claves a una entidad particular. Dicha entidad puede ser una persona, un dispositivo de hardware o un proceso de software. Si el sujeto[1] del certificado es una entidad legal, tal como una persona, entonces se lo conoce como suscriptor.

El certificado de clave pública del usuario, emitido por la CA, contiene el nombre de usuario, la clave pública y otra información de identidad como ser dirección de mail, período de validez, etc.

Para asegurar la autenticidad de la identidad y del contenido del certificado, la autoridad de certificación que lo emite firma el mismo digitalmente. La firma digital de la CA en el certificado puede ser verificada usando la clave pública de la CA que lo avala.

Es esencial comprender que el nivel de confianza que el certificado provee, respecto de la identidad del usuario al cual representa, depende en absoluto del nivel de confianza existente hacia la Autoridad de Certificación que lo haya emitido.

Los certificados pueden ser publicados en directorios públicos o pueden hacerse dsponibles mediante otros medios, permitiendo así que los mismos sean recuperados a fin de verificar firmas o de encriptar documentos. La validez de una firma incluirá, en consecuencia, la validez del certificado de clave pública involucrado. La verificación será efectuada con mayor o menor rigor dependiendo del contexto en el cual se lleve a cabo.

Función de los certificados

Las características recién mencionadas constituyen las razones por las cuales los certificados digitales proveen una alternativa para que una persona pueda obtener la clave pública de otra. Este modelo escalable se basa en la confianza en una tercera parte, la Autoridad de Certificación, la cual autentica las personas y certifica sus claves públicas, entre otras de sus funciones.

[1] Se conoce como sujeto a la entidad asociada con el certificado.

Publicación de certificados

La publicación de certificados como modelo de distribución de claves públicas es escalable dado que un grupo reducido de Autoridades de Certificación puede autenticar un gran número de usuarios. La forma instaurada para obtener determinada clave pública tiene una gran similitud con el mecanismo de búsqueda de determinado número telefónico en una Guía Telefónica.

El servicio de publicación de certificados puede ser también provisto por la CA. El mismo tiene como objetivo hacer posible que el sistema que usa certificados tenga acceso al repositorio de certificados y pueda obtener un certificado de clave pública. Por ejemplo: una persona puede usar ese servicio para ver el certificado de otra persona, incorporar dicho certificado a una aplicación de correo electrónico y enviarle a la misma un mensaje cifrado.

Una autoridad de certificación puede distribuir certificados eligiendo alguno de los métodos existentes, tales como servicio de directorios o distribución vía correo electrónico.

Procesamiento del certificado en el cliente

Un certificado digital es una estructura de datos pública que puede contener información de un dominio específico. El software Cliente puede determinar la autenticidad de un certificado y usar luego la información específica codificada en el mismo para decidir su comportamiento.

Un certificado puede contener toda la información necesaria para guiar el procesamiento en el software Cliente. Es por ello que existe la posibilidad de construir sistemas escalables y menos vulnerables a fallas dado que una gran variedad de información puede ser codificada en el certificado, como ser privilegios de acceso, de uso de recursos, etc.

Certificados X.509

Es fundamental que exista el requerimiento de un formato elaborado para dichos certificados de clave pública, dado que un certificado puede contener otros datos además de la clave pública del sujeto y su identidad. Un certificado para correo electrónico seguro debe contener la dirección de correo electrónico del suscriptor además de su clave pública.

Es imprescindible entonces un formato estandarizado para permitir que las aplicaciones que utilizan certificados puedan interoperar con certificados emitidos por terceras partes. De esta manera, los usuarios finales no están atados a requerir certificados de una CA en particular, y los desarrolladores de software pueden escribir código genérico que trabaje con la mayor parte de los certificados emitidos.

El X.509, Sector de Estandarización de la ITU-T[2], junto a la Comisión Electrotécnica (IEC) de la ISO, publicó en 1988 el formato estándar de los certificados como parte de las recomendaciones del Directorio X.500.

La versión 1 del formato de certificados X.509 (v1) fue extendida en 1993 incorporando dos nuevos campos con el fin de soportar control de acceso al servicio de directorio, dando como resultado la versión 2 del formato X.509 (v2).

Más tarde, la experiencia ganada en el intento de desarrollar el protocolo PEM para asegurar la privacidad en el mail, el cual se basó en los formatos v1 y v2, reveló la deficiencia de los mismos en cuanto a ser formatos abiertos y extensibles. Como resultado de las revisiones correspondientes se procede a permitir campos para extensiones adicionales. Surge así la versión 3 de certificados X509, publicada en junio de 1996.

[2] ITU-T: International Telecommunication Union-Telecommunication .

Detalles de los estándares: X-509 V1, X-509 V2, X-509 V3, PEM, S/MIME.

Certificados X-509 V1 y X-509 V2

Los certificados X509 definen un estándar para el formato los certificados de clave pública y los campos por los cuales los mismos están compuestos.

Campos de un certificado X509 v1 y X509 v2.. Extensiones.

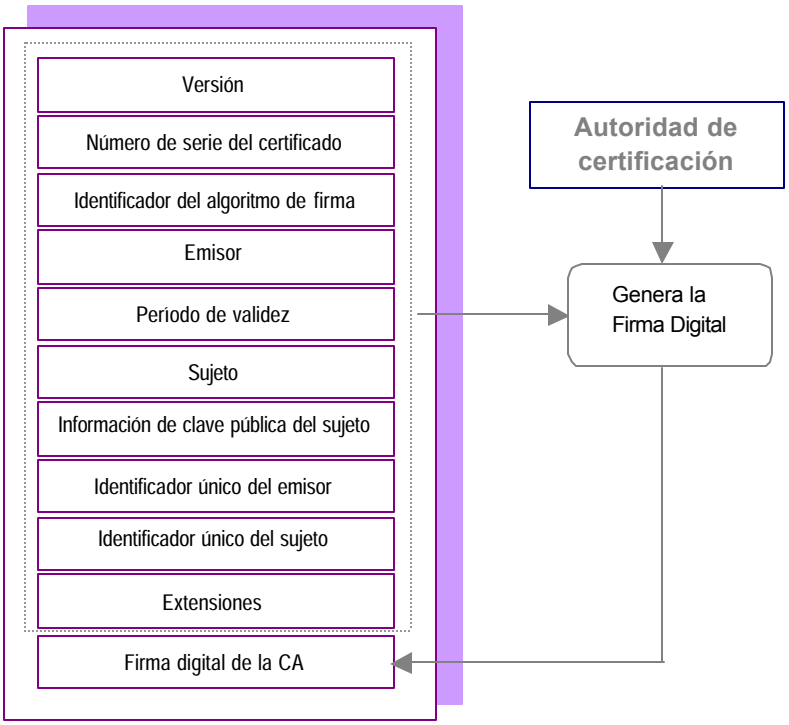
Los campos que componen un certificado X509 versión 1 y X509 versión 2 son:

- ✓ *Versión*: Este campo indica la versión del formato del certificado (1, 2 o 3). Están previstas versiones futuras.
- ✓ *Número de serie*: Este campo especifica el único identificador numérico del certificado en el dominio de todos los certificados de clave pública emitidos por la autoridad de certificación, es decir 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 al algoritmo usado por la CA para firmar el certificado. El identificador del algoritmo es un número registrado con el reconocimiento de una organización estándar, especifica tanto el algoritmo de clave pública como el de hashing.
- ✓ *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

Esta versión introduce un mecanismo en el cual los certificados pueden ser extendidos de una manera estandarizada y genérica, para incluir información adicional. Hay numerosas razones por las cuales dicha información adicional es requerida.

Por ello existen extensiones que se añaden a los campos de la versión anterior. Los certificados no están limitados sólo a las extensiones estándar y cualquiera puede registrar una extensión con las autoridades que corresponda. Es importante destacar que el mecanismo implementado para el manejo de las extensiones es completamente genérico.



Formato del certificado X.509 v3

Cada extensión consiste en tres campos: tipo, indicador de criticidad y valor.

El campo *tipo de la extensión* define el tipo de dato que contiene el valor del campo. El tipo puede ser: una cadena de texto, un valor numérico, etc. Para promover la interoperabilidad, todos los tipos de las extensiones deben ser registrados en una organización de estándares reconocida internacionalmente.

El campo *criticidad* es un flag de un solo bit. Cuando la extensión está indicada como crítica, esto indica que el valor de extensión asociado contiene información de tal importancia que debe ser reconocida. En caso contrario, la misma puede ignorarse. Si un certificado contiene una extensión crítica que no pueda ser reconocida, entonces el certificado debe descartarse.

Hay una distinción importante entre una extensión crítica y la información requerida en un certificado. Una aplicación particular puede requerir que cierta extensión esté disponible en algún certificado procesado por la aplicación. Esto sin embargo no significa que la extensión necesite caratularse como crítica. Las extensiones críticas son únicamente aquéllas reservadas para información tan importante que los datos deben ser interpretados por todas las aplicaciones (por ejemplo puede ser información crítica aquélla que previene el mal uso o el uso inseguro de un certificado). Consecuentemente la mayoría de las extensiones son consideradas críticas. Debe analizarse de manera muy cuidadosa cuando se agrega una extensión crítica a un certificado, ya que esa acción puede llegar a crear problemas de interoperabilidad con otros dominios de CA y con otras aplicaciones.

El campo *valor* contiene los datos actuales para la extensión.

Las extensiones estándar para los certificados de clave pública pueden ser divididas 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 (International Telecommunication Unit) e ISO/IEC (International Organization for Standarization/International Electrotechnical Comisión) han desarrollado y publicado un conjunto de extensiones estándares en una versión corregida del estándar X.509 versión 3. Las mismas son:

- ✓ Límites básicos: Indica si el sujeto del certificado es una CA y la longitud máxima de la cadena de certificación.
- ✓ Política del certificado: Este campo 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 distintos 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, sólo cifrado, sólo descifrado.

Notación y codificación ASN.1

Los formatos de datos X.509, así como ocurre generalmente con los protocolos X.500, son expresados en una notación conocida como ASN.1 [ref. 1], la cual fue desarrollada como parte del programa de estandarización del Modelo OSI (Open System Interconnection).

Esta poderosa notación no es tan popular en algunos ámbitos debido a su complejidad, a la falta de disponibilidad de sus especificaciones y a la existencia de herramientas de implementación de bajo costo.

Sin embargo, como ASN.1 es usado en una gran variedad de protocolos, quienes implementan productos comerciales invierten el esfuerzo necesario para familiarizarse rápidamente con la notación y adquirir herramientas apropiadas.

Revocación de certificados

Cuando se emite un certificado, se espera que esté en uso durante su período de validez completo. Sin embargo, varias circunstancias pueden causar que un certificado se torne inválido antes de que su período de validez termine. Tales circunstancias incluyen cambio de nombre, cambio de asociación entre el sujeto y la CA (por ej., un empleado termina su empleo en una organización), y compromiso o sospecha de compromiso de la clave privada correspondiente.

Por ello es necesario contar con algún mecanismo que permita mantener información de estado de los certificados durante su período de validez y que provea, al mismo tiempo, información de revocación a los sistemas que utilizan dichos certificados.

Existen dos opciones para la implementación de tal funcionalidad:

- ✓ Las listas de revocación [ver sección “Listas de Revocación” del Capítulo 4].
- ✓ Los mecanismos que implementan el OCSP (On-line Certificate Status Protocol).[ver sección “OCSP” del Capítulo 4]

Es importante también hacer mención al hecho de que las claves privadas deben ser guardadas en forma segura para no estar expuestos al compromiso resultante de la pérdida u olvido de la misma.

Las medidas de seguridad que se toman para proteger la clave privada deben ser, al menos, equivalentes a las requeridas para los mensajes cifrados con dicha clave. En general, una clave privada no debe guardarse jamás en formato de texto plano.

El mecanismo de almacenamiento más simple es cifrar la clave privada con una contraseña y guardar el resultado de dicha operación en el disco. Una mejor medida sería almacenar la clave en una computadora que no sea accedida por otros usuarios o en un medio removible que el usuario lleve consigo y se encargue de proteger.

Las claves privadas son también guardadas en hardware portable, como una Smart card. Los usuarios con necesidades de extrema seguridad, tales como los operadores de autoridades de certificación, pueden usar dispositivos “tamper-resistance”(resistentes a la falsificación) para proteger sus claves privadas.

Autoridades de Certificación

Las organizaciones responsables de emitir certificados digitales se conocen como Autoridades de Certificación. La función principal de las mismas consiste en verificar la identidad de la persona (física o jurídica) que solicita el certificado para luego emitir el mismo.

Dichas organizaciones proveen los siguientes servicios de manejo de certificados:

- ✓ Comprobación de la autenticidad de las personas para las cuales se emiten y se revocan dichos certificados.
- ✓ Emisión, revisión y publicación de certificados.
- ✓ Entrega, almacenamiento y archivo de certificados y listas de revocación [ver sección "Listas de Revocación" del Capítulo 4].

La CA firma digitalmente los certificados que emite con el fin de asegurar integridad de los datos contenidos y la identidad del usuario.

El certificado puede ser publicado en un repositorio o puede hacerse disponible mediante otros medios. Los repositorios son bases de datos en línea que almacenan certificados y otra información disponible para recuperar y usar en el proceso de verificación de firmas digitales.

Si el suscriptor pierde el control de la clave privada, el certificado se torna no confiable y la CA puede suspender o revocar el mismo. La CA debe publicar la noticia de revocación o suspensión del certificado.

El único requerimiento es que la autoridad de certificación sea de confianza tanto para el firmante como para el receptor de la información firmada.

La confianza de los usuarios en la CA es fundamental para el buen funcionamiento del servicio. La seguridad física de la CA debe ser muy fuerte en particular en lo que respecta a la protección de la clave privada que utiliza para firmar sus emisiones.

Las CAs junto con la gestión de los certificados constituyen la base de la Infraestructura de Clave Pública conocida también como Infraestructura PKI.

En general, la confiabilidad requerida para los certificados emitidos por autoridades de certificación, difiere de acuerdo a las aplicaciones en las cuales los mismos estén siendo usados, y los requerimientos que se exigen tienen que ver con que el manejo, la operatividad, el sistema y las facilidades requeridas que poseen las autoridades de certificación en cuanto al control de la emisión y a la renovación de certificados son también diferentes.

Obligaciones de las CAs

Una autoridad de certificación tiene la obligación y la responsabilidad de asegurar la confiabilidad y la seguridad de la certificación. Sin embargo la confiabilidad y la seguridad no pueden ser aseguradas sólo por la autoridad de certificación, las demás partes involucradas en el proceso de certificación también deben cumplir con ciertas reglas a la hora de participar en dicho proceso.

Además la autoridad de certificación debe cumplir con las políticas que se determinan en los requerimientos de manejo, de operación, de sistema y de facilidades, garantizando de esta forma la confiabilidad y la seguridad en la CA misma.

Para que tanto la autoridad de registración [ver sección "Autoridades de Registración" del Capítulo 4] como el repositorio resulten confiables y seguros, la autoridad de certificación debe asegurar que las organizaciones externas observan las políticas determinadas por ella, para ser consistentes con la misma en lo que se refiere a garantías de seguridad y confiabilidad.

Otra obligación de gran importancia que debe cumplir una CA es la divulgación de información apropiada a los suscriptores y a las partes que confían en los certificados que ella emite.

También existen obligaciones que deben ser cumplidas por el suscriptor del certificado. Entre las mismas podemos mencionar las siguientes:

- ✓ El suscriptor debe presentar la información precisa a la autoridad de certificación y a la autoridad de registración cuando solicita un certificado. Así como también debe chequear que la información descriptiva contenida en el certificado emitido por la CA sea certera.
- ✓ Es necesario que proteja la clave privada generando el par de claves pública/privada usando hardware y software confiable. Luego debe prevenir que la clave privada sea espiada por otras personas.
- ✓ El suscriptor debe tomar prontas acciones de revocación en el caso de que exista compromiso o robo de la clave privada o en el caso de que la información contenida en el certificado haya cambiado significativamente.

No debemos olvidar que la parte que confía debe determinar cuando recibir o no un certificado es apropiado para su propósito. En caso de recibir el certificado, es su obligación comprobar si el mismo es un certificado válido, verificando su propósito de uso, su período de validez y la procedencia de la firma. Además debe chequear si existe revocación para dicho certificado.

Responsabilidades de las CAs

La autoridad de certificación debe determinar sus deberes así como aquellos de las personas que solicitan o utilizan certificados. Debe establecer y divulgar sus políticas concernientes a los compromisos y las garantías involucradas en el proceso de certificación.

La autoridad de certificación debe publicar sus políticas relevantes en su CPS (Certification Practice Statement) [ref. 1] y asegurar que todos los ítems importantes que no estén allí hayan sido adecuadamente divulgados.

Cada autoridad de certificación debe definir en su CPS su nivel de responsabilidad y la compensación que le compete frente a pérdidas que son consecuencia de haber quebrantado sus obligaciones. Algunos casos, entre otros, en los cuales la CA se reconoce responsable son:

- ✓ Cuando las organizaciones de regulación especificadas por la CA han sido violadas a través de actividades criminales dentro de la CA.
- ✓ Cuando se violan los requerimientos de control de certificados (por ej. Se emite un certificado incompleto, por error en el procedimiento de registración, que causa que el usuario sufra pérdidas).
- ✓ Cuando se violan los requerimientos de control de revocación: la información de revocación provista por el suscriptor no fue correctamente registrada en una lista de revocación y el certificado fue subsecuentemente aceptado por un usuario como un certificado efectivo a través de la referencia a la lista de revocación, causando pérdidas a dicho usuario.

Divulgación de la información (CPS)

La CA debe divulgar información adecuada incluyendo información administrativa, información técnica e información operacional, excepto que la divulgación pueda comprometer la seguridad de la CA.

Entre los principales fines de esta divulgación podemos mencionar:

- ✓ Permitir a los usuarios confirmar la firmeza administrativa de la CA, dado que los datos administrativos, incluyendo los datos de status financiero, deben ser revelados.
- ✓ Hacer posible que los usuarios puedan juzgar las prácticas de negocios de la CA, ya que las mismas deben ser publicadas en intervalos regulares de tiempo dando a conocer los resultados de la auditoría de la administración de negocios.

- ✓ Dar la posibilidad a los usuarios de valorar la confiabilidad, la seguridad y la viabilidad económica de la CA para lo cual es indispensable que la CA publique su CPS.

Los siguientes son ejemplos de CPS de autoridades de certificación operativas:

- ✓ http://www.bankgate.com/agree.htm
- ✓ http://www.verisign.com/repository/CPS1.1/CPSCH13.HTM

Seguridad y confiabilidad de su información

Un componente de la CA debe encargarse de manejar la información confidencial. Dicho proceso deberá tener en cuenta la existencia de métodos apropiados para chequear si las operaciones de la CA siguen los procedimientos establecidos.

Para prevenir que la información del suscriptor sea usada para propósitos impropios o para realizar actos ilegales, los procedimientos serán establecidos y mantenidos de manera tal que aseguren un manejo apropiado y definan el carácter confidencial de tal información. Cuando hablamos de información del suscriptor se incluye también a la información adquirida por la CA en el transcurso de su operación.

Un interrogante que puede surgir es qué ocurre cuando una CA termina su operación por alguna razón. En este caso, debe haber sido determinado un procedimiento de terminación para notificar la ocurrencia de dicha terminación a todos los suscriptores y demás personas afectadas directamente.

Registración de certificados

Existe un componente de la CA, el cual suele llamarse “Control” que verifica la autenticidad de los demandantes y la autenticidad de la información que maneja la aplicación antes de que la CA emita o revoque un certificado.

Dicho proceso de control para la emisión de un nuevo certificado generalmente incluye la verificación de autenticación de los datos, así como la notificación de los resultados de control y registración.

Existen algunos puntos a ser tenidos en cuenta cuando se lleva a cabo la verificación de la información de autenticación y de la información personal:

- ✓ La autenticación de la información aplicada debe ser verificada comparándola con la información provista en forma independiente por algún origen confiable o con la información ya verificada. Es deseable usar diversas fuentes de información para que la verificación de la misma pueda considerarse de alto nivel.
- ✓ Para verificar la identidad de un demandante, se usan diferentes métodos. Para asegurar una alta confiabilidad, lo más conveniente es realizar un control en persona.

La información personal puede autenticarse de diversos modos.

Cuando un usuario hace un requerimiento de un certificado a través de una aplicación on-line, el usuario inspecciona el formulario que la autoridad de certificación presenta en su pantalla, ingresa la información correspondiente en los campos de entrada y envía la información a la autoridad de certificación. En este caso la identidad del usuario es establecida corroborando la información que el mismo envía con la información provista por alguna organización confiable o con información que posee la CA. Finalmente, los resultados del control son enviados al usuario, lo cual puede realizarse, por ejemplo, a través de la dirección de mail registrada.

Si el requerimiento se hace por correo postal, se le exige al usuario que presente documentación de soporte como prueba de identidad. La identidad del mismo es establecida chequeando la información descriptiva y el sello del documento. Puede ocurrir también que el requerimiento se efectúe en persona. La identidad del usuario es controlada chequeando tanto la foto presentada como la información contenida en la solicitud.

La notificación de la aceptación de una demanda será reenviada por la autoridad de certificación al usuario junto con una solicitud para que el mismo confirme su razón para realizar la demanda.

El componente "Control" de la CA debe garantizar que tanto el nombre del sujeto como la clave pública del usuario sean únicos entre todos los certificados emitidos por la misma.

Es recomendable confirmar también que un usuario tiene su propia clave privada, correspondiente a la clave pública que será incluida en el certificado.

Una vez que la CA recibe y valida un requerimiento de un certificado realizado por un usuario o proveniente de una RA (Registration Authority), ésta genera el certificado correspondiente y lo firma con su propia clave privada.

Una vez generado el certificado en cuestión, la CA lo envía al suscriptor o a la RA. Opcionalmente la CA puede guardar el mismo en una copia de seguridad o enviarlo a un repositorio de certificados.

Manejo de la revocación de certificados

Ante la necesidad o el pedido de renovación de un certificado, es necesario controlar el certificado en cuestión, para lo cual se llevarán a cabo los siguientes procesos: chequeo del identificador, confirmación de unicidad, confirmación de las razones para la demanda, notificación de los resultados del control y registración.

En lo que a las funciones de la CA respecta, el control a realizarse en caso de revocación o suspensión de certificados, incluye procesos tales como: chequeo del identificador, notificación de los resultados del control y registración.

La siguiente tabla muestra quienes pueden requerir revocación de un certificado, cuáles son las razones que pueden tener para hacerlo y qué método utiliza la CA para confirmar la fuente del requerimiento:

Fuente de requerimiento	Razón típica para la revocación	Método típico para confirmar la fuente del requerimiento
Sujeto	Pérdida de la clave privada (por acceso indebido).	Firma del sujeto
	Pérdida de la clave privada (por olvido de la frase secreta, pérdida de archivo, etc).	El mismo procedimiento que para la emisión del certificado.
	Cambio de información de certificación importante.	El mismo procedimiento que para la emisión del certificado.
Autoridad de Registración	Cambio de organización.	Firma del representante de la autoridad de registración.
	Uso ilícito.	Firma del representante de la autoridad de registración.
Autoridad de Certificación	Error de la CA.	Confirmación de la CA.
	Requerimiento ficticio del usuario.	Confirmación de la CA.

Siempre es necesario validar a la persona que requiere la revocación, ya que la CA debe prevenir que una tercera parte maliciosa solicite la revocación del certificado de otra persona.

La verificación de la identidad del sujeto que realiza una nueva emisión de certificado es necesaria tanto en los casos de compromiso o pérdida de clave como en los casos de que la información importante del suscriptor haya cambiado.

Las razones por las cuales un usuario puede solicitar la revocación de un certificado pueden ser: la detección de un error en un certificado, el descubrimiento de un uso no autorizado del mismo, la emisión no autorizada de un certificado, etc.

Las autoridades de certificación pueden volver a emitir un certificado en circunstancias límites, dependiendo de la razón dada para la revocación.

Si la revocación requiere cambios en la clave pública o en información importante, el procedimiento es el mismo que para la emisión de un nuevo certificado; lo mismo ocurre si la revocación se basa en un requerimiento solicitado por otra parte que no sea el sujeto.

Modos de operación (Monitoreo, Manejo de la clave privada)

Es de vital importancia que la autoridad de certificación maneje sus pares de claves públicas/privadas con seguridad y alta confiabilidad durante su ciclo de vida completo.

Con respecto a la generación de claves debemos destacar los siguientes puntos:

- ✓ Un sistema criptográfico confiable de generación de claves debe ser usado para generar pares de claves y claves secretas. Será preferible instalar las funciones del sistema dentro de un módulo criptográfico.
- ✓ Los pares de claves y claves secretas serán generados bajo control dual, lo cual significa que dichos procesos serán supervisados por personas de diferentes organizaciones que tengan autoridad para ello.

Con respecto al almacenamiento de claves:

- ✓ Las claves generadas en un sistema criptográfico de generación de claves, serán divididas en múltiples componentes cada uno de los cuales formará parte de la clave. Los mismos se guardarán en forma separada para prevenir la divulgación no autorizada de algún componente simple mediante lo cual se comprometa la integridad de la clave completa. Cada uno de esos componentes será guardado en forma independiente por una persona autorizada
- ✓ Si las claves son guardadas dentro de un módulo criptográfico, las mismas serán guardadas bajo control dual, lo cual significa que el módulo criptográfico no puede ser accedido hasta que todas las personas autorizadas estén presentes.

Con respecto al uso de claves:

Las claves privadas y las claves secretas para firma digital y cifrado, serán usadas y colocadas en un módulo criptográfico.

Para garantizar la más alta seguridad en el uso de claves, es preferible operar los sistemas, incluyendo el módulo criptográfico, en modo “stand-alone”.

El realizar la copia de seguridad de las claves privadas y las claves secretas generadas es imprescindible para prevenir al suscriptor ante el detenimiento de operación de la CA debido al borrado accidental de dichas claves. El grado de seguridad para la copia de respaldo debe ser igual o mayor al requerido para el almacenamiento de claves.

La clave privada de una CA cuyo período de validez haya expirado, será destruida por razones de seguridad.

La CA debe asignar un período de validez a las claves de la CA con el fin de que las mismas sean renovadas en forma periódica.

Existen pautas a cumplir para las situaciones en que deba realizarse un procedimiento de recuperación debido a que la clave ha sido comprometida o que ha ocurrido un desastre:

- ✓ La CA debe preestablecer los procedimientos a seguir para los casos en que: se comprometa su clave privada debido a actividades criminales dentro de la misma, su clave privada sea descubierta por una tercera parte o la CA haya sufrido daños debido a un desastre.
- ✓ La CA debe revocar prontamente su certificado en caso de pérdida o compromiso de su clave privada. Además en esta situación, deberá revocar cada uno de los certificados de sus suscriptores que hacen uso de tal clave privada y notificarlos de dicha revocación.
- ✓ Otras medidas que debe tomar la CA en caso de pérdida o compromiso de su clave privada son: confirmación de la recuperación por medio de una operación de transferencia desde un backup seguro, renovación de la clave de la CA y de su

certificado y puesta en marcha de la nueva emisión de los certificados de los suscriptores.

Una buena táctica consiste en monitorear las formas en que los certificados son usados. Así como también es preferible emitir nuevamente un certificado sólo cuando el suscriptor realice el requerimiento para ello y no automáticamente.

Estructuras entre múltiples autoridades de certificación

Una estructura entre múltiples CAs provee uno o más caminos de certificación entre un suscriptor y una aplicación que utiliza certificados.

Un camino de certificación es una secuencia de uno o más nodos conectados entre el suscriptor y la autoridad de certificación raíz.

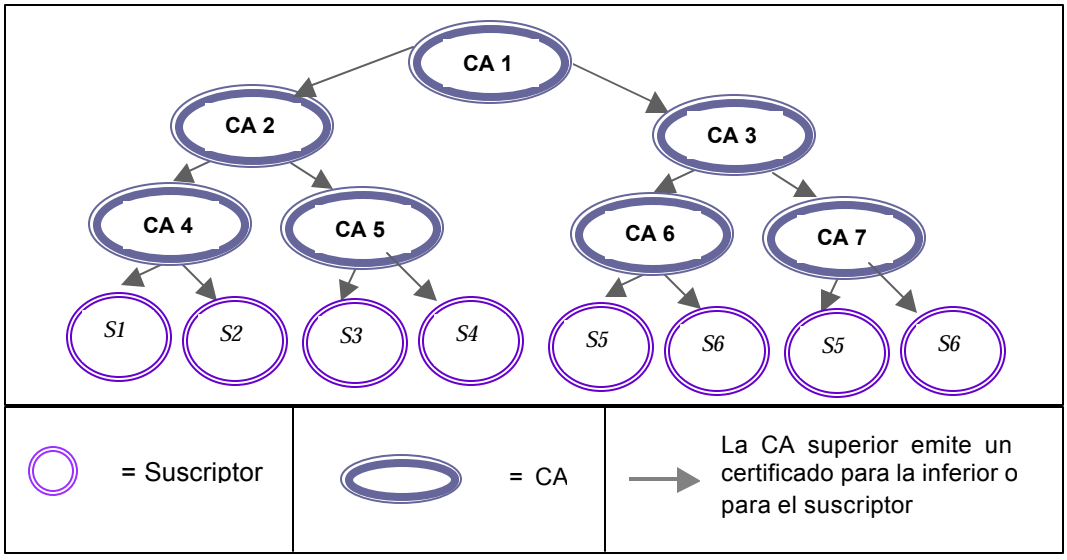
Una autoridad de certificación raíz es una autoridad en la cual la aplicación confía, importando en forma segura el certificado de la misma y guardándolo.

Un camino de certificación indica la forma en la cual una aplicación que usa certificados confía en el certificado de un suscriptor. Dicho camino incluye certificados de múltiples autoridades de certificación. Por ello es necesario el desarrollo de sistemas de clave pública que soporten la existencia de múltiples autoridades de certificación y las relaciones de confianza necesarias entre ellas.

Las estructuras encargadas de soportar tales relaciones de confianza varían ampliamente dependiendo de varios factores: la comunidad de usuarios, la naturaleza de las aplicaciones que requieren los certificados, el alcance geográfico, etc.

Tipos de estructuras existentes:

En el siguiente gráfico se muestra un ejemplo de una *estructura jerárquica top-down*:



Ejemplos basados en el gráfico:

1. Camino de certificación entre el *suscriptor 1* y el *suscriptor 2* como *CA 4* emitió los certificados para ambos, entonces el *suscriptor 1* considera a *CA 3* como su autoridad de

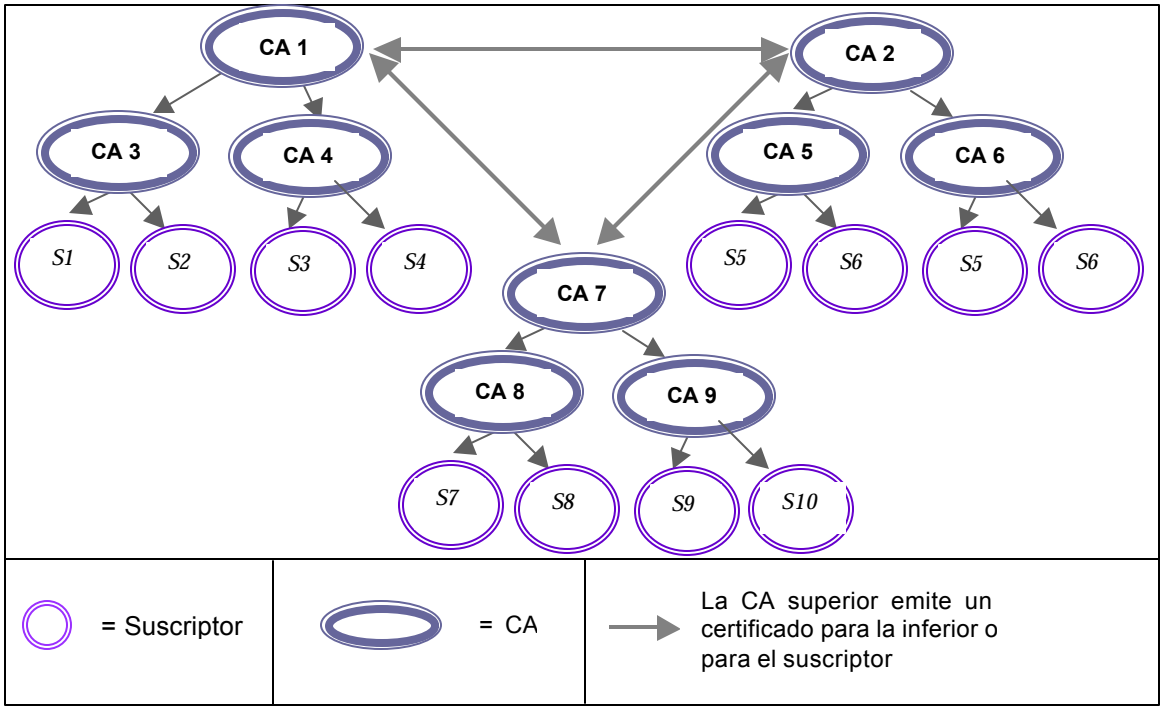
certificación raíz, por lo tanto el camino de certificación entre el *suscriptor 1* y el *2* incluye sólo el nodo 2.

2. Camino de certificación entre el *suscriptor 1* y el *suscriptor 6*: Si el *suscriptor 1* necesita confiar en el *suscriptor 6* debe considerar a la *CA 2* como su autoridad de certificación raíz. El camino de certificación entre el *suscriptor 1* y el *6* está compuesto por el *suscriptor 6* y la *CA 5*.

En este tipo de estructura es imprescindible que todos los suscriptores confíen en la CA Raíz para poder confiar en otro suscriptor, por ello resulta adecuada para aquellos modelos de organizaciones naturalmente jerárquicos en las cuales la confianza fluye desde la cima de la pirámide jerárquica hacia sus niveles más bajos.

El tipo de estructura anteriormente mencionada no resulta práctico para un desarrollo global de sistemas de clave pública en la comunidad de Internet. No sería realista pensar que la población de Internet completa confiara en una única CA. La confianza en este ambiente se desarrolla en forma de *"islas de confianza"*, en cada una de las cuales la comunidad confía en una CA en particular.

La manera que se utiliza para conectar dichas "islas" es que cada CA Raíz confíe en la CA raíz de las demás "islas". La interconexión entre las diferentes CAs se encuentra representada en el siguiente gráfico:



Validación y descubrimiento del camino de certificación

En todas aquellas estructuras de certificación que incluyan múltiples CAs es necesario contar con procedimientos de validación y descubrimiento del camino de certificación.

El problema del descubrimiento del camino de certificación puede tener varios grados de dificultad. El caso más sencillo es aquél en el cual tanto el suscriptor como el sistema que utiliza certificados confían en la misma Autoridad de Certificación. En ese caso el camino de certificación incluye un solo nodo que es el certificado del suscriptor.

El extremo contrario, en términos de dificultad de resolución del camino de certificación, es el caso en el cual el sistema que usa certificados debe resolver un problema teórico de grafos complejo, para determinar el camino en cuestión.

En cuanto a la validación del camino de certificación, la dificultad radica en resolver el problema de verificar la asociación entre el nombre del suscriptor y su clave pública, basándose en el camino de certificación. Las siguientes reglas describen el método usado para realizar la validación del camino de certificación:

1. Verificar que cada certificado en el camino de certificación esté firmado por el suscriptor cuya clave pública es la correspondiente a la del próximo certificado en el camino de certificación. Si el certificado es el último en el camino entonces debe ser verificado usando la clave pública de la CA Raíz.
2. Ninguno de los certificados en el camino debe haber expirado.
3. Ninguno de los certificados en el camino debe haber sido revocado.
4. Las extensiones críticas de cada certificado deben ser reconocidas y procesadas.

La regulación de organizaciones, control del personal y prácticas de negocios

Es importante para una CA mejorar su seguridad operacional y su confiabilidad en términos de tecnología, organización, control del personal y prácticas de negocios.

No debemos dejar de destacar la importancia que tiene el hecho de que las autoridades de certificación sean tan independientes como sea posible en términos de la influencia de empresas comerciales específicas, cuerpos y organizaciones, para conservar de esa forma su punto de vista de "TERCERA PARTE".

En los casos en los cuales muchas autoridades de certificación estén mutuamente relacionadas para incrementar la conveniencia de su uso, su condición de "tercera parte" crecerá y se ganará mayor confianza en ellas por parte de los usuarios con respecto a las otras Autoridades de Certificación.

En cuanto a aspectos humanos respecta, para contar con la habilidad necesaria, una CA debe disponer de un grupo de personas expertas en las técnicas de seguridad de datos, auditoría de sistemas y especialidades relacionadas con esta infraestructura.

Es particularmente importante para las autoridades de certificación tener acceso al personal que posea este conocimiento especial y la experiencia necesaria para abordar los problemas que puedan suscitarse en los servicios de certificación.

Con respecto a las capacidades de una CA a nivel organizacional, podemos nombrar ciertas pautas que deben cumplirse como ser:

- ✓ Los departamentos que tienen acceso a los datos críticos deben estar aislados de los otros.
- ✓ Para detectar problemas deben realizarse inspecciones internas dentro del departamento.
- ✓ Los departamentos deben utilizar auditores externos y otras formas de inspección externa para realizar el control.
- ✓ Si ocurre una desgracia, la causa de la misma debe ser identificada prontamente y corregida.

Las operaciones de la CA deben ser ejecutadas por personal de confianza y dicho personal debe ser controlado en forma continua para mantener la firmeza de la misma.

Las prácticas del negocio deben ser estrictamente reguladas de acuerdo con las políticas de seguridad relacionadas con la tecnología requerida y el equipamiento. Debe existir una política de seguridad que regule el acceso a las oficinas, al equipamiento y a la información.

Además con respecto a aspectos que tienen que ver con las finanzas, las autoridades de certificación que proveen servicios para un amplio rango de consumidores y corporaciones juegan un rol trascendental en las redes de información. Si una CA no puede continuar existiendo debido a

que ha sufrido una corrupción, un certificado emitido es efectivo hasta el término de su período de validez, pero el control de clave del emisor del certificado, base de la confianza, se torna críticamente débil.

Es también importante que una organización en donde está funcionando una CA tenga seguridad física, emplee a especialistas y técnicos expertos en certificación, criptografía, computación y leyes, y tenga el capital suficiente como para ser capaz de desarrollar y operar sistemas de certificación seguros y avanzados para garantizar la confiabilidad.

¿ Una CA es un Centro de Distribución de claves ?

Por los puntos mencionados anteriormente puede deducirse que una autoridad de certificación no es una simple extensión lógica de un Centro de distribución de claves. Existen importantes diferencias que no deben perderse de vista: El centro de distribución de claves se encarga simplemente de entregar las claves públicas a cada usuario que lo requiere. No hay un período de validez de la clave ni existe la seguridad de que la clave sea la clave pública auténtica de dicho usuario. Esto último se debe a que un "intruso" puede reemplazar la clave pública con su propia clave pública para poder de esa manera descifrar los datos cifrados con la misma.

La autoridad de certificación soluciona esta debilidad gracias a que en lugar de distribuir las claves públicas, distribuye certificados. Dichos certificados contienen un período de validez (antes y luego del cual la clave pública asociada no puede ser usada para verificar firmas). Incluso al estar los certificados firmados digitalmente con la clave privada de la CA, no pueden ser falsificados o reemplazados a través de un ataque.

Seguridad física de la Autoridad de Certificación

En cuanto a la seguridad física del Servidor en donde funciona la CA, este debe estar detrás de un firewall[1] y es conveniente que sea usado sólo para funciones propias de la CA.

[1]Firewall: es un conjunto de sistemas ubicado en el sitio central de conexión de una red. Su función principal es controlar el acceso desde y hacia la red protegida. Usualmente es colocado como protección para la conexión de la red al INTERNET.

Repositorios de datos

Hoy en día existe una noción formal de un repositorio como almacenamiento de certificados e información de estado de revocación, tal como *CRLs* [ver sección "Listas de Revocación" del Capítulo 4].

La designación oficial de una Base de Datos que actúe como repositorio tiene como propósito dejar en claro que son requisitos indispensables la confiabilidad[1] y la seguridad en el funcionamiento[2].

No resulta adecuado cargar a la CA con las responsabilidades de operar el repositorio. La misma puede no ser capaz de mantener la Base de Datos al nivel de servicio requerido. Por lo tanto esta funcionalidad (el proceso de publicación de la información referente a los certificados) puede ser llevada a cabo por un proveedor de dicho servicio.

Tanto X500[3], LDAP[4], como otros servicios de directorio propietarios, son alternativas viables para la distribución de certificados y listas de revocación.

Para ser aceptado, un proveedor de repositorio debe cumplir con las condiciones impuestas por la política de la CA en lo que respecta a sus tareas.

[1] En este caso, la confiabilidad se relaciona con la confianza legal, como ser que los registros del sistema no pueden ser adulterados sin autorización.

[2] Una característica que podemos mencionar como ejemplo de seguridad en el funcionamiento es "alta disponibilidad".

[3] X500: Dicho servicio provee una forma de diseminar gran variedad de información, incluyendo certificados de clave pública, a sistemas remotos. [ref. 1]

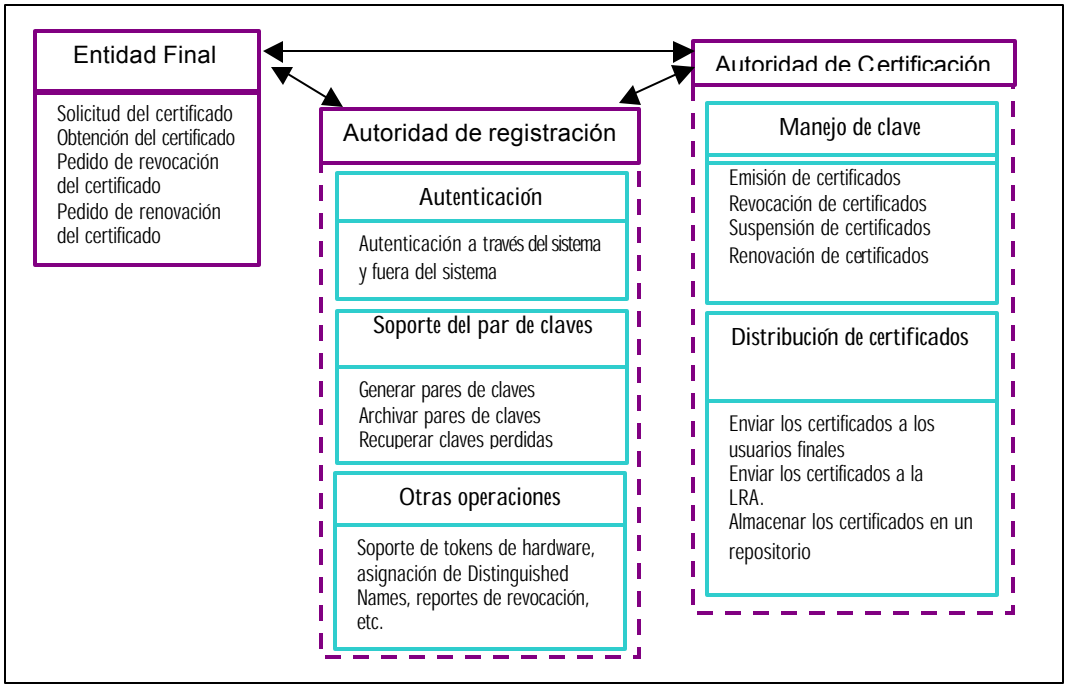
[4] LDAP (Lightweight Directory Access Protocol): Es un estándar para distribuir información en Internet. (Especificado en la RFC 1777). [ref. 2]

Autoridades de registración

En algunas situaciones es necesario repartir las tareas de manejo de certificados. Es allí cuando entran en juego las Autoridades de Registración.

Una Autoridad de Registración ejecuta varias tareas, como ser: asignación de un *"distinguished name"* (DN) a los suscriptores, confirmación de la información de identidad, reporte de revocación, aceptación de pedidos de renovación de certificados, distribución de tokens, etc. La misma puede además generar, archivar y recuperar pares de claves.

Las funciones relacionadas con el manejo de claves no pueden ser efectuadas por la RA. 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.



Modelo abstracto del modo de operación de una LRA

Las autoridades de registración son componentes opcionales dentro de la infraestructura PKI.

Generalmente se incluyen cuando una CA tiene la misión de autenticar suscriptores de una amplia región geográfica. Esto se debe a que puede ocurrir que los certificados manejados por dicha autoridad deban proveer un alto nivel de seguridad que requiera que las personas se hagan presentes ante la autoridad a la cual solicitan su certificado o la revocación del mismo. Para este fin es necesario contar con autoridades de registración locales que realicen tal verificación.

Listas de revocación

Cuando se emite un certificado, se espera que esté en uso durante su período de validez completo. Sin embargo, varias circunstancias pueden causar que el mismo se torne inválido antes de su vencimiento. Tales circunstancias incluyen cambio de nombre, cambio de asociación entre el sujeto y la CA (por ej., un empleado termina su empleo en una organización), y compromiso o sospecha de compromiso de la clave privada correspondiente.

X.509 define un método de revocación de certificados. Este método involucra el hecho de que cada CA emita periódicamente una *lista de revocación de certificados (CRL)*,

Una lista de revocación es una lista de certificados que han sido revocados antes de su fecha de expiración.

Funcionamiento o comportamiento

Una CRL es mantenida por una CA, y provee información de revocación relacionada con los certificados emitidos por dicha CA.

Cada certificado revocado se identifica en una CRL por un número de serie. Cuando un sistema usa un certificado, además de chequear la firma y validez del mismo debe acceder a la CRL más reciente que corresponda y constatar que el número de serie del certificado no está en esa CRL. Al examinar la CRL en cuestión, el sistema puede tener la seguridad de que el certificado del firmante no ha sido revocado.

Una entrada es agregada a la CRL como parte de una nueva actualización siguiendo a la notificación de revocación.

Cada CRL es colocada en un repositorio público para ser libremente accedida. También las CRLs pueden distribuirse a través de comunicaciones y sistemas Servidores no confiables.

Estas listas almacenan únicamente los certificados avalados actualmente por la CA. Los certificados son borrados de la lista en la que están incluidos al llegar la fecha de expiración de los mismos.

La verificación de revocación en línea puede reducir en forma significativa el tiempo de latencia entre un reporte de revocación y la distribución de la información correspondiente a las partes afectadas. Una vez que la CA acepta el reporte como auténtico y válido, cualquier consulta al servicio en línea que nombramos reflejará los impactos de la revocación del certificado.

Implementación

En términos más técnicos, una lista de revocación de certificados, más conocida como CRL, es una estructura de datos, digitalmente firmada por la CA emisora, que contiene la fecha y la hora de publicación de la misma, el nombre de la CA emisora y los números de serie de todos los certificados revocados cuyo período de expiración no se haya completado aún.

Cuando confiamos en un certificado, la aplicación que lo utiliza será la responsable de obtener la CRL emitida por la CA y chequear que el número de serie correspondiente no esté presente en la misma, asegurando de esta forma que se trata de un certificado válido.

La frecuencia con la cual se emiten actualizaciones de las CRLs es determinada por la CA. Para ello es necesario hacer un análisis costo/beneficio.

Las actualizaciones frecuentes de las CRLs incrementan la probabilidad de que una aplicación detecte y rechace entonces un certificado revocado, ya que la información siempre es muy reciente. Al mismo tiempo, las actualizaciones frecuentes traen como consecuencia el aumento del ancho de banda requerido para diseminar las CRLs.

En general, las transacciones con mayores requerimientos de confiabilidad requieren intervalos cortos entre actualizaciones, mientras que aquéllas en las cuales la confiabilidad no es un ingrediente fundamental, aceptan actualizaciones menos frecuentes.

A pesar de que no exista certificado alguno que haya sido revocado desde la última actualización, la CA debe emitir las CRLs de acuerdo a los intervalos por ella preestablecidos.

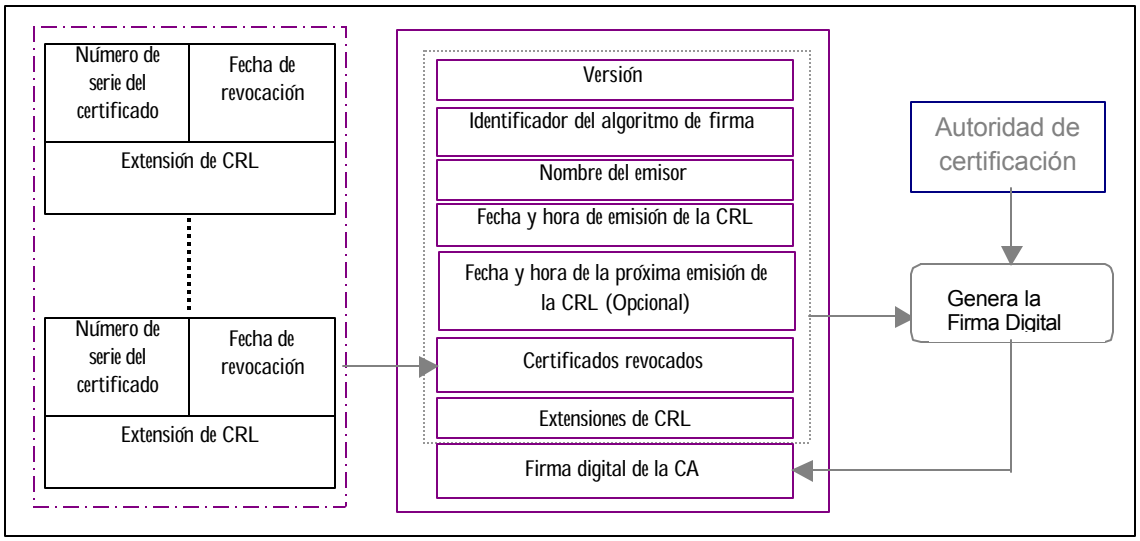
CRLs X.509

El formato para CRL X.509 es un estándar de ITU-T e ISO/IEC, publicado en 1988 como versión 1. Dicha versión fue posteriormente modificada para permitir campos de extensión, resultando ello en la versión 2 de formato para CRL X.509.

Campos básicos de una CRL X.509:

Los campos básicos del formato de CRL X.509 versión 2, los cuales ya estaban presentes en la versión 1, son los siguientes:

- ✓ *Versión* : especifica la versión del formato que se utiliza. Su valor es 2 si existe algún campo de extensión, en caso contrario este campo se omite.
- ✓ *Firma*: Este campo contiene el identificador del algoritmo usado para firmar la CRL.
- ✓ *Nombre del emisor* : especifica el nombre de la entidad que emite y firma la CRL.
- ✓ *Ultima actualización* : indica la fecha y hora de emisión de la CRL.
- ✓ *Próxima actualización* : indica la fecha y hora de la emisión de la siguiente CRL. Si todas los sistemas que utilizan certificados emitidos por la CA conocen la frecuencia con la que la misma emite las actualizaciones de las CRLs, dicho campo puede omitirse.
- ✓ *Certificado de usuario* : dicho campo contiene el número de serie del certificado revocado.
- ✓ *Fecha de revocación* : dicho campo contiene la fecha efectiva de revocación del certificado.



Formato de CRL X.509 v2

Extensiones de CRL

Cada comunidad PKI puede definir también sus propios campos de extensión y codificar información en los mismos de acuerdo a sus necesidades en cuanto a las CRLs.

Un ejemplo de este tipo de campo es el código de razón, el cual identifica el motivo por el cual se llevó a cabo la revocación, cuyo valor puede ser: clave no especificada, compromiso de clave y compromiso de CA, entre otros.

ANSI X9, ISO/IEC e ITU han definido un conjunto de extensiones estándares para CRLs X.509 v2 utilizando los mismos subcampos de los certificados X.509 v3. De esta forma las organizaciones pueden definir libremente sus propias extensiones.

Distribución de la información de CRLs

Existe una gran variedad de métodos para propagar dicha información a los sistemas que utilizan certificados. Los mismos se detallan a continuación:

Obtención de CRLs

Una aplicación que utiliza certificados puede acceder a una CA y obtener la CRL más reciente. Al igual que los certificados, las CRLs pueden estar almacenadas en repositorios de datos inseguros y pueden ser transmitidas a través de canales de comunicación inseguros debido a que las mismas están firmadas digitalmente por la CA que las emite.

Si utilizan este mecanismo, las aplicaciones necesitan conocer el momento de la próxima actualización de la CRL para determinar cuándo deberán acceder a la CA para obtener la última CRL. Como se explicó anteriormente, el período de actualización puede estar indicado en la estructura de datos que representa la CRL.

Esta forma de operación impone un período de funcionamiento inseguro, durante el cual la aplicación puede confiar en un certificado que ha sido recientemente revocado por la CA emisora, pero cuyo estado de revocación no se encontrará disponible hasta la próxima actualización.

Es recomendable que el período entre actualizaciones de CRLs sea breve, sin embargo existen aplicaciones que no pueden tolerar intervalos muy cortos entre actualizaciones.

Promoción de CRLs

Una CA puede promover CRLs a las aplicaciones que utilizan certificados cada vez que un certificado es revocado. Esta manera de diseminar la información logra eliminar el retardo inherente al método anteriormente mencionado.

El primer problema que se presenta a causa del uso de esta técnica es que se torna imprescindible que la infraestructura garantice que la información de revocación sea efectivamente entregada a las aplicaciones que utilizan certificados y no sea borrada por algún intruso mientras se encuentra en tránsito, hecho que no será detectado por la aplicación, la cual seguirá operando normalmente.

El segundo problema es la sobrecarga de la red debido a la propagación de información de revocación que ocurre frecuentemente. Una forma de disminuir los requerimientos de ancho de banda es publicar sólo las revocaciones críticas tales como las causadas por compromisos de clave privada.

Además en una comunidad PKI no organizada puede resultar dificultosa la tarea de determinar qué aplicaciones requieren que se les provea información de revocación así como aquélla consistente en determinar la infraestructura que deberá construirse para lograr la distribución de la información en cuestión.

Chequeo en línea de la información de estado (On-line Checking Status Protocol)

Como hemos mencionado anteriormente, una aplicación que usa certificados puede ejecutar una consulta en línea a una CA para determinar el estado de revocación de cierto certificado. El chequeo de estado en línea tiene una gran cantidad de ventajas. Este elimina el retardo existente en el método de obtención sin incurrir en el problema de sobrecarga propio del método de publicación dado por la distribución de una gran cantidad de información hacia una gran cantidad de destinos.

Con esta técnica tampoco resulta necesario un sistema de manejo central para determinar qué aplicaciones necesitan la información de revocación ya que son dichas aplicaciones las cuales realizan el requerimiento a través de las consultas en línea.

Sin embargo, el chequeo de estado en línea requiere que la CA cuente con un repositorio de datos confiable que esté disponible en todo momento. Además, dado que el servidor debe firmar digitalmente la respuesta a cada consulta, los recursos necesarios para el funcionamiento del servidor determinarán que dicha opción resulte costosa de implementar.

Como hemos mencionado en párrafos anteriores, hay un costo asociado al hecho de obtener información actualizada. Algunas aplicaciones pueden trabajar con información que tiene un día de antigüedad, otras requieren en cambio que la información sea casi instantánea. Parece entonces razonable asumir que, si las partes confiables requieren información actualizada carguen con los costos de contar con un servicio de actualización que implica un aumento del tiempo de procesamiento y del costo del uso de la red. Las partes que tengan necesidades de actualización menos exigentes pueden elegir no pagar esos costos. Sin embargo existe la posibilidad de que la autoridad exija que sólo su información más actualizada sea aceptada.

Otra opción es que una tercer parte, un proveedor de servicio produzca y distribuya la información de revocación para varias CAs diferentes. Si dichas CAs ofrecen su información de revocación de acuerdo a varios protocolos diferentes, entonces la tercer parte puede presentar una interfase simple a las partes confiables, con el fin de simplificar el procesamiento requerido por ellas.

Desventajas existentes y soluciones

El método estándar para transmitir la información de revocación, el cual hemos descripto en esta sección, presenta ciertos inconvenientes:

- ✓ Las listas son excesivamente largas (dado que, una CRL X.509 V1 emitida por una autoridad de certificación debe incluir todos los certificados válidos emitidos por esa autoridad que hayan sido revocados, el tamaño de la misma es directamente proporcional a la cantidad de suscriptores de la CA, al tiempo de vida de los certificados y a la probabilidad de revocación de cada uno). Para esta desventaja existen dos soluciones: la fragmentación de las CRLs, donde se divide el conjunto de certificados emitidos por una autoridad en subconjuntos y cada certificado X.509 V3 posee un puntero al fragmento de CRL que le corresponde; la segunda solución es el uso del OCSP [ver sección “OCSP” del Capítulo 4], el cual provee a los usuarios información de revocación de un certificado en particular.

- ✓ Como es costoso actualizar las listas en forma frecuente, es dificultoso proveer información actualizada. Nuevamente son dos las alternativas para solucionar este problema: se tienen “Delta CRLs” que usan una CRL base a la cual actualizan. Las Delta CRLs son emitidas con mayor frecuencia que las CRLs base. Otra alternativa es que OCSP sea usado para proveer información de revocación oportunamente.

- ✓ Cuando existe una única CRL, la misma es creada por una única entidad, por ello a veces se usan CRLs indirectas que son Puntos de Distribución de CRL firmados por otras entidades, para incrementar las garantías de confianza.

PKIX

El grupo de trabajo *PKIX* (Public Key Infraestructure X.509) [ref. 1] ha desarrollado documentos que cubren cinco áreas diferentes relacionadas con la puesta en marcha de las infraestructuras PKI.

La primer área incluye los puntos relacionados con las características de los estándares de criptografía de clave pública (PKC) X.509 v3 y de CRL X.509 v2.

La segunda área tiene en cuenta los protocolos operacionales en los cuales se confía con el fin de obtener cierta información como ser claves públicas o estado de las mismas. Como ejemplos de tales protocolos podemos mencionar LDAP(Lightweight Directory Access Protocol) [ref. 2] y OCSP [ver sección “OCSP” del Capítulo 4].

La tercer área cubre los protocolos de manejo utilizados por diferentes entidades para intercambiar información necesaria para el propio manejo de la PKI.

La cuarta área provee información acerca de las políticas y las prácticas de certificación, cubriendo las áreas de seguridad que no están directamente tratadas en el resto de PKIX.

La quinta área se relaciona con los servicios de certificación y de time stamping que pueden brindarse, los cuales pueden ser usados para construir servicios tales como el no repudio.

PKIX especifica, de esta forma, mecanismos innovadores, los cuales hacen posible que PKI constituya un verdadero soporte para el comercio electrónico.

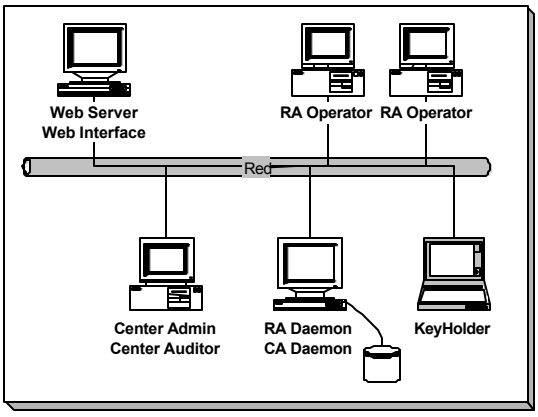
APuN-CA - Ejemplo de una Infraestructura PKI

Una de las herramientas existentes para definir una infraestructura de clave pública es APuN-CA. En esta sección nos dedicaremos a realizar una caracterización de la misma, mostrando sus componentes y describiendo su funcionalidad.

Desarrollado por un grupo de trabajo de la Administración Pública Nacional, este producto permite construir una Infraestructura PKI que cuente con las características necesarias para proveer el Servicio de Certificación a una determinada comunidad de usuarios.

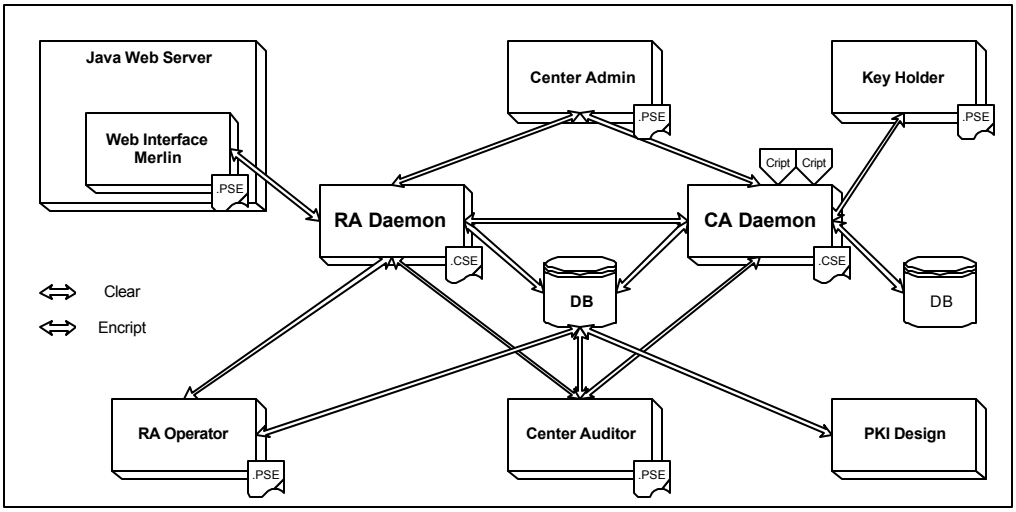
Este software, implementado en JAVA, posee una compleja arquitectura:
APuN-CA se compone de módulos independientes entre sí que interactúan entre ellos por medio de conexiones seguras para solicitar tareas. De esta manera es posible instalar cada uno de estos módulos en equipos independientes permitiendo una mayor escalabilidad y seguridad.

Las conexiones seguras entre módulos se logran estableciendo conexiones SSL con autenticación del cliente, las cuales ocurren cada vez que dos módulos de APuN-CA se comunican.
Cada una de las aplicaciones u operadores requiere de un certificado asociado al módulo al que se desea conectar, para poder establecer una conexión segura.



Módulos que constituyen APuN-CA

La arquitectura entre los módulos de ejecución es totalmente independiente, con una jerarquía propia de certificados que sólo es utilizada para intercomunicar las aplicaciones o módulos.



Descripción de las aplicaciones(módulos) de ApuN-CA

- ✓ *Proceso Autoridad de Certificación (Certification Authority Daemon)*
  Es el módulo central de generación de los certificados y CRLs. Controla los certificados activos.

- ✓ *Proceso Autoridad de Registración (Registration Authority Daemon)*
  Es el módulo encargado de aceptar los pedidos de certificados y de revocación, de permitir al verificador aprobar las solicitudes válidas y de ofrecer información sobre los certificados y las CRLs en distintos formatos.

- ✓ *Proceso Autoridad de Certificación y Registración (Certification & Registration Authority Daemon)*
  Son las funcionalidades de los dos módulos descriptos anteriormente en uno solo. Para instalaciones que no requieren la división.

- ✓ *Administrador Central (Center Administrator*)
  Esta aplicación permite habilitar y deshabilitar las tareas del proceso Autoridad de Certificación o del proceso Autoridad de registración. Un administrador puede suspender las tareas o dar de baja el servicio.

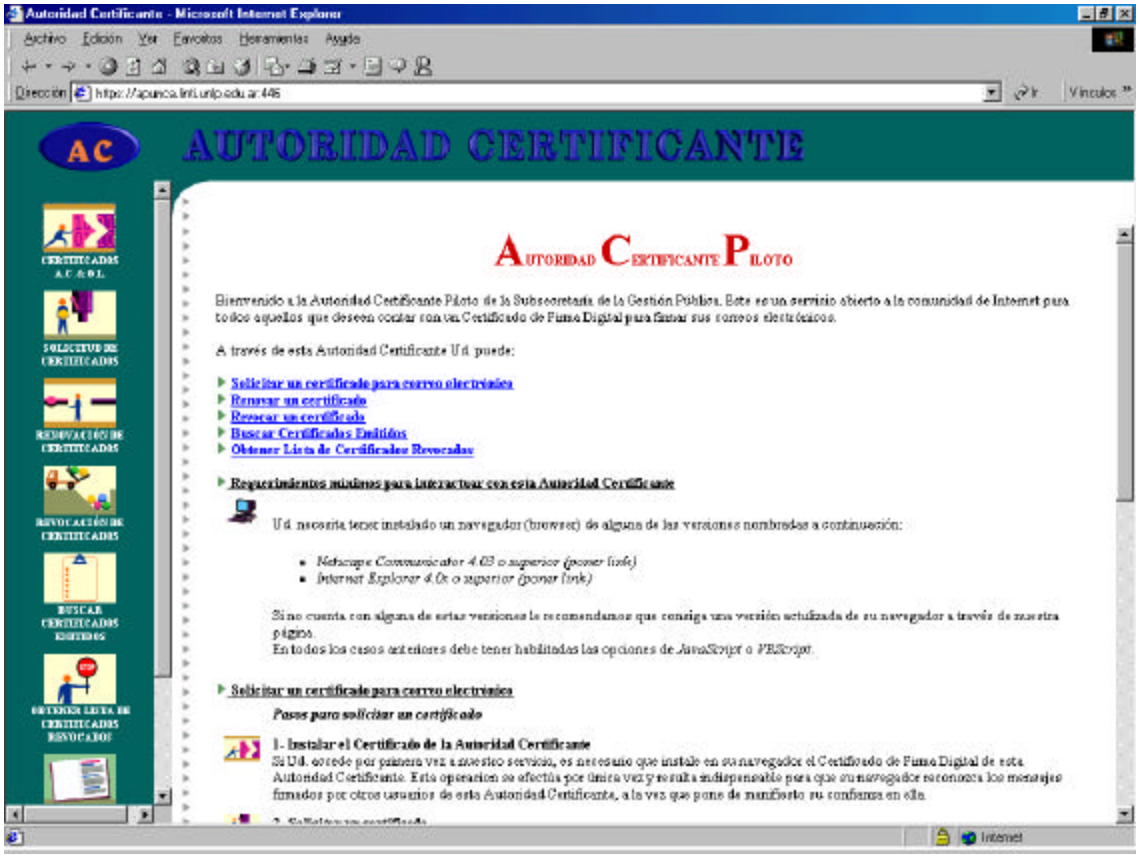
- ✓ *Auditor Central (Center Auditor)*
  Permite revisar las actividades de un Auditor Central (que se encuentra incorporado dentro de cada proceso). Tanto el Proceso CA como el Proceso RA mantienen una cola de operaciones que deben ser procesadas.
  Esta aplicación puede entonces observar las acciones que intervienen en el procesamiento de estas operaciones.

- ✓ *Encargado de la clave (Key Holder)*
  El responsable de la clave privada de la Autoridad Certificante. Es el encargado de instalar la clave privada para que puedan ser firmados los certificados y CRLs necesarios de modo que la CA pueda operar normalmente.

- ✓ *Operador de la Autoridad de Registración (Registration Authority Operator)*
  Permite la verificación de cada una de las solicitudes de certificado y su aprobación o rechazo, actividad que será realizada por el Operador o Verificador. La aplicación también permite ver los certificados emitidos y exportarlos en distintos formatos.

- ✓ *Diseñador de PKI (PKI Designer)*
  La estructura de la PKI es establecida desde esta aplicación. La misma permite definir los módulos (servidores y operadores), los cuales son necesarios para poder operar la PKI.
  Las Políticas de Certificación activas también se pueden diseñar desde esta aplicación.

Es importante destacar que APuN-CA cuenta con una interfaz WEB a través de la cual el usuario puede hacer uso de los servicios de certificación como ser requerir un certificado, buscar un certificado en particular o realizar pedidos de revocación.



Definición de la infraestructura PKI

La definición de la PKI consta de dos partes: la parte operacional de los servidores y operadores, y la definición de los certificados y claves privadas para firmar.

Los pasos a seguir son:

- ✓ *Definición general de la PKI*
  Se crea un nodo del tipo PKI que contiene a todos los demás módulos
- ✓ *Definición del servidor de CA /RA/CRA*

Es necesario definir un servidor que cumpla las funciones de CA. Un subnodo de este deberá ser un RA. Sin embargo es posible generar un CRA si estas dos funciones van a ser ejecutadas en el mismo servidor.

- ✓ *Definición de Acceso a la Base de Datos*
  Es necesario definir un módulo de base de datos para poder acceder a la misma. Esta definición será utilizada tanto por la CA y la RA a las cuales estén asociadas como por los operadores pertenecientes a la infraestructura.
- ✓ *Definición del Servidor de Correo Electrónico*
  Es necesario definir un módulo de correo electrónico para poder verificar la validez de la dirección de correo especificada por el usuario o para notificar al mismo la emisión de su certificado.
- ✓ *Definición de Operadores*
  Deben definirse tantos operadores como sean necesarios. A cada uno de ellos se le deben asignar las funciones que cumplirán. Es posible definir un acceso específico a la Base de Datos para cada uno de los operadores de tal manera de poder restringir el acceso a la misma dependiendo del rol de cada uno de ellos. Por ejemplo, el auditor sólo debe tener acceso de lectura a la base de datos.

La segunda etapa del diseño de la PKI corresponde a la creación del par de claves raíz y del requerimiento del certificado. Este último puede ser remitido al Organismo Licenciante o autofirmarse.

Una vez generado el par de claves es posible generar el requerimiento o autofirmar el mismo. Posteriormente es necesario exportar el par de claves y el certificado asociado para que pueda ser utilizado para firmar los certificados emitidos por la Autoridad Certificante. El archivo .pvk generado debe ser entregado al Responsable de la clave para que este pueda incorporarlo a la CA.

Por último es necesario exportar el certificado de la Autoridad Certificate en formato .DER y .PEM, para poder ser incorporados en la interfaz Web de APun-CA.

Características de la interfaz WEB de APun-CA

Al acceder a la página de la CA, el usuario puede visualizar o instalar en su navegador el certificado de dicha CA, así como obtener el mismo en formato PEM.
Si el usuario decide instalar dicho certificado en su navegador significa que ha decidido confiar en la CA.

También puede solicitar certificados.
En el caso de las solicitudes, la interfaz presenta los distintos tipos de certificados que pueden requerirse de acuerdo a las políticas anteriormente definidas.
Luego, ante la solicitud de un tipo de certificado en particular, ApunCA presenta un formulario en el que incluye el ingreso de datos necesarios para construir el certificado (el formulario lo arma dinámicamente y es acorde a lo definido en las políticas).

A través de esta interfaz se puede también:

-Realizar solicitudes de renovación de certificados.

- Realizar solicitudes de revocación de certificados

- Consultar la lista de certificados revocados (esta funcionalidad no se encuentra implementada en la versión 1.5.1con la cual hemos trabajado).

Brinda también la posibilidad de buscar un certificado en particular ingresando cualquier cadena de caracteres que aparezca en el nombre distintivo del certificado que se está buscando.

Permite consultar la política de certificación definida por la CA.
Además, con el fin de divulgar su CPS, APun-CA permite asociar una página html a cada política, en el momento en que la misma es definida.

Firma Digital

En algunas actividades comerciales la confianza tiene una importancia vital, el comercio electrónico no constituye una excepción. Cuando se realiza una oferta o se acepta un contrato es habitual que cada parte coloque algún identificador personal para mostrar que conocen el contenido del documento y lo aceptan. Tradicionalmente este identificador es a menudo una firma escrita. Como en el mundo digital no es posible realizarlo de esta forma es necesario contar con alguna alternativa. Esto no significa que simplemente la digitalización de una firma escrita sea un procedimiento adecuado, ya que es muy fácil que una tercera parte la copie y la reproduzca para adjuntarla a un documento electrónico.

Para entender las necesidades de la firma digital es conveniente tener claro previamente el concepto de firma en lo que a la ley respecta. Por ello en esta sección se presentarán también algunos conceptos relacionados con el tema.

Propósitos generales de la firma

- ✓ *Evidencia:* una firma autentica un escrito identificando al firmante con el documento firmado.
- ✓ *Ceremonia:* el acto de firmar el documento llama la atención del firmante con respecto al significado legal del acto de firmar.
- ✓ *Aprobación:* una firma expresa la aprobación o autorización del escrito, o la intención del firmante de que el mismo tenga un efecto legal.
- ✓ *Eficiencia y logística:* una firma en un documento escrito concede un sentido de claridad y finalidad de la transacción.

Con el fin de cumplir con estos propósitos, una firma debe contar con los siguientes atributos:

- ✓ *Autenticación del firmante:* indica quién firma el documento haciendo imposible que otra persona pueda producirlo sin autorización.
- ✓ *Autenticación del documento:* Puede identificar qué se está firmado logrando así que el documento no pueda falsificarse o alterarse sin detección.

Estos dos atributos son ingredientes fundamentales de lo que a menudo se llama *"Servicio de no repudiación"* en términos de la seguridad de la información.

Es válido afirmar entonces que, la simple digitalización de una firma escrita, no es suficiente para asegurar su validez legal, tal como ocurre en el acto de firma tradicional. Por ello surge la firma digital.

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).

Concepto

En términos técnicos, una firma digital es una secuencia de bits que resultan de la aplicación de una función de hash[1] sobre el mensaje transmitido, obteniendo así un resumen del mismo. El resumen o digesto resultante es entonces cifrado usando un algoritmo de clave pública y la clave privada del emisor.

Un receptor que posee la clave pública del emisor del mensaje puede entonces saber si la secuencia de bits fue creada usando la clave privada del firmante y además comprobar si la comunicación ha sido alterada o no desde que la secuencia de bits fue generada.

La firma digital es una cadena de caracteres alfanuméricos inteligible.

Según lo expresado anteriormente podemos afirmar que la firma digital constituye un arma más poderosa que la firma escrita, ya que esta última sólo garantiza la autenticidad[2]. El firmante expresa su acuerdo o la aceptación de lo que firma, y su firma se considera única para este individuo. Sin embargo, una vez que alguien firma un documento pueden realizarse agregados o modificaciones al mismo, sin el conocimiento del firmante. Por ello la firma escrita no provee verificación de la integridad del documento; mientras que dicho problema si es resuelto por la firma digital.

Mecanismos de uso de la firma digital

Las firmas digitales pueden ser usadas entonces para firmar documentos digitales, asegurar su integridad, certificar código de una aplicación y sus componentes, actualizar topologías de redes públicas seguras, cifrar datos, etc.

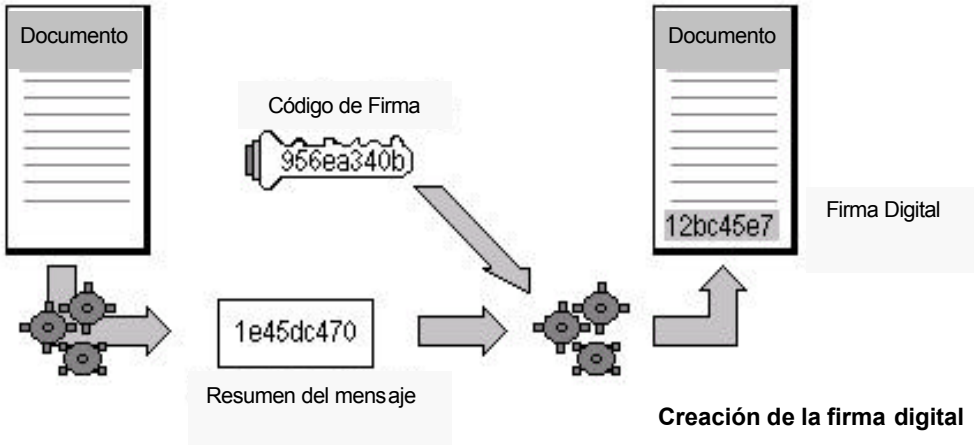
Utilizan algoritmos matemáticos avanzados para producir una representación de un documento que depende tanto del documento como de la persona que lo firma.

Las firmas digitales usan dos números de código únicos: un código de firma, o clave privada, y un código de verificación o clave pública. Dichos códigos son comúnmente llamados par de claves. Como ya hemos mencionado el código de firma es conocido sólo por quien va a efectuar la firma y no debe ser revelado a nadie; mientras que el código de verificación es distribuido libremente para cualquiera que quiera verificar la firma.

El uso de firmas digitales usualmente involucra dos procesos:

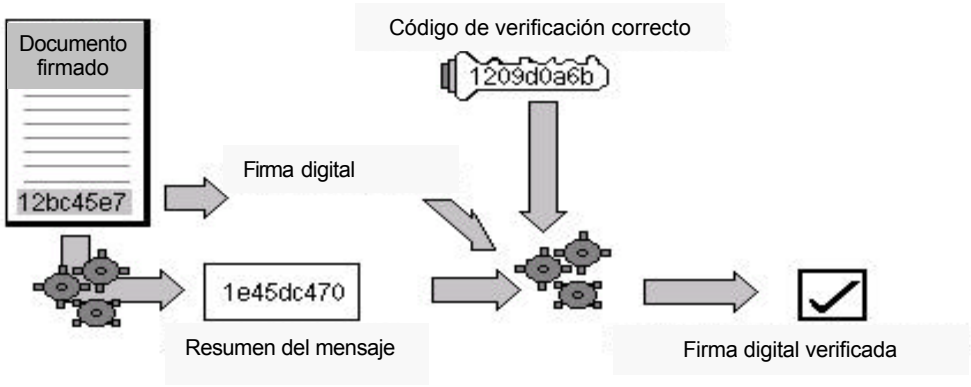
- ✓ *Creación de la firma digital:* Como primer paso, el firmante establece en forma precisa los límites de aquello que quiere firmar, lo cual constituye el mensaje. Entonces, una función de hash[1] en el software del firmante computa un resumen único para el mensaje. Luego, transforma dicho resumen en una firma, usando la clave privada del firmante (la cual puede residir en un archivo o en una Smart Card[3]), junto con el algoritmo de clave asimétrica.
  La firma digital resultante consiste en un conjunto único de números. Dicha firma es típicamente adjuntada al mensaje o documento correspondiente y es guardada o transmitida con él.

[1] Función de hash: Una función de hash o resúmen es un algoritmo que toma una entrada, generalmente de longitud indefinida, y generan un número pequeño de forma determinística (mensajes iguales siempre generan el mismo resultado).
[2] Autenticidad: La propiedad de ser genuino, capaz de ser verificado y, por lo tanto, confiable
[3] Las Smart cards son usualmente pequeñas tarjetas que se asemejan a las tarjetas de crédito y que pueden ser utilizadas para almacenar la clave privada de una persona, así como su certificado.



Creación de la firma digital

- ✓ *Verificación de firma digital*: Es el proceso de validar la firma digital usando para ello el mensaje original y la clave pública dada.
  La verificación de una firma digital es realizada computando un nuevo resumen del mensaje original (digesto) por medio de la misma función de hash usada para crear la firma digital.



Verificación de la firma digital

El verificador valida entonces:

1. Que la firma digital haya sido creada usando el correspondiente valor de clave privada y
2. Que el nuevo resumen computado se corresponda con el resumen original (es decir que sea exactamente igual), el cual fue transformado en la firma digital durante el proceso de firmado.

Por lo tanto se ratifica que:

1. La clave privada del firmante haya sido usada para firmar digitalmente el mensaje.
2. El mensaje no haya sido alterado.

En consecuencia, podemos afirmar que, si dicho mecanismo de chequeo falla, ello puede significar dos cosas: que el firmante no sea quien dice ser y/o que el documento haya sido alterado luego de haber sido firmado.

Varios criptosistemas asimétricos crean y verifican firmas digitales usando diferentes algoritmos y procedimientos, pero comparten su completo patrón operacional.

El proceso de creación de la firma digital y de verificación es realizado para obtener los efectos anteriormente mencionados para algunos propósitos legales: autenticación del firmante, autenticación del mensaje, acto afirmativo y eficiencia.

Los sistemas basados en clave pública deben estar seguros de que cada vez que ellos confían en una clave pública, la clave privada asociada pertenece al sujeto con el cual ellos se están comunicando. Sin embargo el par de claves no tiene asociación intrínseca con una persona. Por ello es necesario basar esta confianza en el uso de certificados los cuales ligan personas a sus claves públicas.

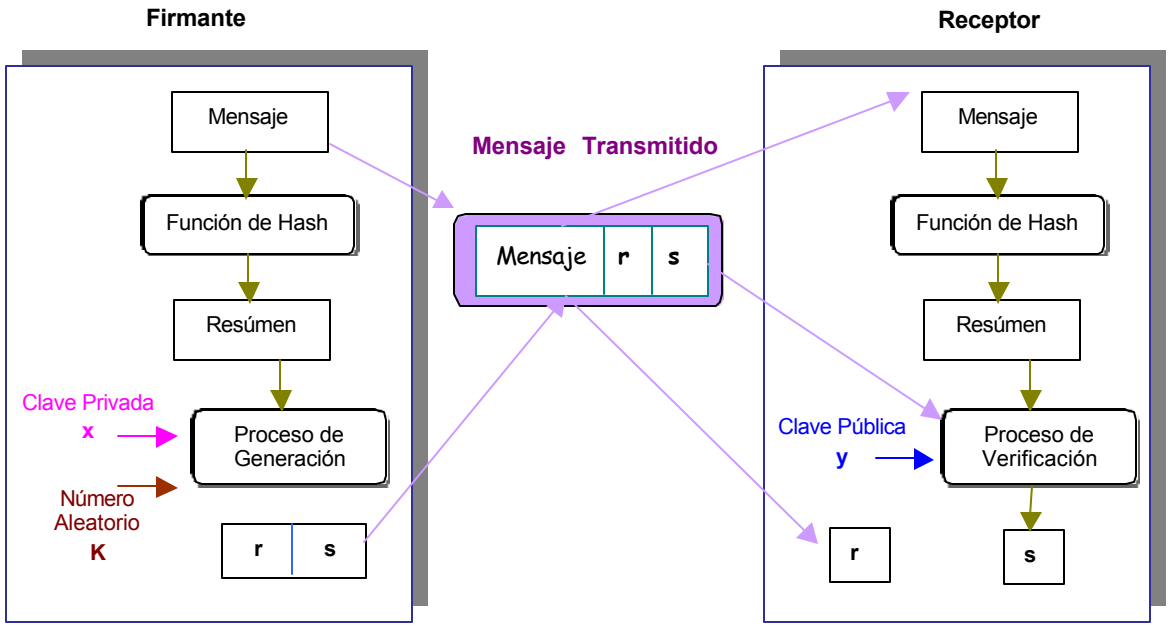
Algoritmo de Firma Digital (DSA)

Este algoritmo se encuentra en el Estándar de Procesamiento de Información Federal 186 de los Estado Unidos y en la primera parte del estándar ANSI X9.30.

DSA utiliza un sistema de clave pública irreversible, basado en ElGamal. Su seguridad depende de la dificultad de calcular algoritmos discretos: $\mathbf{Y = g^x \bmod p}$

Donde **p** es un número primo y **g** es un elemento elegido de orden grande modulo p. Es simple calcular **y**, dados **g, x** y **p**, pero es muy difícil de calcular **x**, dados **y, g** y **p**. Esto es la base del sistema de clave pública donde **x** *es la clave privada* **e y** *es la clave pública.*

El proceso de firmar y verificar un mensaje se encuentra ilustrado en el siguiente gráfico:



Algorítmo de Firma Digital (DSA)

El firmante genera un resumen del mensaje a ser firmado usando una unción de Hash. Este resumen luego es procesado por la operación de firmado de DSA el cuál requiere de la clave privada x y el cuál genera una firma comprendida por dos números r y s de 160 bits cada uno. Esta firma acompaña el mensaje original cuando es almacenado o transmitido.

Para firmar un mensaje que tiene un resumen **h**, el usuario elige un número aleatorio **k** (con 0<k<q) y calcula:

$$r = (g^k \bmod p) \bmod q$$
$$s = (k^{-1}(h+xr)) \bmod q$$

donde k^{-1} es un múltiplo de **k** mod **q**. El par de valores **(r,s)** *constituyen la firma del mensaje* en cuestión.

Digital Time stamping

Un servicio de *digital time stamping* (marca de tiempo digital) emite marcas de tiempo las cuales asocian una fecha y hora a un documento digital al cual previamente se le ha aplicado una función criptográfica (como por ejemplo una firma). Luego, el digital time-stamp, puede ser usado para probar que el documento electrónico existía en el momento especificado en su **"marca de tiempo".**

Una Autoridad de Time Stamp (TSA) es la tercera parte encargada de proveer la "prueba de existencia" de un dato particular en un instante de tiempo dado.

En el servicio de time stamping, una tercera parte(TSA) firma un mensaje con el fin de proveer evidencia de la existencia del mismo en un momento dado. Este servicio provee soporte para no repudio en el caso de que un usuario afirme que una transacción fue llevada a cabo luego del compromiso de su clave privada, dado que la "marca de tiempo firmada" garantiza que la transacción en cuestión ha sido creada en el momento indicado por dicha marca.

Sabemos además que X.509 define un método de revocación de claves públicas. En el mismo, cada CA emite periódicamente una estructura de datos firmada conocida como lista de revocación de certificados. La CRL es una lista con "marca de tiempo" que contiene los identificadores de las claves públicas que han sido revocadas. La misma es publicada en un repositorio de libre acceso y es accedida por los sistemas que utilizan certificados, los cuales además de chequear la validez de la firma deben asegurarse que el número de serie del certificado involucrado no está incluido en la CRL correspondiente, la cual está asociada a un determinado instante de tiempo.

Aspectos legales de la firma digital

Desde hace algún tiempo, el manejo de documentos electrónicos y el uso de información electrónica, constituye una gran parte de las actividades comerciales y administrativas. Sin embargo, en lo que se refiere a transacciones importantes, como ser el hecho de efectuar contratos, el uso de dicha información se encontraba, en un principio, restringido a grupos cerrados, cuyos miembros acordaban previamente el uso de la tecnología y las consecuencias legales derivadas del mismo.

Este hecho se debía a la falta de confianza, que aún existe, en el uso de tecnologías y las consecuencias legales de las transacciones electrónicas que se llevan a cabo en "ambientes abiertos", como Internet. Esta resistencia está siendo eliminada, por un lado, por la introducción de sistemas de información confiables, y por otro, por la adaptación del soporte legal vigente en lo que a las transacciones electrónicas respecta.

Hemos mencionado anteriormente aspectos relacionados con las tecnologías que permiten la construcción de los sistemas de información confiable, como ser la firma digital así como las razones por las cuales constituye un arma más poderosa que la firma manual. Vamos a introducirnos ahora en los aspectos relacionados con el soporte legal de la firma digital.

La deseada difusión y el uso efectivo de la infraestructura de clave pública depende en gran medida de la resolución de aspectos legales no certeros ya que en lo que a tecnología respecta, se han logrado mayores avances. A raíz de esta necesidad se han modificado en forma incremental

las leyes existentes. Además existen iniciativas privadas que proveen alguna infraestructura organizacional para el desarrollo de la tecnología de clave pública. Ciertos grupos profesionales (como American Bar Association) y organizaciones internacionales (como UNCITRAL) desarrollaron también sus propios proyectos relacionados al tema.

Cada país busca regular en la medida de lo posible el marco de las denominadas transacciones electrónicas. En países, como España, Colombia, EEUU, México y Francia, existen normativas vigentes en materia de comercio electrónico.

En otros como Argentina, Chile y Venezuela, en cambio, se está trabajando actualmente en proyectos que sustenten el uso de firma digital y regulen el comercio electrónico.

❖ Aspectos legales de la Firma Digital en Europa

El cuadro que presentamos a continuación intenta resumir la situación legal europea en lo que se refiere a aspectos relacionados con la firma digital y la seguridad de la información:

País	Legislación o reglamentación existente
Bélgica	-Proyecto de ley relativo al comercio electrónico, adoptado por la cámara de diputados en Julio del año 2000. Esta ley rige el uso de firmas electrónicas y regula además el uso de la criptografía y la prestación de servicios de certificación. Establece además las pautas para que se lleven a cabo las comunicaciones comerciales y los contratos electrónicos. [http://rechten.kub.nl/simone/luxlaw_fr.htm]
Francia	-Ley que regula la adaptación del derecho de la prueba a las nuevas tecnología y a la firma digital. Define la firma digital y establece las condiciones que determinarán la fiabilidad de la misma. [http://www.lafirmadigital.com/legislacion/europa/francia/art1.htm]
Alemania	-Proyecto de ley sobre firma digital cuyo propósito es crear las condiciones generales para las firmas digitales. -"Ordenanza de la Firma Digital": Documento que se encuentra en vigencia desde 1997 con el fin de regular el uso de la firma digital y de establecer importantes normas en cuanto a lo que a Autoridades de Certificación y a certificados respecta. [http://www.iid.de/iukdg/sigve.html]
Italia	-Ley General sobre la reforma del Servicio Público y principios del reconocimiento legal de los documentos electrónicos. Esta ley, sancionada en marzo de 1997, regula el tratamiento de los datos, actos y documentos con instrumentos informáticos o telemáticos provenientes de la administración pública y del sector privado, así como su archivo y transmisión mediante medios informáticos. El esquema de regulación descripto en esta ley también afecta a los contratos efectuados en forma electrónica. [http://www.interlex.com/testi/attielet.htm]
España	-El estado español ha tenido una activa participación en el logro de la utilización de la firma digital como método para proteger la seguridad y la integridad de las comunicaciones electrónicas. -El decreto-ley sobre firma electrónica tiene como objetivo establecer una regulación clara del uso de la firma digital, atribuyendo a la misma la eficacia jurídica y previendo el régimen aplicable a los prestadores de servicios de certificación. Dicho decreto respeta el contenido de la Directiva europea sobre firma digital (la cual se describe más adelante). -La administración española se ha hecho también un gran esfuerzo para adaptarse al papeleo virtual. La Fábrica Nacional de Moneda y Timbre se está

	erigiendo ya como principal certificadora de los documentos electrónicos emitidos por la administración. [http://www.lafirmadigital.com/legislacion/europa/espana/leyespa.htm]
Irlanda	-Existe una ley que sustenta el reconocimiento legal de la firma digital así como también busca regular los métodos existentes para proteger la información electrónica. [http://www.irlgov.ie/tec/communications/ecommercebill2000.PDF]
Suiza	-Se encuentra en vigencia Ordenanza sobre los servicios de certificación electrónica, dicha ordenanza define, con carácter experimental, las condiciones bajo las cuales deben operar los proveedores de servicios de certificación y regula las actividades relacionadas a los certificados electrónicos que realizan dichos proveedores. -Su objetivo consiste en promover el reconocimiento jurídico de las firmas digitales y permitir el reconocimiento internacional de los servicios de certificación y de sus prestaciones.
Austria	- Ley Federal de Firma Digital: Establece la infraestructura legal que gobierna la creación y el uso de las firmas digitales así como los procedimientos llevados a cabo con el fin de proveer los servicios de certificación. Esta ley se aplica tanto a sistemas cerrados en los cuales las partes que los forman se han puesto de acuerdo como a transacciones electrónicas sobre Internet. [http://www.a-sit.at/Englisch/Signature%20Law_E.pdf]

Las leyes en Europa

El 13 de mayo de 1999, la Comisión de Telecomunicaciones de la Unión adoptaba oficialmente el borrador de directiva que sentaría las bases del futuro desarrollo de las Autoridades de Certificación electrónicas (CAs). En dicho documento se sientan las normas de seguridad y responsabilidad que deberán seguir las CAs, además de asegurar su reconocimiento legal en la Unión Europea. Aunque algunos países, como Alemania o Italia, ya cuentan con legislación sobre el tema, la mayoría estaba también a la espera de que se establecieran normas comunes.

El problema es que, ahora, hay islas de certificación, y no existen relaciones cruzadas entre ellas, y ésto es necesario porque debe existir delegación de confianza de tal forma de armar una cadena de certificación jerárquica. Aunque ha habido intentos de confraternización entre CAs, por parte de fabricantes, universidades y la Internet Task Force, nada está claro.

Hasta el momento, el ejemplo norteamericano era de libertad de empresa total en el desarrollo de las CAs. Europa, en cambio, se muestra más sensible a la opinión de los expertos en seguridad quienes creen que el gobierno debe velar porque haya criterios mínimos a los que todo el mundo pueda llegar y permitir que todos se entiendan.

El campo de acción borrador de la Unión Europea (UE) se reduce al público en general, excluyendo a las corporaciones o los sistemas bancarios, y su intención principal es definir las normas de juego más esenciales, como los efectos legales o la cooperación de CAs europeas con terceros países.

En la Propuesta Directiva de la UE [ref. 1] se hace mención a varias reglas relacionadas con el rol de los Estados Miembros frente a las firmas electrónicas y los certificados, los requisitos que deben cumplir los prestadores de servicios de certificación y el contenido de los certificados.

Existe también una Propuesta del Parlamento Europeo sobre una Estructuración Común de las Firmas Electrónicas [ref. 2] y un proyecto denominado ICE-TEL el cual define una infraestructura de certificación en Europa [ref. 3].

❖ Aspectos legales de la Firma Digital en Asia

En ASIA, encontramos también algunos países que han instaurado leyes de firma digital o que cuentan con proyectos relacionados al tema, como se transcribe a continuación:

País	Legislación o reglamentación existente
Malasia	- Ley de firma digital que data de 1997. - Ley de regulación de firma digital escrita en 1998. Ambas contienen regulaciones relacionadas con el control y el licenciamiento de las autoridades de certificación, los requerimientos que deben cumplir las autoridades de certificación, los deberes a los que están comprometidos tanto los suscriptores como las autoridades de certificación, los efectos de la firma digital y los servicios de repositorios y de timestamp. [Ref http://www.cca.gov.my/legislation.html]
Honkong	Existe una ordenanza, con posteriores correcciones, que contiene aspectos ligados a las firmas digitales, los contratos electrónicos, el reconocimiento de las autoridades de certificación y los certificados, entre otros. [http://www.info.gov.hk/itbb/english/it/eto.htm]
Singapur	- Ley de firma digital: Establece las condiciones para el reconocimiento legal de la firma digital y de los registros electrónicos. http://www.cca.gov.sg/eta/framecontent.html Define también las características necesarias para que un proveedor de servicios de certificación sea considerado confiable. Así como incluye reglamentación relacionada con: efectos de la firma digital, requisitos de la firma digital, obligaciones de las autoridades de certificación, obligaciones de los suscriptores, etc. - Ley Certificadoras: Se encarga de regular el licenciamiento de las autoridades de certificación, estableciendo para ello determinados criterios. Establece además las pautas que rigen la revocación de la licencia de una AC, los requerimientos para el repositorio y otros aspectos de administración de las autoridades de certificación. [http://www.cca.gov.sg/eta/framecontent.html]

❖ Aspectos legales de la Firma Digital en América

Mirando hacia nuestro continente, podemos mencionar a Colombia, Chile, Uruguay, Brasil, Perú, EEUU, México y Venezuela, como ejemplos de países en los cuales existe un gran avance en cuanto a proyectos y leyes existentes dentro de esta área.

La situación en estos países, presentada en forma sintética sería la siguiente:

País	Legislación o reglamentación existente
Colombia	- Proyecto de ley sobre Comercio electrónico (con fecha de abril de 1998), el cual define y reglamenta el acceso y uso del comercio electrónico, el uso de las firmas digitales y la implementación de las entidades de certificación. Esta ley es aplicable a todo tipo de información en forma de mensaje de datos. [http://www.qmw.ac.uk/~tl6345/colombia_sp.htm]
Uruguay	- Realización de reuniones con los involucrados en el tema de Firma Digital. - Desarrollo de programas para el firmado y verificación de Firmas con certificados X.509 emitidos por alguna autoridad de Certificación.

	- Evaluación de diferentes productos de encriptado y firma. - Definición de un formato de intercambio de mensajes firmados. [http://www.aduanas.gub.uy/Firma_digital.htm]
Brasil	- Decreto que rige desde septiembre del 2000 que establece normas acerca de la constitución de la Infraestructura de claves públicas del Poder Ejecutivo Federal. Dicha infraestructura dará lugar al uso de la tecnología de claves públicas, la cual facilitará, en el ámbito de los órganos y las entidades de la Administración Pública Federal, la garantía de los servicios de autenticidad e integridad de datos, la irrevocabilidad y la inmutabilidad de las transacciones electrónicas así como de las aplicaciones que utilicen certificados digitales. - Anteproyecto de ley de firma digital. En dicho documento se mencionan varios puntos relacionados con el comercio electrónico, la validez jurídica de un documento electrónico y la firma digital. Entre los puntos que se tratan se hallan: un detalle de los contenidos requeridos en un contrato electrónico, pautas acerca de la privacidad de la información que se transmite en una transacción electrónica, obligaciones y derechos de los intermediarios, etc. [http://www.lafirmadigital.com]
Chile	- Existe un decreto que establece las normativas que rigen para el manejo de documentos electrónicos y para la regulación de la firma digital. - Actualmente, tres proyectos de ley relativos al comercio electrónico están en discusión en el Parlamento Chileno.
México	- Rige un decreto en el cual se reforman y agregan diversas disposiciones del Código Civil para el Distrito Federal en Materia Común y para toda la República en Materia Federal, del Código Federal de Procedimientos Civiles, del Código de Comercio y de la Ley Federal de Protección al Consumidor.
Venezuela	- Ley sobre Firma Digital en la cual se definen ciertas pautas de los mensajes de datos, su emisión y su recepción, las firmas electrónicas y el servicio de certificación
Perú	- Ley de firmas y certificados digitales del Perú: Publicada en mayo del 2000, esta ley tiene por objeto regular la utilización de la firma electrónica otorgándole la misma validez y eficacia jurídica que el uso de una firma manuscrita u otra análoga que posea las mismas atribuciones que la manuscrita.
EEUU	- La ley 106-229 regula el uso de firmas electrónicas en los actos de comercio nacionales e internacionales.

❖ La firma digital en la Argentina

En lo que a nuestro país respecta, el uso de la firma digital se encuentra actualmente regulado por el decreto número 427, el cual reglamenta la implementación de la tecnología de firma digital con el objetivo de permitir a la Administración Pública Nacional hacer más eficientes sus circuitos administrativos, al hacer posible la identificación fehaciente del autor y/o emisor de la información, y al otorgar garantías de que el documento digital no ha sido alterado desde el momento de su firma.

La Secretaría de la Función Pública integra desde hace dos años el Subcomité de Criptografía y Firma Digital (Integrado además por: ANMAT, ANSeS, Banco Central de la República Argentina, Banco de la Nación Argentina, Comisión Nacional de Valores, Ministerio de Economía, Ministerio de Justicia, Ministerio de Relaciones y la Secretaría de Comunicaciones) que ha dedicado importantes esfuerzos a viabilizar la introducción de esta tecnología en nuestro país.

Como resultado, el 16 de abril de 1998, el Sr. Presidente de la Nación, Dr.Carlos Saúl Menem, firmó el Decreto Nº 427/98, que permite el uso de esta tecnología para los actos internos del Sector Público Nacional que no produzcan efectos jurídicos individuales en forma directa, y sienta las

bases para la creación de una Infraestructura de Autoridad Certificante de la Administración Pública Nacional.

En este decreto se tienen en cuenta ciertas consideraciones necesarias en nuestro país, entre otras, como ser: la necesidad de optimizar la actividad de la Administración Pública Nacional adecuando sus sistemas de registración de datos, tendiendo a eliminar el uso del papel y automatizando sus circuitos administrativos, lo cual amerita la introducción de tecnología de última generación, entre las cuales se destacan aquéllas relativas al uso de la firma digital, susceptible de la misma o superior garantía de confianza que la firma ológrafa; la necesidad de estimular la difusión de las citadas tecnologías a través del dictado de una norma de jerarquía superior, que promueva la extensión del uso de la firma digital a todo el ámbito del Sector Público Nacional; que el mecanismo de la firma digital cumple con la condición de no repudio, siempre y cuando su implementación se ajuste a los procedimientos descriptos en este decreto y que resulta indispensable establecer una Infraestructura de Firma Digital para el Sector Público Nacional con el fin de crear las condiciones de un uso confiable del documento suscripto digitalmente.

El contenido de dicho decreto consiste en diez artículos que establecen, entre otros puntos, los siguientes, los cuales, según nuestro criterio, vale la pena destacar:

- ✓ La autorización por el plazo de dos años para implementar los manuales de procedimiento y definir los estándares referentes al empleo de la firma digital en la instrumentación de los actos internos del Sector Público Nacional, que no produzcan efectos jurídicos. En el régimen del presente Decreto la firma digital tendrá los mismos efectos de la firma ológrafa, siempre que se hayan cumplido los recaudos establecidos en el decreto dentro del ámbito del sector Público Nacional, dentro del cual se comprende la administración centralizada y la descentralizada, los entes autárquicos, las empresas del Estado, Sociedades del Estado, Sociedades Anónimas con participación estatal mayoritaria, los bancos y entidades financieras oficiales y todo otro ente, cualquiera que sea su denominación o naturaleza jurídica, en el que el Estado Nacional o sus organismos descentralizados tengan participación suficiente para la formación de sus decisiones.
- ✓ Se resuelve que los organismos del Sector Público Nacional deberán arbitrar los medios que resulten adecuados para extender el empleo de la tecnología de la firma digital. Se determina que un certificado de clave pública emitido por una Autoridad Certificante Licenciada representará la correspondencia entre una clave pública (elemento del par de claves que permite verificar una firma digital) y el agente titular de la misma.
- ✓ Se asigna a la Contaduría General de la Nación cumplir las funciones de Organismo Auditante en los términos de lo establecido en el Decreto.

Este decreto, alternativa al decreto 333/85, dictamina además que:

- ✓ Cada persona posea su CERTIFICADO DE CLAVE PUBLICA.
- ✓ Cada persona posea su CLAVE PRIVADA que nadie deberá conocer.
- ✓ Los CERTIFICADOS DE CLAVE PUBLICA son emitidos y administrados por las AUTORIDADES CERTIFICANTES LICENCIADAS.
- ✓ El ORGANISMO LICENCIANTE otorga las licencias y supervisa técnicamente a las Autoridades Certificantes.
- ✓ El ORGANISMO AUDITANTE audita a ambos.
- ✓ Cualquier organismo de la APN, centralizado o descentralizado, bancos, entidades financieras oficiales, entes autárquicos, empresas del Estado, pueden ser AUTORIDADES CERTIFICANTES.

Este decreto cuenta además con dos anexos.

Existen también dos resoluciones vigentes ligadas al tema que estamos tratando:

La Resolución SFP 194/98 referente a los estándares aplicables a la Infraestructura de Firma Digital, describe especificaciones técnicas, obligaciones y recomendaciones para el Organismo

Licenciante y las Autoridades Certificantes, tendientes a resguardar la CONFIABILIDAD del sistema.

La Resolución SFP 212/98 referente a la Política de Certificación para el licenciamiento de las Autoridades Certificantes, la cual establece los criterios para el licenciamiento de las Autoridades Certificantes y los requisitos y condiciones para la emisión de los certificados de clave pública.

Plan piloto de Firma Digital en el sector público

El objetivo de este plan piloto fue firmar documentos electrónicos utilizando certificados digitales, de acuerdo a los estándares de la industria; enviarlos y distribuirlos desde un remitente hacia uno o varios destinatarios mediante correo electrónico.

Se describen:

- Estrategia
- Metodología
- Definiciones y requerimientos previos

Etc.

Situación actual

En la República Argentina, en el seno del Congreso de la Nación existen dos proyectos sobre "Firma Digital" que esperan tratamiento son ellos:

- ✓ El proyecto de los senadores Pedro Del Piero y Luis Molinari Romero, proyecto que cuenta con el apoyo del gobierno. El mismo intenta establecer el reconocimiento del empleo de la firma electrónica y de la firma digital, marcando las diferencias entre las mismas. Contiene reglamentación relacionada con los requisitos de los certificados digitales, su período de validez, las autoridades de aplicación, etc.
- ✓ El otro proyecto fue presentado en la Cámara Baja por el diputado Pablo Fondevila con el apoyo de Irma Parentella y Norberto Nicotra . El objetivo de dicha ley, al entrar en vigencia, sería la habilitación y la regulación del empleo de la firma digital así como la implementación del servicio público de certificación.

Este proyecto de ley define, además de otros aspectos referentes al tema, la organización institucional subyacente, aspectos relacionados con el contenido y la validez del certificado de clave pública, así como funciones y obligaciones de las CAs.

El 16 de Agosto del corriente año, la Honorable Cámara de Diputados de la Nación dio sanción al Proyecto de Ley de Firma Digital, de autoría del Diputado Nacional Pablo Fontdevila. Se prevee que en los próximos días el Proyecto pase al Honorable Senado de la Nación para su tratamiento. Para más información acerca del proyecto referirse al sitio:

http://www.elprincipe.com/parlamento/refor_estado/index.shtml

Implementación de seguridad en Servicios de Internet

Seguridad en la transmisión de mensajes de correo electrónico

Las aplicaciones de mensajes, incluyendo el correo electrónico, presentan un claro ejemplo de la clase de aplicaciones cuyas necesidades de seguridad no se satisfacen sólo con las medidas de seguridad de la red (por ejemplo: firewalls, VPNs, etc).

Los mensajes seguros requieren protección desde el escritor hasta el lector en un ambiente en el cual es posible que los mismos atraviesen múltiples conexiones de red o sean guardados y redirigidos por un gateway de nivel de aplicación desconocido.

Existen dos niveles de servicios de seguridad para la transmisión de mensajes:

- ✓ *Servicios de protección básicos.* En este grupo se encuentran los siguientes: autenticación del origen, integridad del contenido, confidencialidad del contenido y no repudio del origen.
- ✓ *Servicios de protección avanzados.* Estos consisten en servicios de confirmación que proveen notificaciones al origen de que el mensaje fue recibido por el destinatario o por algún punto en el camino del mensaje.
  Los servicios de confirmación son: *prueba de entrega* (asegura al origen que el mensaje fue entregado al receptor correspondiente sin sufrir modificaciones durante la transferencia); *prueba de envío* (asegura al origen que el mensaje fue aceptado por su Servidor de correo para ser transferido al receptor correspondiente, *no repudio de entrega* (prueba al origen, con una evidencia fuerte, que el mensaje fue recibido por el receptor sin ser alterado), *no repudio de envío* (prueba al origen con una evidencia fuerte, que el mensaje fue aceptado por su Servidor de correo para ser luego transmitido).

Existen varios protocolos que tienen como objetivo la transmisión de mensajes seguros. Algunos como PEM [ver sección "PEM" de este Capítulo] y X.400 [ver sección "X.400" de este Capítulo] cuentan con un sólido fundamento teórico pero no han sido utilizados en implementaciones comerciales debido a su falta de interoperabilidad con el sistema de correo utilizado en Internet. Otro protocolo muy difundido, PGP, es muy efectivo en comunidades de usuarios que se comunican en forma casual a través del correo electrónico pero no resulta adecuado cuando existen requerimientos de escalabilidad. Otra desventaja de los protocolos mencionados es que los mismos no son interoperables.

Existe un protocolo que es compatible con el sistema de transmisión de mensajes en Internet y que es reconocido por proveedores comerciales importantes, como Netscape y Microsoft, el cual ha sido ampliamente desarrollado. Este protocolo es conocido como S/MIME (Secure Multipurpose Internet Mail Exchange) y permite a los usuarios intercambiar mensajes sin preocuparse por los aspectos relacionados con la interoperabilidad.

S/MIME

S/MIME [ref. 1] es un protocolo que agrega firma digital y encriptado a los mensajes MIME [ref. 2] de Internet. Dicha especificación para mensajes seguros fue creada en 1995 por un grupo de proveedores comerciales liderados por RSA Data Security.

Si bien MIME (formato oficial estándar para los mensajes extendidos de correo electrónico), define cómo se estructura el cuerpo de un mensaje, el mismo no provee servicios de seguridad. Por ello, el propósito de S/MIME es definir tales servicios, siguiendo la sintaxis dada en PKCS #7 para firmas digitales y cifrado.

La sección del cuerpo del mensaje MIME transportará entonces un mensaje PKCS #7, el cual representará el resultado de un proceso criptográfico de otras secciones del cuerpo MIME.

Además de proveer los dos mecanismos de seguridad de firma digital y encriptado para dotar al correo electrónico inseguro de los servicios de autenticación, integridad y confidencialidad, la aplicación de S/MIME permite a otros productos de software, como el sistema de intercambio de datos electrónicos (EDI)y los servicios de comercio electrónico en línea, incorporar *seguridad*.

Mecanismos y servicios de seguridad de S/MIME

S/MIME provee cuatro servicios de seguridad que el cliente de mail tradicional[1] puede aplicar a los mensajes que este envía o recibe para proteger a los emisores y a los receptores del mismo frente a una variedad de ataques posibles en una red abierta.

Los servicios de seguridad provistos por S/MIME son los siguientes:

Servicio de seguridad	Mecanismo de seguridad
Autenticación del origen	*Firma digital*
Integridad del mensaje	*Firma digital*
No repudio del origen	*Firma digital*
Confidencialidad del mensaje	*Encriptación*

- ✓ *Autenticación del origen*: Este servicio asegura al receptor que el mensaje ha sido enviado por el que dice ser el emisor (cuya dirección de correo electrónico figura como valor del campo FROM del mensaje).
- ✓ *Integridad del mensaje*: Este servicio garantiza que el contenido del mensaje no sea alterado durante su tránsito desde el origen hacia el destino.
- ✓ *No repudio del origen*: Este servicio protege a un receptor frente a la posibilidad de que el emisor niegue haber enviado el mensaje.
- ✓ *Confiabilidad del mensaje*: Este servicio protege al contenido del mensaje contra un acceso no autorizado realizado por un intruso en la ruta desde el origen hasta el destino.

Quien decide qué servicios de seguridad serán aplicados a determinado mensaje es el usuario o el sistema (este último caso ocurre cuando la aplicación envía mensajes).

Como la aplicación de los servicios de seguridad no está limitada al correo electrónico, cada protocolo que transmita objetos MIME puede hacer uso de dichos servicios.

Encriptación usada en S/MIME

Las aplicaciones compatibles con S/MIME (agentes S/MIME) pueden soportar una gran variedad de algoritmos de encriptado de clave simétrica debido a que la sintaxis subyacente al estándar PKC utilizado para encriptación, es bastante general.

Un agente S/MIME puede soportar tanto RC2 de 40 bits (conocido como algoritmo de encriptación débil) como triple DES (conocido como algoritmo de encriptación fuerte). Los algoritmos de encriptación provistos por un agente en particular dependen de factores tales como soporte legal existente, restricciones de exportación vigentes o acuerdos entre partes privadas.

Además, un agente receptor puede contar con capacidades de encriptación diferentes al agente emisor.

[1] Este es conocido como Mail user agent (MUA)

La decisión de utilizar un algoritmo de encriptación fuerte o débil depende de varios factores, tales como capacidades de encriptación del agente emisor, capacidades de desencriptado del agente receptor, preferencias del usuario, etc.

Cabe la posibilidad de que como resultado de la elección del emisor, el receptor no sea capaz de descifrar el mensaje. Para evitar tal inconveniente, S/MIME soporta un atributo llamado Capacidades, el cual permite a un agente receptor anunciar sus capacidades en orden de preferencia.

Longitud del par de claves

El agente S/MIME debe soportar el uso de pares de claves RSA para firmado y cifrado. Puede además soportar algoritmos adicionales y comunicarlo en su atributo "Capacidades".

La clave pública RSA presente en el certificado y su correspondiente clave privada deben tener una longitud mayor a 512 y menor a 1024 bits. Lo recomendado es que tengan al menos 768 bits ya que 512 bits resultan criptográficamente inseguros.

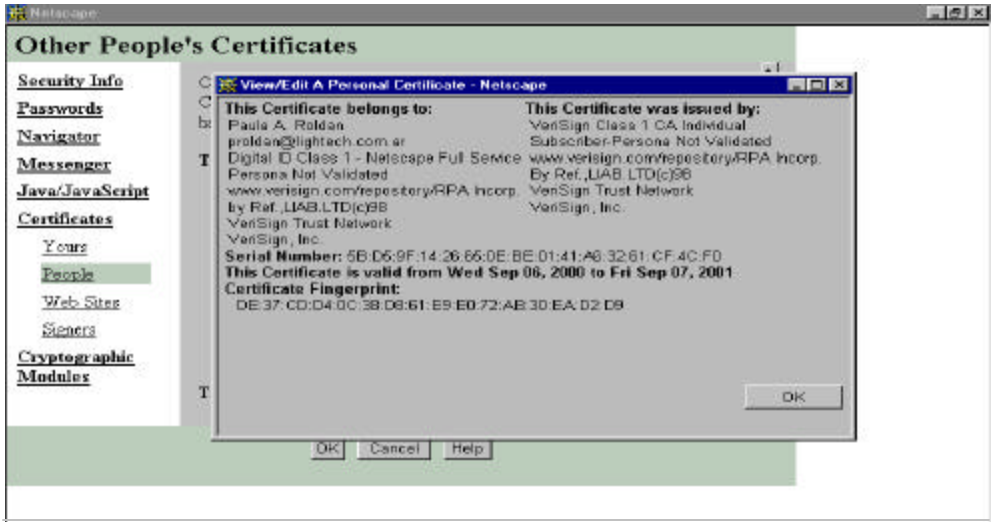
Certificados S/MIME

Un certificado S/MIME es esencialmente lo mismo que un certificado personal con la particularidad de que se requiere que la dirección de correo electrónico del sujeto del certificado aparezca en el campo "identificador único del sujeto" del certificado X.509 v3.

Un agente emisor debe asegurar que la dirección de mail presente en el encabezado FROM del mail se corresponde a la dirección de mail que figura en el certificado del firmante. Un agente receptor debe realizar dicho chequeo nuevamente y tomar acciones apropiadas si el mismo falla.

Un certificado S/MIME debe proveer al menos la seguridad de que el sujeto del certificado tiene la dirección de mail certificada en el mismo. Un certificado que provea esta única garantía se conoce como certificado de bajo grado de seguridad. Un certificado que provea alto grado de seguridad, en cambio, debe garantizar también la autenticidad de información adicional, tal como la identidad del sujeto del certificado. En ambos casos se garantiza la integridad del contenido del mensaje dada por el hecho de que la dirección de correo del origen no ha sido alterada durante su transmisión hacia el destino.

Los certificados S/MIME pueden obtenerse de una CA que emita dicho tipo de certificados.



Un agente S/MIME puede generar un par de claves, recoger información de identidad de un usuario, crear un requerimiento del certificado y enviarlo a la AC, la cual emite el certificado de acuerdo al requerimiento recibido y lo envía al agente.

S/MIME usa PKCS#10 para realizar los requerimientos de certificados y el formato PKCS#7 para retornar los certificados.

Si el agente S/MIME no cuenta con la capacidad necesaria para crear el par de claves o para requerir el certificado, entonces confía en otra aplicación, tal como un navegador, a la cual le delega la tarea de llevar a cabo el requerimiento.

Requisitos para verificar la firma

Un mensaje debe incluir un camino de certificados válido para el certificado del firmante, de modo tal que el receptor del mensaje pueda verificar la firma.

Además, un receptor debe acceder a la lista de revocación asociada más reciente para asegurarse de que el certificado usado para firmar el mensaje no ha sido revocado. Otra opción es que el agente emisor incluya una CRL en el mensaje firmado, acompañando al mensaje firmado y al canal de certificación enviados.

El hecho de incluir el certificado cuando se envía un mensaje significa un mayor gasto de espacio para su transferencia y su almacenamiento. Este problema se ve más agravado si se decide incluir información de revocación, ya que las listas suelen ser bastante largas.

Una alternativa, entonces, para incluir información del certificado y la CRL correspondiente en el mensaje firmado, es proveer al receptor una referencia a la ubicación en la cual se halla tal información.

La decisión de incluir o no los certificados en un mensaje firmado está relacionada con una variedad de aspectos que deben analizarse: la capacidad del receptor para almacenar certificados, la capacidad del receptor en cuanto al acceso a un Servicio de Directorio, etc.

Interoperabilidad

En términos de interoperabilidad, existen dos cuestiones a tener en cuenta:

- ✓ Los diferentes productos que implementan S/MIME deben ser interoperables entre sí. Por ejemplo, si un usuario utiliza Microsoft Outlook Express para enviar un mensaje encriptado, el receptor, que lo recibe usando el Netscape Messenger, debe ser capaz de procesar el mensaje.
- ✓ Otra consideración concerniente a la interoperabilidad es el hecho de que los productos S/MIME sean compatibles con los clientes de correo electrónico tradicionales. Para ello S/MIME ofrece dos alternativas para construir mensajes firmados, una de las cuales es compatible con los productos que no implementan S/MIME. Si un agente S/MIME envía un mensaje firmado usando este método, el cliente tradicional podrá mostrar el mensaje firmado, aunque no procesará la firma.

Especificaciones S/MIME v3

El grupo de trabajo IETF formado oficialmente en 1997 como "IETF S/MIME" tuvo como objetivo definir un estándar de Internet para los objetos digitalmente firmados y para los objetos encriptados, cuyo formato se basó en el protocolo de seguridad PKCS#7. El conjunto de especificaciones IETF S/MIME v3 incluye:

- ✓ Sintaxis para mensajes criptográficos (CMS).[2]
- ✓ Especificación de mensajes S/MIME v3.
- ✓ Especificación de manejo de certificados S/MIME v3.
- ✓ Servicios de seguridad adicionales[3] para S/MIME.

[2] Más conocida como CMS, del inglés Cryptographic Message Syntax.

CMS define una sintaxis estándar para la información presente en las comunicaciones criptográficas, al servicio de la protección del contenido de los mensajes. La sintaxis está basa en le versión 1.5 de PKCS#7, usada en los productos comerciales S/MIME que se conocen actualmente.

CMS define el formato de los "datos firmados" para proveer integridad, autenticación y no repudio a través del uso de firmas digitales. Define además la sintaxis de "datos ensobrados" para proveer confidencialidad a través de la encriptación del contenido.

La especificación PKCS#7 sólo soporta RSA como algoritmo de encriptado. La mayor diferencia entre las especificaciones PKCS#7 y CMS es que esta última incorpora campos a la sintaxis de "datos ensobrados" para conseguir independizarse de los algoritmos.

La especificación de mensajes S/MIME v3 define la codificación usada para proteger el contenido de los mensajes. Esta detalla opciones para encapsular los datos protegidos en los mensajes MIME, entre las que se encuentran las siguientes: multipart/signed, application/PKCS7-signature, etc. Las aplicaciones compatibles con MIME deben implementar esta especificación.

La especificación de manejo de certificados establece que los agentes S/MIME deben soportar los certificados X.509 v3, como bien se ha dicho anteriormente. También establece el uso de tales certificados y listas de revocación como herramientas de autenticación y manejo de claves.

Existen características adicionales que pueden ser ofrecidas por el estándar S/MIME v3:

- ✓ Confirmaciones firmadas, lo cual provee pruebas autenticadas para comprobar la efectividad de la entrega del mensaje.
- ✓ Etiquetas de seguridad, las cuales permiten calificar los datos con valores de sensibilidad (como por ejemplo: "secreto", "confidencial", etc).
- ✓ Listas de mail, brindando la posibilidad de expandir mensajes seguros a un conjunto de receptores predeterminado.

Creación de mensajes S/MIME

Cuando S/MIME quiere proveer seguridad a una entidad MIME utiliza la sintaxis para formato de mensajes criptográficos PKCS#7. Dicha entidad es insertada en un objeto PKCS#7. Este último es incluido luego en una nueva entidad MIME, la cual es finamente transportada a su destino. El receptor extrae el objeto PKCS7 de la entidad MIME que le envían, y una vez completado este paso, extrae la entidad MIME original.

PKCS#7 describe una sintaxis general para los datos que requieren operaciones criptográficas, como firma digital u encriptación. PKCS#7 define 6 tipos de contenido para manejar datos criptográficos, tres de los cuales son utilizados en S/MIME**:** *datos, datos firmados, datos ensobrados.*

El tipo de contenido *datos* se refiere a los datos arbitrarios, tales como archivos de texto ASCII, los cuales pueden no contar con una estructura de datos interna.

El tipo de contenido *datos firmados* especifica una estructura de datos que permite alojar la firma digital, todos los identificadores de algoritmos requeridos, los certificados asociados y las CRLs, así como toda la información necesaria relacionada con el firmante. Este tipo de contenido da lugar a los servicios de autenticación del origen, integridad y no repudio del origen.

El tipo de contenido *datos ensobrados* describe una estructura de datos que incluye datos encriptados usando una clave secreta, la clave secreta es encriptada usando la clave pública RSA de cada receptor; se incluyen también los identificadores de los receptores y los identificadores de los algoritmos utilizados. La utilización de este tipo de contenido permite implementar el servicio de confidencialidad.

[3] Cuya sigla es ESS, del inglés Enhanced Security Services.

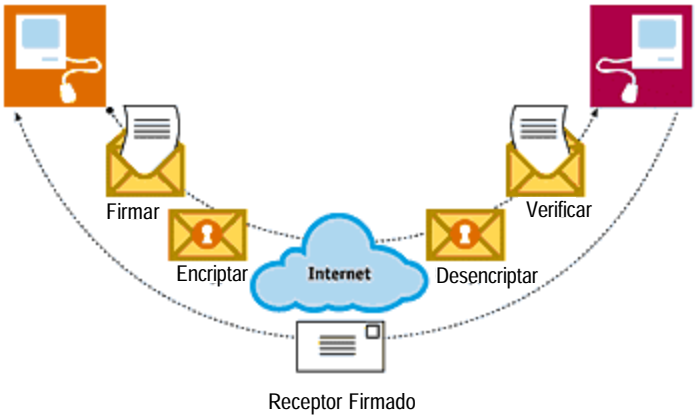
No podemos dejar de mencionar ciertos pasos necesarios de la transformación de la entidad MIME original en la entidad MIME finalmente transportada, pasando por la construcción del objeto PKCS#7.

El primer paso es la conversión del mensaje a una forma estándar, luego de lo cual es codificado para su posterior transferencia. Finalizada esta conversión, se aplican los servicios de seguridad cuya consecuencia es la creación del objeto PKC. Este objeto resultante es insertado en una entidad MIME, la cual conforma la entidad a transportar.

La forma estándar a la cual la entidad MIME a ser firmada o ensobrada es convertida, es única y no ambigua en el ambiente en el cual la firma digital es generada y en el ambiente en la cual la firma digital es verificada.

La transformación de las entidades MIME, en su mayoría de tipo textual, es muy importante dado que el texto tiene, frecuentemente, distintas representaciones en diferentes ambientes. Dicha transformación consiste en convertir cada fin de línea en un par de caracteres <CR><LF> y en la selección de un conjunto de caracteres para representar el texto. Si el texto no es convertido a un formato canónico antes de ser transportado por la red, su representación puede cambiar durante su tránsito o en el ambiente del receptor. Este hecho podría causar que la verificación correspondiente a la firma asociada dé un resultado negativo aunque la misma haya sido correcta y la información no haya sido alterada durante su tránsito.

La infraestructura subyacente al protocolo de transporte estándar *Simple Mail Transfer Protocol* (SMTP) puede manejar sólo texto codificado con 7 bits. Sin embargo algunos segmentos de la infraestructura de transporte manejan una codificación basada en 8 bits o datos binarios. Es por ello que es un requisito indispensable, para lograr un resultado exitoso, la transformación de los datos codificados con 8 bits o los datos binarios a una representación basada en 7 bits. Dicha codificación debe ser previa a la aplicación de operaciones criptográficas. Las codificaciones base-64 y Quoted-Printable son dos técnicas para transformar texto codificado con 8 bits o datos binarios a texto codificado con 7 bits.



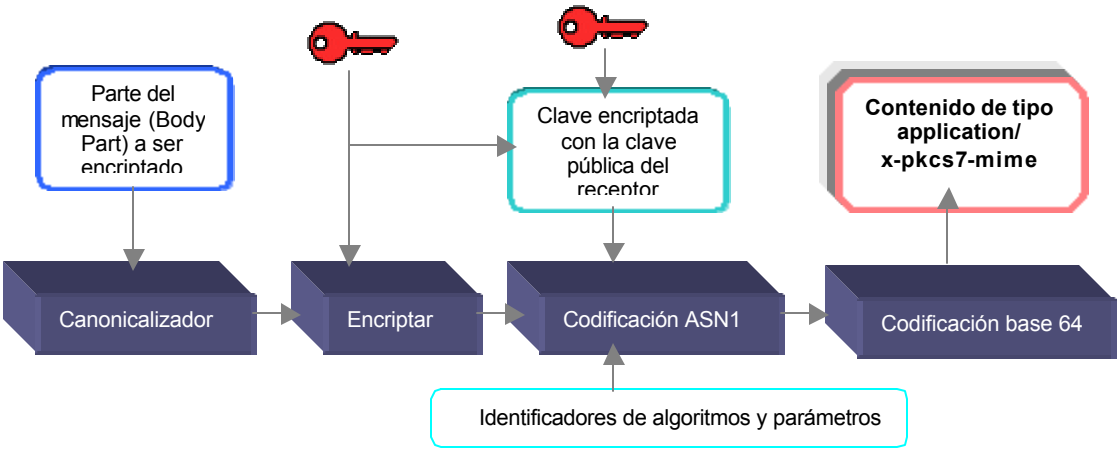
Ciclo de vida de un mensaje S/MIME

Mensajes encriptados

Luego de haber hecho la transformación antes mencionada, el agente S/MIME genera una clave secreta aleatoria, la cual es usada para encriptar el mensaje. El algoritmo usado para generar la clave secreta puede ser RC2, triple DES u otro. Dependerá de la aplicación si el usuario puede tener control sobre el algoritmo y la longitud de la clave a utilizar. Luego, el agente encriptará la clave secreta con la clave pública del receptor aplicando el método RSA.

El agente construye el objeto PKCS#7 de tipo *dato ensobrado* a partir del mensaje encriptado, de la clave secreta de encriptado, de los identificadores de los algoritmos y otros datos relevantes.

El objeto resultante es codificado respetando las reglas de codificación básicas[4] de la sintaxis ASN.1 y luego el mismo es transformado a una codificación base-64. Esta última transformación es necesaria debido a que el objeto PKCS#7 es un objeto binario que no puede ser transportado sobre la infraestructura tradicional SMTP de 7-bit.



Proceso de Encripción S/MIME

Finalmente el agente incluye el objeto PKCS#7 en una entidad MIME cuyo tipo de contenido es application/pkcs7-mime.

```
Content-Type: application/pkcs7-mime;
        smime-type="enveloped-data";
        name="smime.p7m"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename="smime.p7m"

MIAGSqLKJSDisdLLJDSA34lasdfjlLAJSFALS09SDLKJSAD87dsdadLKJSDcjl
laksJ
AkASDKolsd9asdf0a9AKSDJ65LKJ2MDCOiajYSTERJFLj8knKNSUYTVSkm
HY5gz
Jhsdio3JDSANmKykasdcoLKFNDoi8jkNCSEDOlaNIujcYADBEJAKD7lKSDFJ
lKAS0
```

Ejemplo de un mensaje S/MIME encriptado

Mensajes firmados

S/MIME soporta dos métodos alternativos para firmar datos MIME. El primero de ellos uso el formato *multipart/signed* y el tipo *application/pkcs-signature*, mientras que el segundo usa el formato *application/pkcs-mime* y establece el valor del parámetro *smime-type* a *signed-data.*

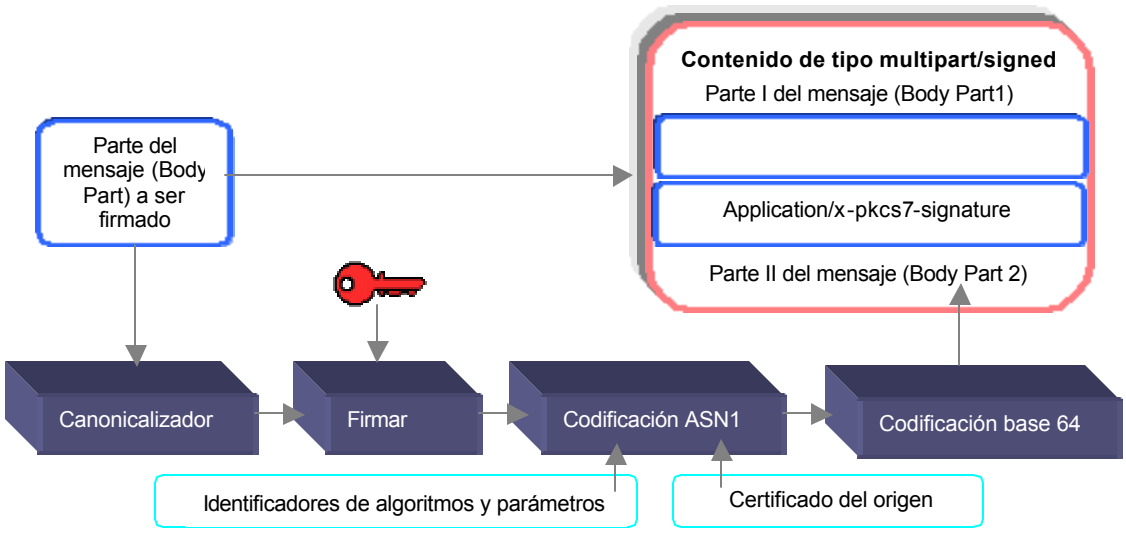
El formato *multipart/signed* es el método más aceptado dado que permite a los agentes que no son S/MIME compatibles manejar mensajes firmados y mostrarlos al usuario.

En el proceso de firma del mensaje se realizan varias tareas como se describen a continuación. El mensaje es llevado a una forma canónica y luego es codificado. Una vez realizados estos pasos, se aplica una función de hash usando el algoritmo SHA-1 y se obtiene un

[4] Las reglas de codificación básicas son llamadas VER: del inglés Basic Encoding rules.

digesto o resumen del mensaje de 20 bytes. Dicho digesto es firmado, encriptando el mismo con la clave privada del emisor del mensaje. Como resultado de este proceso surge la firma del mensaje.

La firma digital, los identificadores de los algoritmos, los certificados del emisor (esta última es opcional) y otros datos pertinentes son utilizados para construir un objeto PKCS#7 firmado. Dicho objeto es transformado a una estructura de datos ASN.1 y luego codificado. El objeto que resulta de tal codificación se inserta entonces en una entidad MIME cuyo tipo es application/pkcs-signature. Luego de lo cual, esta entidad MIME y el mensaje original serán incluidos en otra entidad MIME de tipo multipart/signed, para ser finalmente transportados.



Generación de la firma digital S/MIME usando el tipo multipart/signed

Un cliente de correo electrónico que no sea S/MIME compatible podrá mostrar el mensaje, aunque no será capaz de verificar la firma. Esto es posible porque la firma del mensaje es independiente del mensaje.

```
Content-Type: multipart/signed;
        protocol="application/pkcs7-signature";
        micalg=SHA-1
        boundary="----=_NextPart_000_0053_01BCF822.D6CF0800"

----=_NextPart_000_0053_01BCF822.D6CF0800
Content-type: text-plain;
        Charset="iso-8859-1"
Content-Transfer-Encoding= 7bit

Este mensaje está firmado digitalmente

----=_NextPart_000_0053_01BCF822.D6CF0800
Content-type: application/pkcs7-signature;
        name="smime.p7s"
Content-Transfer-Encoding= base64
Content-Disposition= attachment;
        Filename="smime.p7s"

MIAGSqLKJSDisdLLJDSA34lasdfjlLAJSFALS09SDLKJSAD8dfewrggscsdadLKJSDcjl
laksJ
AkASDKolsd9asdf0a9AKSDJ65LKJ2MDCOiaPJkLKMKYTERJFLj8knKNSUYTVSkn
HY5gz
Jhsdio3JDSANmKykasdcoLKFNDoi8jkNCSDOlaNIujcYADBEJ
AKD7lKSDFJlKAS0

----=_NextPart_000_0053_01BCF822.D6CF0800
Content-type: text-plain;
        Charset="iso-8859-1"
Content-Transfer-Encoding= 7bit
```

Ejemplo de un mensaje S/MIME firmado

Mensajes firmados y encriptados

Para componer un mensaje firmado y encriptado, un agente S/MIME puede firmar primero el mensaje y luego realizar los procesos inherentes para obtener el mensaje encriptado. También puede realizar las acciones en el orden inverso.

Esta flexibilidad se desprende de la sintaxis estándar PKCS#7, la cual permite que objetos PKCS con diferentes tipos de contenido sean incluidos en otros, arbitrariamente.

Si la primera acción es encriptar el mensaje, la firma será verificada antes de realizar el proceso de desencriptado. Además el mensaje se desencriptará únicamente si el proceso de verificación tiene un resultado positivo. En cambio cuando la primera acción es firmar el mensaje, contamos con la ventaja de que el ensobrado posterior, realizado a través del encriptado del mensaje, oculte la firma.

Otras alternativas: PEM, MOSS, PGP, X.400

Existen otras alternativas en la construcción de mensajes seguros. A continuación describiremos brevemente algunas de ellas:

PEM (Privacy Enhanced Mail)

Es un sistema para proteger el correo electrónico de posibles ataques. Constituye un estándar de Internet para proteger cualquier tipo de mensaje de correo.

Presenta los servicios de Integridad, Autenticación y No Repudio de Origen y permite opcionalmente la Confidencialidad. El funcionamiento se basa en garantizar la autenticidad mediante una firma digital generada a partir de un resumen criptográfico del mensaje. Si se ha solicitado la confidencialidad, se cifra todo mediante un algoritmo de clave simétrica. Esta clave se envía junto con el mensaje, protegida mediante la clave pública del destinatario, de forma que sólo él será capaz de descifrarlo.

Está preparado para poder soportar diferentes algoritmos, sin embargo en la actualidad tan sólo usa RSA, DES y Triple DES con CBC o ECB y hash MD2 o MD5.

PEM especifica una clave simétrica y una clave pública para el manejo de claves, sin embargo sólo la clave pública ha sido implementada.

También define su propio formato de mensaje para proteger mensajes de correo inseguros aplicando a los mismos servicios de seguridad. El formato del mensaje PEM está basado en mensajes de 7 bits, los cuales no son compatibles con S/MIME, razón por la cual no tienen gran aceptación comercial. [ref. 3]

PGP (Pretty Good Privacy)

Es, al mismo tiempo, una especificación y un producto de software de libre distribución, desarrollado por Phil Zimmermann.

Provee los mismos servicios de seguridad que PEM a través del uso de firma digital y encriptación.

PGP especifica su propio formatos de mensajes, aunque, a diferencia de lo que ocurre con PEM, estos formatos pueden ser convertidos a tipos MIME y de esta forma, pueden ser manejados por agentes de correo MIME compatibles.

Es importante mencionar que PGP define su propia infraestructura de clave pública y confía en los usuarios de correo electrónico para realizar el intercambio de claves y para establecer relaciones de confianza. El modelo de confianza se basa en confiar en una persona, la cual puede avalar el certificado de un tercero. Este mecanismo se conoce como “anillo de confianza”. En él, la responsabilidad de la validación de las claves públicas recae en los usuarios en los que uno confía.

PGP es una buena opción para una pequeña comunidad de usuarios que intercambian mensajes de correo electrónico en forma casual, pero no resulta adecuado para una comunidad numerosa de usuarios. [ref. 4]

MOSS (MIME Object Security Services)

MOSS es otro intento del IETF (Internet Engineering Task Force) para posibilitar el envío seguro de mensajes de correo electrónico a través de Internet.

En lugar de definir sus propios formatos de mensajes, MOSS emplea el formato MIME y especificó dos nuevos tipos de contenido MIME: el *multipart/signed* y el *multipart/encripted*, para proveer soporte para el manejo de mensajes encriptados y firmados digitalmente.

El hecho de carecer de una especificación detallada trae como consecuencia que diferentes implementaciones MOSS no puedan interoperar.

Al igual que ocurre con PEM, MOSS no tiene gran aceptación a nivel comercial y no fue jamás desarrollado en ambientes de gran escala. [ref. 5]

X.400

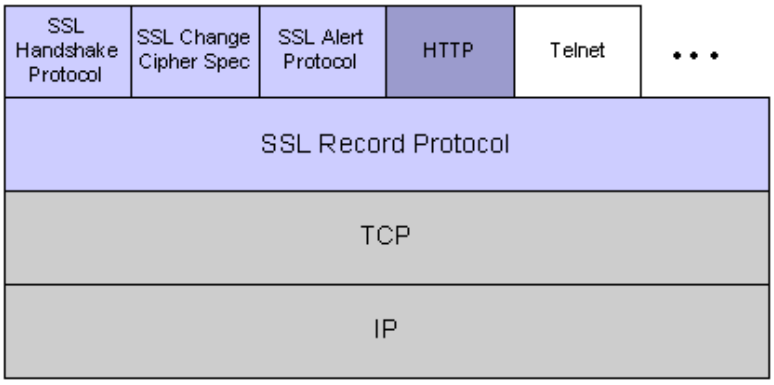
Luego del surgimiento del estándar SMTP (Simple Mail Transfer Protocol) [ref. 6] se creó otro estándar para el manejo de mensajes en INTERNET y se llamó X.400. Este estándar nunca llegó a imponerse en la INTERNET debido a su complejidad, a la falta de flexibilidad de las direcciones y a su falta de interoperabilidad con otros protocolos.

X.400 constituye un conjunto de protocolos estándares aplicables al correo electrónico. Fue desarrollado por la ITU (International Telecommunication Union), la ISO (International Organization for Standarization) y la IEC (International Electrotechnical Comisión). En 1988 se agregaron al mismo nuevas características que incluyen servicios de protección básica de mensajes y servicios de confirmación.

Seguridad en la WEB

❖ SSL

El protocolo SSL es un sistema propuesto y diseñado por Netscape Communications Corporation. Su objetivo es proteger las comunicaciones dadas en varios protocolos de aplicación que operan en Internet. Se ubica entre las capas de transporte y de aplicación del modelo OSI.



Este sistema de cifrado de datos dinámico permite, por ejemplo, el envío y recepción de información vía WWW de forma segura, impidiendo que intrusos puedan acceder a los datos que se transmiten de forma no autorizada.

Proporciona los servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico, típicamente DES o RC4 , y cifrando la clave de sesión mediante un algoritmo de cifrado de clave pública, típicamente RSA o DSS. La clave de sesión es la que se utiliza para cifrar tanto los datos que provienen del Servidor Seguro como aquellos que se dirigen hacia él.

Se genera una clave de sesión distinta para cada transacción, lo cual permite que, aunque dicha clave sea descubierta por un intruso en una transacción dada, la misma no sirva para descifrar futuras transacciones.

El algoritmo de resumen (hash) usado es MD5.

SSL consiste en dos subprotocolos, el *SSL Record Protocol* (Protocolo de registro SSL) y el *SSL Hanshake Protocol*(Protocolo de acuerdo SSL). El primero define el formato básico para todos los datos enviados en la sesión mientras que el segundo se encarga de negociar qué algoritmos de protección serán usados para autenticar al cliente y al servidor, para transmitir certificados y para establecer claves de sesión.

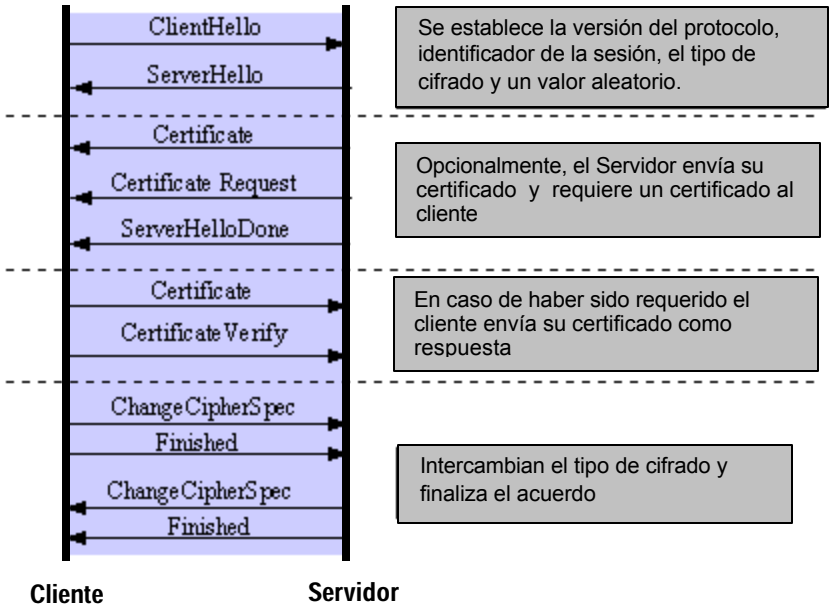
SSL provee cifrado de datos, autenticación de servidores, integridad de mensajes y, opcionalmente, autenticación de cliente para conexiones TCP/IP.

Cuando el cliente pide al servidor una comunicación segura (previa autenticación del servidor), el servidor abre un puerto cifrado, gestionado por el *SSL Record Protocol*, situado sobre TCP. Será el software de alto nivel, *SSL Hanshake Protocol*, quien utilice el *SSL Record Protocol* y el puerto abierto para comunicarse de forma segura con el cliente.

SSL Handshake Protocol :

Durante el SSL Handshake Protocol, el cliente y el servidor intercambian una serie de mensajes para negociar los parámetros de sesión. Este protocolo sigue las siguientes seis fases:

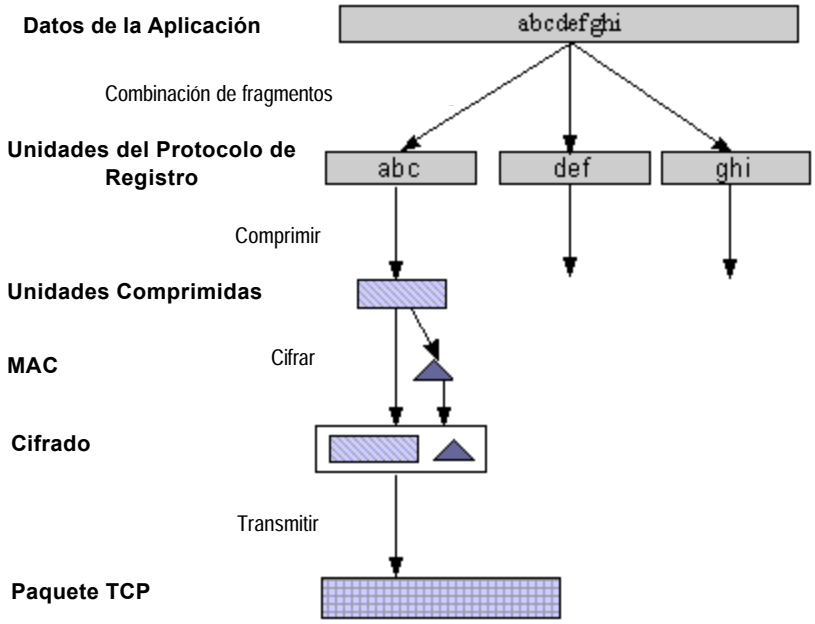
- ✓ La fase *Hello* (Hola), usada para ponerse de acuerdo sobre el conjunto de algoritmos para mantener la privacidad y garantizar la autenticación.
- ✓ La fase de *intercambio de claves*, en la que intercambia información sobre las claves, de modo que al final ambas partes comparten una clave maestra. Aquí se utiliza cifrado asimétrico o de clave pública.
- ✓ La fase de *producción de clave de sesión*, que será la usada para cifrar los datos intercambiados.
- ✓ La fase de *verificación del servidor*, presente sólo cuando se usa RSA como algoritmo de intercambio de claves, y sirve para que el cliente autentique al servidor.
- ✓ La fase de *autenticación del cliente*, en la que el servidor solicita al cliente un certificado X.509 (esta fase es opcional, sólo se realiza si es necesaria la autenticación del cliente).
- ✓ Por último, la fase de *fin*, que indica que ya se puede comenzar la sesión segura.



SSL Record Protocol

Este subprotocolo especifica la forma de encapsular los datos transmitidos y recibidos. La porción de datos del protocolo tiene tres componentes:

- ✓ MAC-DATA, el código de autenticación del mensaje.
- ✓ ACTUAL-DATA, los datos de aplicación a transmitir.
- ✓ PADDING-DATA, los datos requeridos para rellenar el mensaje cuando se usa cifrado en bloque.



❖ S-HTTP

El protocolo S-HTTP fue desarrollado por Enterprise Integration Technologies (EIT). Al igual que SSL, permite tanto el cifrado como la autenticación digital. Sin embargo, a diferencia de SSL, S-HTTP es un protocolo de nivel de aplicación.

La propuesta de S-HTTP sugiere una nueva extensión para los documentos, shttp, y el siguiente nuevo protocolo:

Secure * Secure-HTTP/1.1

Usando el mensaje GET, un cliente solicita un documento, le dice al servidor qué tipo de cifrado puede manejar y le dice también dónde puede encontrar su clave pública. Si el usuario con esa clave está autorizado a acceder al documento, el servidor responde cifrando el documento y enviándoselo al cliente, que usará su clave secreta para descifrarlo y mostrárselo al usuario.

Las negociaciones entre el cliente y el servidor tienen lugar intercambiando datos formateados. Estos datos incluyen una variedad de opciones de seguridad y algoritmos a utilizar. Las líneas usadas en la cabecera incluyen:

- ✓ Dominio privado S-HTTP, que especifica la clase de algoritmos de cifrado así como la forma de encapsulamiento de los datos (PEM o PKCS-7).

- ✓ Tipo de certificado S-HTTP, que especifica el formato de certificado aceptable, típicamente X.509.
- ✓ Algoritmos de intercambio de clave S-HTTP, que indican los algoritmos que se usarán para el intercambio de claves (RSA).
- ✓ Algoritmo de firma S-HTTP, que especifica el algoritmo para la firma digital (RSA o DSS).
- ✓ Algoritmo de resumen de mensaje S-HTTP, que identifica el algoritmo para proporcionar la integridad de los datos usando funciones de resumen "hash" (RSA-MD2, RSA-MD5 o NIST-SHS).
- ✓ Algoritmo de contenido simétrico S-HTTP, que especifica el algoritmo simétrico de cifrado en bloque usado para cifrar los datos. Los posibles algoritmos son:
  - o DES-CBC , DES-EDE-CBC, DES-EDE3-CBC, DESX-CBC, IDEA-CFB, RC2-CBC, RC4, CDMF
- ✓ Algoritmo de cabecera simétrica de S-HTTP, que proporciona una lista del cifrado de clave simétrica utilizada para cifrar las cabeceras. Las diferentes posibilidades son:
  - o DES-ECB, DES-EDE-ECB, DES-EDE3-ECB, DESX-ECB, IDEA-ECB, RC2-ECB, CDMF-ECB
- ✓ Mejoras de la privacidad de S-HTTP, que especifica las mejoras en la privacidad asociada con los mensajes, como firmar, cifrar o autenticar.

❖ SSL vs S-HTTP

S-HTTP y SSL utilizan aproximaciones distintas con el fin de proporcionar servicios de seguridad a los usuarios de la Red. Por servicios de seguridad se entiende: confidencialidad, autenticación, integridad, no repudio, control de acceso y disponibilidad.

SSL ejecuta un protocolo de negociación para establecer una conexión segura a nivel de socket[5]. Los servicios de seguridad de SSL son transparentes al usuario y a la aplicación.

Por otra parte, los protocolos S-HTTP están integrados con HTTP. Aquí, los servicios de seguridad se negocian a través de las cabeceras y atributos de la página. Por lo tanto, los servicios de S-HTTP están disponibles sólo para las conexiones de HTTP.

Dado que SSL se integra en la capa de sockets, también permite ser usado por otros protocolos además del HTTP (ej:FTP, Telnet), mientras que el S-HTTP está concebido para ser usado exclusivamente en comunicaciones HTTP.

[5] Socket: permite la comunicación entre procesos que se están ejecutando en máquinas remotas, pero integradas en una red. Los sockets corresponden al nivel 4(transporte) del modelo OSI.

❖ S/WAN

La iniciativa de S/WAN (Secure Wide Area Network) designa especificaciones para la implementación de IPSec[6], la arquitectura de seguridad para el Protocolo de Internet (IP) para asegurar la interoperabilidad entre firewalls y productos TCP/IP. El objetivo de S/WAN es usar IPSec para permitir a las organizaciones poder combinar distintos productos de firewall y TCP/IP para construir Redes Privadas Virtuales[7] (VPNs).

S/WAN soporta encripción a nivel IP, lo cual provee seguridad de bajo nivel, a diferencia de los protocolos de alto nivel como ser SSL y S-HTTP.

Para garantizar la interoperabilidad de IPSec, S/WAN define un conjunto común de algoritmos y opciones. Usa RC5 con claves de tamaño desde 40 bits (para exportación) hasta 128 bits. También puede usar DES.

[6] IPSec: es un grupo de extensiones de la familia del protocolo IP. Provee servicios criptográficos de seguridad que permiten la autenticación, integridad, control de acceso, y confidencialidad. Provee servicios similares a SSL, pero a nivel de redes, de un modo que es completamente transparente para sus aplicaciones y mucho más robusto. Puede usar cualquier protocolo IP sobre IPSec. Puede crear túneles cifrados (VPNs), o simple cifrado entre computadoras. Debido a que dispone de tantas opciones, IPSec es muy complejo.

[7] VPN(Virtual Private Network): Las redes privadas virtuales crean un túnel dedicado de un sitio a otro. Los datos se encriptan y se envían a través de la conexión, protegiendo la información.
La tecnología de VPN proporciona un medio para usar Internet como un canal apropiado
para comunicar los datos privados.

Implementación

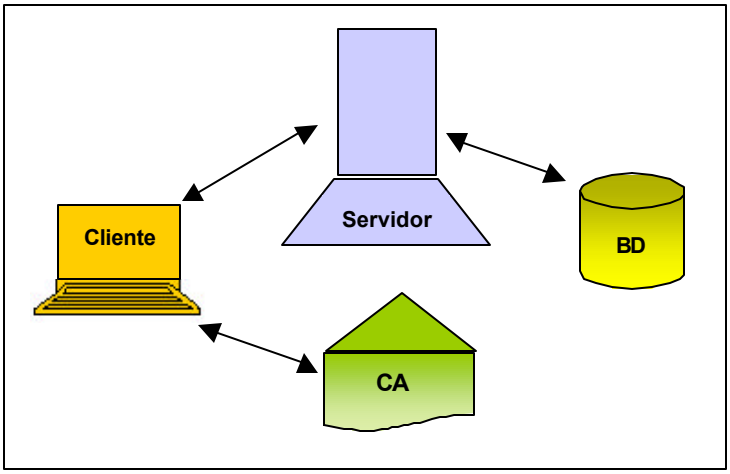
Una vez concluida la etapa de investigación y realizadas las pruebas necesarias para comprender el funcionamiento de la Infraestructura PKI, decidimos construir una arquitectura en la cual se pudiese integrar este modelo como soporte para garantizar la seguridad.

Objetivo

Así como en otras carreras, en la carrera de Licenciatura en Informática una de las últimas instancias, con el fin de obtener el título de grado, consiste en la realización del trabajo de grado, el cual presenta un procedimiento propio[1] (ciertos pasos administrativos por parte de los alumnos, del personal administrativo y docente; un régimen de trabajo diferente al resto de las materias de la carrera: no se exige que el alumno asista a clases en forma regular; existen determinados plazos que deben ser cumplidos, la necesidad de realizar ciertas operaciones que requieren la presencia física de la persona, etc).

Tomando como escenario esta instancia y aprovechando tanto las ventajas ofrecidas por el Servicio WWW como la posibilidad de efectuar transacciones seguras a través del uso de firma digital, desarrollaremos una aplicación cuyo objetivo será facilitar la interacción entre todos los participantes de este sistema respetando las normas existentes para el mismo.

Arquitectura del sistema



El modelo implementado está compuesto por:

- *Una Autoridad de Certificación*: Entidad encargada de emitir los certificados para crear el contexto legal de la Firma Digital.
- *Una interfaz de usuario*: se basará en HTML o extensiones, de manera de poder usarla en un entorno como Internet.

[1] La reglamentación que regula este proceso se encuentra disponible en http://www.info.unlp.edu.ar/TrabajoDeGrado.htm

- *Un Servidor de WEB*: el cual contendrá las páginas y scripts que implementarán la interfaz y relacionarán las partes del sistema. Dichos scripts serán los encargados de interactuar con la Base de Datos.

- *Una Base de datos:* Lugar donde se guardará y recuperará toda la información circunscripta como ser: registros de alumnos, estado de tesis, informes, etc.

Inicio del Desarrollo

Para el desarrollo de este modelo tuvimos que estudiar, analizar y probar distintas herramientas. A grandes rasgos, los pasos realizados fueron:

- Una de las primeras tareas que realizamos fue el análisis de distintas *Autoridades de Certificación* para poder comprender la forma de trabajo de las mismas y el rol que cumplen dentro de la Infraestructura PKI. Los productos que instalamos y analizamos fueron: Microsoft Certificate Server, Netscape Enterprise Certificate Server y APuN-CA. Se probó la funcionalidad de cada una de ellos, prestando mayor atención en lo que se refiere a: requerimiento y emisión de certificados, políticas, revocación de certificados y procedimientos para la verificación de los datos.
  Llegado el momento de implementar nuestra propia infraestructura PKI la opción elegida fue APuN-CA.

- Como *lenguaje de programación* del sistema elegimos JAVA. Además estudiamos distintos componentes para representar las interacciones del sistema. Entre ellos se encuentran los applets y los servlets. Además, dado que la interfaz del sistema está basada en web se estudiaron conceptos relacionados con HTTP y Javascript.
  Para poder implementar todos los aspectos relacionados con criptografía buscamos librerías que nos proveyeran las operaciones necesarias. Estudiamos también la API de Java Security al igual que las librerías de IAIK.
  Para poder implementar el envío de mensajes firmados fue necesario comprender el manejo de tipos MIME.

- Como *Servidor Web* elegimos desde un principio Apache, principalmente por ser un software de libre distribución y por su popularidad y robustez. Además tuvimos que investigar los requerimientos adicionales de dicho servidor para contar con soporte para transacciones seguras y manejo de servlets. Se probaron distintas instalaciones de los módulos requeridos debiendo compilarse el código fuente para que funcione con todos los módulos necesarios.
  También experimentamos los problemas de compatibilidad existentes entre las distintas versiones del Servidor Web y los distintos módulos.

- Para contar con un *repositorio de datos* instalamos una base de datos ORACLE.
  La selección de dicha base se centró en que Oracle es una de las Bases de Datos que ofrece mayor funcionalidad en muchas áreas, y es la marca que tiene mayor mercado.
  Luego, probamos la interacción de la Base de Datos con el sistema.

Descripción de las herramientas utilizadas

JAVA

JAVA comprende dos conceptos: un lenguaje de programación y una plataforma.

JAVA: el lenguaje

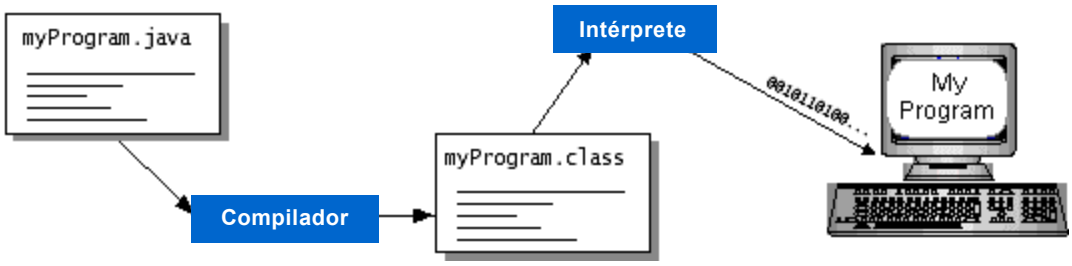
Java es un lenguaje de programación, independiente de la plataforma, desarrollado por SUN.

El lenguaje de programación JAVA es robusto y versátil. Permite a los programadores escribir programas en una plataforma y ejecutarlos en otra, así como crear programas que se ejecuten en el contexto de un navegador, desarrollar aplicaciones que corran del lado del servidor (para procesar un formulario HTML, por ejemplo), entre otras cosas.

Este lenguaje está caracterizado por ser simple, orientado a objetos, distribuido, dinámico, seguro y portable. Posibilita el desarrollo de aplicaciones seguras y de alta performance.

Cada programa JAVA es tanto compilado como interpretado. Esto se debe a que el compilador traduce un programa JAVA a un lenguaje intermedio, los bytescodes, semejantes a las instrucciones de ensamblador e independientes de la plataforma.

Luego el intérprete Java parsea y ejecuta cada instrucción Java bytecode. La compilación se lleva a cabo sólo una vez mientras que la interpretación ocurre cada vez que el programa es ejecutado.

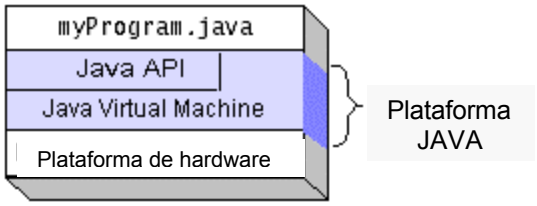


JAVA: la plataforma

La plataforma Java es sólo una plataforma de software, que se ejecuta sobre distintas plataformas de hardware.

La plataforma Java cuenta con dos componentes:

- ✓ La Java Virtual Machine (JVM): Es la base de la plataforma y puede ser incorporada a la mayoría de los sistemas operativos. Contiene el Intérprete JAVA.
- ✓ La Java Application Programming Interface (Java API): Es una amplia colección de componentes de software que proveen una amplia gama de funcionalidades, como GUI, I/O, etc. Esta API está agrupada en librerías de componentes relacionados. [ref.1]



Los tipos de programas más comunes que pueden ser creados en Java son las aplicaciones independientes y los applets.

Los applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

Existen otros tipos de programas que pueden escribirse en JAVA: Los servlets.

JAVA SERVLETS

Los servlets son piezas de código JAVA que adicionan funcionalidad al Servidor WEB de manera similar a aquélla en que los applets agregan funcionalidad a los navegadores WEB. A diferencia de los applets, los servlets no proveen una interfaz de usuario gráfica.

Esta tecnología permite entonces mejorar y extender la capacidad de dichos Servidores.

Los servlets están diseñados de manera tal de soportar el modelo de requerimiento/respuesta que es comúnmente usado en los servidores WEB. En este modelo, el cliente envía un mensaje al servidor realizando una solicitud. El servidor responde a dicha solicitud enviando un mensaje relacionado con el resultado del procesamiento del requerimiento. Por ejemplo, un servlet puede procesar los datos enviados a través de HTTPs usando un formulario HTML, incluyendo una orden de compra o un número de tarjeta de crédito.

Los servlets proveen un método independiente de la plataforma y basado en componentes para construir aplicaciones basadas en WEB, eliminando las limitaciones de performance existentes en los programas CGI[2]. Brindan una forma de generar documentos dinámicamente y son, al mismo tiempo, fáciles de escribir y de rápida ejecución.

A diferencia de los mecanismos propietarios de extensión del servidor (tales como la API del Servidor Netscape o los módulos de APACHE), los servlets son independientes del servidor y de la plataforma. Esto constituye una ventaja dado que nos otorga libertad a la hora de seleccionar los servidores, las plataformas y las herramientas con las que vamos a trabajar.

Los servlets tienen acceso a todas las APIs de Java, incluyendo la API JDBC (descripta luego en el presente capítulo) la cual provee acceso a bases de datos. Los servlets también tiene acceso a una librería de llamados HTTP específicos y reciben todos los beneficios del lenguaje JAVA que mencionamos anteriormente, a los cuales podemos sumar la reusabilidad.

El Java Servlet Development Kit (JSDK) provee tanto el motor de servlets como la "Java Servlet API".

El motor de servlets, conocido también como servlet engine o servlet container, se encarga de capturar los requerimientos, pasar el requerimiento al servlet, enviar las respuestas y manejar el ciclo de vida del servlet.

Existen dos variantes para que un servidor web cuente con soporte para servlets:

1. Instalar un módulo especial que se incorpora al servidor web y que permite ejecutar servlets. Actualmente encontramos disponible este soporte para servidores web como Apache Web Server, Microsoft IIS, iPlanet Web Server, y otros.
2. Instalar un servidor web en el cual el Servlet Container haya sido integrado.

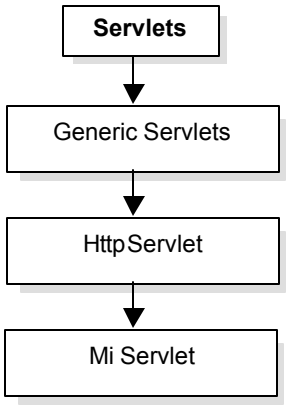
[2] CGI (Common Gateway Interface): Es un estándar para las comunicaciones entre los documentos WEB y los scripts CGI. Un script CGI es simplemente un programa que se comunica de alguna forma con los documentos WEB. Estos scritps se usan para lograr que un Servidor WEB pueda obtener datos de una BD, de documentos o de otros programas y presentar tales datos a los usuarios a través de la WEB.
El lenguaje de programación más popular para escribir scripts CGI es PERL.

La Java Servlet API puede ser usada para crear servlets que respondan a los requerimientos de los clientes.

Esta API está basada en un conjunto de interfaces JAVA que son provistas por los paquetes de extensiones de JAVA estándar (javax).

El paquete javax.servlet provee interfases y clases que posibilitan la programación de los servlets.

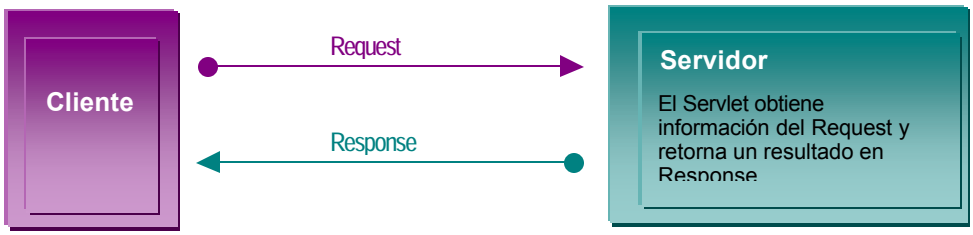
La Interfase Servlet



Herencia de la Interfase Servlet

La interfase Servlet declara métodos que manejan la comunicación entre el servlet y los clientes, aunque no los implementa. Los programadores serán entonces los encargados de proveer alguno o todos los métodos, en el momento de desarrollar el servlet.

Interacción con el cliente



Cuando un servlet acepta el llamado del cliente, recibe dos objetos:

- ✓ Un ServletRequest[3], el cual encapsula la comunicación desde el cliente hacia servidor.
- ✓ Un ServletResponse[4], el cual encapsula la comunicación desde el servidor hacia el cliente.

[3] La interfase ServletRequest permite al Servlet acceder a información tal como nombre de los parámetros pasados al cliente, protocolo usado por el cliente, etc.

[4] La interfase ServletResponse permite al Servlet retornar valores al cliente tal como una página HTML

Tanto ServletRequest como ServletResponse son interfaces definidas en el paquete javax.servlet.

Implementación de funciones criptográficas en JAVA

Más allá de los detalles anteriormente descriptos acerca de los formatos y la codificación de certificados, fue necesario comprender también, para lograr una implementación final del sistema propuesto, los aspectos relacionados con el desarrollo de programas que manejan certificados.

Como respuesta ante el requerimiento de estandarizar la criptografía y el manejo de claves, SUN nos provee de una API conocida como Java Cryptography Extension (JCE) la cual extiende la API Java Cryptography Architecture (JCA) brindando características adicionales como soporte para cifrado e intercambio de claves. Originalmente esta API sólo podía usarse en EEUU.

Java Cryptography Architecture

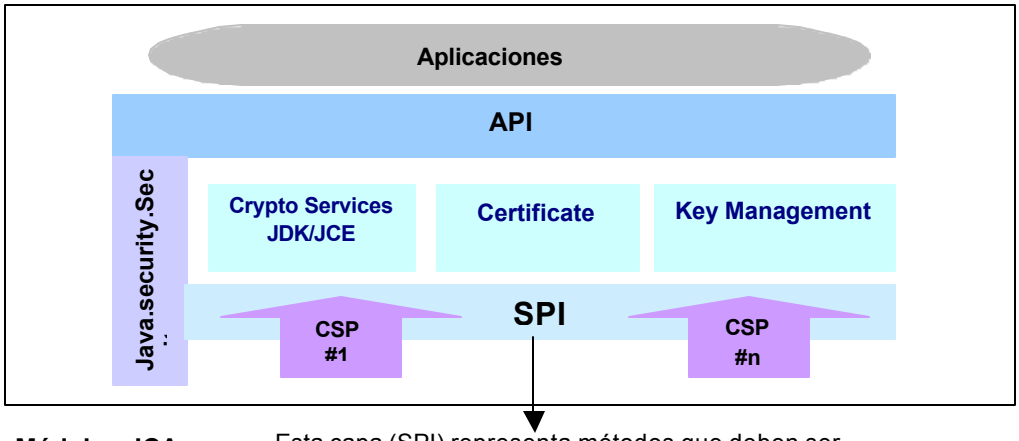
La API "JDK Security" es la API principal del lenguaje de programación JAVA, construida en base al paquete java.security.

Esta API fue diseñada para permitir a los desarrolladores incorporar distintos niveles de seguridad a sus programas.

La primera versión de JDK Security en JDK 1.1 introdujo la arquitectura llamada "Java Cryptography Architecture" (JCA), la cual define un modelo para construir código con funcionalidad criptográfica para la plataforma JAVA.

En JDK 1.1, la JCA incluyó APIs para el manejo de firma digital y el manejo de resumen de mensaje (o digesto). En JDK1.2 agrega 5 tipos más de servicios: creación de claves, manejo de los parámetros de los algoritmos, generación de los parámetros de los algoritmos, soporte para realizar conversiones entre los distintos formatos existentes para representar las claves y soporte para generar certificados y listas de revocación.

JCA incluye una arquitectura denominada "provider" que posibilita la existencia de múltiples implementaciones de criptografía y permite su interoperabilidad. Dicha interoperabilidad está sustentada en el uso de interfases comunes.



Módulos JCA

Esta capa (SPI) representa métodos que deben ser implementados por los proveedores de servicios crriptográficos

Un "provider" o proveedor JCA puede implementar la firma digital, el resumen del mensaje y algoritmos de generación de pares de claves.

La API JCE ("Java Cryptography Extensión") extiende JDK para incluir APIs que implementan: algoritmos de encriptación, intercambio de claves y códigos de autenticación de mensajes(MAC). JCE es distribuido en forma independiente.

El proveedor SUN

El *JRE (Java Runtime Environment)* desarrollado por SUN incluye un proveedor llamado "SUN", el cual incluye los siguientes paquetes:

- ✓ Una implementación del algoritmo de firma digital DSA.
- ✓ Una implementación de los algoritmos de resumen o hash MD5 y SHA-1.
- ✓ Un generador de pares de claves DSA.
- ✓ Un generador de parámetros del algoritmo DSA.
- ✓ Un manejador de parámetros del algoritmo DSA.
- ✓ Un "Key Factory" DSA.
- ✓ Una implementación del algoritmo propietario de generación de números pseudo-aleatorios "SHA1PRNG".
- ✓ Un "Certificate Factory" para certificados X.509 y CRLs.
- ✓ Una implementación del "KeyStore" llamada "JKS".

En nuestro trabajo, además de utilizar los métodos provistos por la API JCE de SUN, debimos incorporar la API IAIK-JCE.

IAIK-JCE

IAIK-JCE o *"IAIK Java Cryptography Extension"* es un conjunto de APIs e implementaciones de funciones criptográficas, incluyendo cifrado simétrico, asimétrico, cifrado de flujo y de bloque.

Provee una re-implementación de las Extensiones Criptográficas de Java. La arquitectura de IAIK-JCE sigue los mismos principios de diseño utilizado por JCA.

IAIK-JCE cuenta con su propio proveedor de seguridad, el cual ofrece una gran variedad de servicios criptográficos y algoritmos. La clase principal del paquete IAIK security provider es la clase IAIK del paquete iaik.security.provider. Este extiende la clase java.security.Provider para incorporar las implementaciones específicas de seguridad que facilita el proveedor IAIK.

La API está equipada con una librería para manejo de ASN.1. Soporta además los siguientes estándares PKCS: PKCS#1, PKCS#5, PKCS#7, PKCS#8, PKCS#10, PKCS#12.

También puede ser usada para manejar certificados X509. La arquitectura de certificados X509 y de CRLs implementada por IAIK-JCE se adapta fácilmente a la estructura de certificados impuesta por java.security. IAIK-JCE soporta todos las extensiones estándares existentes para certificados X509 V3 así como las extensiones más importantes de las CRLs X509 V3 y de los certificados Netscape.

Algunos de los paquetes contenidos en esta API son los siguientes:

PAQUETES	
iaik.asn1	Este paquete contiene interfases y clases que permiten representar objetos ASN.1 y diferentes tipos ASN.1.
iaik.asn1.structures	Este paquete está compuesto por clases que implementan distintos tipos ASN.1, cada uno de los cuales está representado por una estructura de datos que agrupa los distintos componentes que lo forman (Por ejemplo la clase Name, la cual representa la estructura de nombre X500 o Distinguished

	Name).
iaik.pkcs	Este paquete contiene la clase PKCS7CertList, la cual implementa el estándar PKCS#7 para manejar cadenas de certificados usadas por Netscape Navigator y Microsoft Internet Explorer. Además contiene las clases PKCSException (la cual sirve para agrupar todas las clases para manejo de excepciones incluidas en el paquete iaik.pkcs) y PKCSParsingException (para manejo de objetos PKCS que poseen una codificación DER inválida o que poseen características DER no soportadas).
iaik.pkcs.pkcs1	Este paquete contiene la clase RSACipher, la cual implementa el algoritmo RSA a través de la interfase Cipher.
iaik.pkcs.pkcs10	Este paquete está compuesto por la interfase CertRequest, la cual es usada por las clases que representan los requerimientos de certificados PKCS#10 y Netscape.
iaik.pkcs.pkcs12	Este paquete está compuesto por varias clases (entre las cuales se encuentran Atributtes, CertificateBag y KeyBag) las cuales sirven para representar claves privadas, certificados que respetan el formato definido en el estándar PKCS#12, atributos de los certificados PKCS#12, etc.
iaik.pkcs.pkcs7	Este paquete contiene interfaces y clases que posibilitan el manejo de los distintos tipos PKCS#7 existentes, como ser los tipos SignedAndEnvelopedData, SignerInfo, SignedData, etc.
iaik.pkcs.pkcs8	Este paquete provee dos clases que implementan la sintaxis estándar para la información de claves privadas PKCS#8 para claves privadas encriptadas y para el almacenamiento de las mismas.
iaik.security.cipher	Este paquete contiene varias clases que representan varios algoritmos de cifrado existentes, como ser: TripleDES, RC2, RC4, RC5, etc.
iaik.security.provider	Este paquete está compuesto por la clase IAIK, la cual representa la clase principal del paquete IAIK Security Provider para la API Java Security.
iaik.x509	Este paquete provee diferentes clases y excepciones que posibilitan el manejo de certificados X509, listas de revocación X509, certificados revocados, extensiones X509, información de la clave privada, entre otros.
iaik.x509.extensions	Este paquete está compuesto por varias clases cada una de las cuales representa alguna de las extensiones que se definen en el estándar X509. Entre ellas podemos mencionar: AuthorityInfoAccess, CRLNumber, ExtendedKeyUsage, IssuerAltName y SubjectAltName.
javax.crypto	Este paquete contiene clases relacionadas principalmente con la generación de claves privadas y con la creación de objetos cifrados, así como su encriptado y desencriptado.

IAIK-JCE Applet Edition

Es la versión para applets de la librería IAIK-JCE, la cual es absolutamente idéntica a la librería estándar IAIK-JCE salvo en los cambios (re-implementación de algunas clases) y características adicionales que incorpora por ser necesarios para que las clases funcionen correctamente en todos los navegadores compatibles con JDK 1.1.

IAIK-S/MIME

IAIK es una implementación de la versión 2 del protocolo S/MIME 100% desarrollada en JAVA.

Para cumplir con su propósito de extender el estándar MIME adicionando al mismo los servicios de seguridad de autenticación, integridad del mensaje, no repudio del origen y privacidad y seguridad de datos, IAIK-S/MIME opera junto a IAIK-JCE.

IAIK-S/MIME se basa fundamentalmente en el paquete iaik.pkcs.pkcs7 de IAIK-JCE. De esta forma, IAIK S/MIME aprovecha las ventajas de la implementación de stream de la librería PKCS7 posibilitando manejar grandes cantidades de datos sin que ello traiga aparejado problemas de memoria o de performance. Por esta razón, la clase SMimeSigned del paquete iaik.security.smime extiende la clase SignedDataStream del paquete iaik.pkcs.pkcs7, y la clase SmimeEncrypted extiende la clase EnvelopedDataStream.

IAIK-S/MIME puede ser incorporada fácilmente a la API javax.mail de SUN.

La misma soporta los siguientes tipos para los mensajes de correo electrónico:

- ✓ *signed* (multipart/signed; application/pkcs7-mime).
- ✓ *encrypted* (application/pkcs7-mime).
- ✓ *signed and encrypted*.
- ✓ *certs only.*
- ✓ *certificate requests* (application/pkcs10).

IAIK confía en el Java Activation Framework[5] para definir las siguientes clases como manejadores de contenido para los correspondientes tipos MIME:

Tipo MIME	Manejador del contenido
multipart/signed	iaik.security.smime.signed_content
application/x-pkcs7-signature	iaik.security.smime.signed_content
application/x-pkcs7-mime	iaik.security.smime.encrypted_content
application/x-pkcs7-signature	iaik.security.smime.signed_content
application/x-pkcs7-signature	iaik.security.smime.encrypted_content
application/x-pkcs10	iaik.security.smime.pkcs10_content
application/x-pkcs10	iaik.security.smime.pkcs10_content

Esta librería contiene los paquetes: iaik.security.smime y demo.

Algunas de las clases del paquete *iaik.security.smime*:

Clases	
encrypted_content	Esta clase implementa el manejador de contenido de datos (DataContentHandler) para el tipo MIME: "application/(x-)pkcs7-mime".
EncryptedContent	Esta clase puede ser usada para crear mensajes encriptados S/MIME en combinación con el paquete javax.mail package.
pkcs10_content	Esta clase implementa el manejador de contenido de datos (DataContentHandler) para el tipo MIME: "application/(x-)pkcs10".
PKCS10Content	Esta clase puede ser utilizada en combinación con el paquete javax.mail para crear mensajes de tipo S/MIME application/pkcs10.
signed_content	Esta clase implementa el manejador de contenido de datos

[5] JAF (Java Activation Framework). Para más información dirigirse a: http://java.sun.com/products/javabeans/glasgow/jaf.html

	(DataContentHandler) para los tipos MIME: *"application/(x-)pkcs7-signature"* y "*multipart/signed"*.
SignedContent	Esta clase puede ser usada en combinación con el paquete javax.mail para crear mensajes S/MIME signed.
SMimeBodyPart	Esta clase extiende la clase estándar MimeBodyPart del paquete javax.mail.internet adicionando el cálculo del resumen del mensaje.
SMimeEncrypted	Esta clase representa la parte del mensaje S/MIME que se encarga de transportar el objeto PKCS#7 EnvelopedData
SMimeMultipart	Extiende la clase MimeMultipart del paquete javax.mail.internet.
SMimeSigned	Esta clase representa la parte del mensaje S/MIME que se encarga de transportar el objeto PKCS#7 SignedData

Excepciones	
SMimeException	Esta excepción es levantada cuando ocurre un problema con S/MIME.

Conexión con la Base de Datos. Conceptos de JDBC

Se usó Oracle como soporte de Base de datos.

ORACLE es la empresa líder en el mundo en la venta de datos relacionados. El software de ORACLE corre en casi todas las computadoras, desde máquinas personales hasta supercomputadoras.

ORACLE 8 es la nueva generación de base de datos de ORACLE, la cual proporciona nueva funcionalidad y significativos avances sobre la versión ORACLE 7. En nuestra arquitectura se trabajó con ORACLE versión 8.0.6.

Como componente clave de la tecnología NCA (Network Computing Architecture), ORACLE 8 está diseñado para soportar configuración empresariales que manejen varios tipos de información bajo esquemas de múltiples usuarios.

Así mismo, ORACLE 8 cuenta con una amplia gama de extensiones que agregan funcionalidad específica, lo que permite que crezca la capacidad de manipulación y administración de datos.

Para que una aplicación pueda hacer operaciones en una base de datos, ha de tener el correspondiente Driver que conecte la aplicación con esta última. Las alternativas posibles son: la API JDBC [ref. 2] o la API ODBC [ref. 3].

JDBC (Java DataBase Connectivity) es una API de JAVA que permite ejecutar instrucciones SQL (Structured Query Language:Lenguaje estructurado de consultas), el cual es un lenguaje de alto nivel para crear, manipular, examinar y gestionar bases de datos relacionales.

La API JDBC es básicamente un paquete de JAVA (java.sql) que contiene un conjunto de interfaces y clases escritas en JAVA.

Se puede resumir en tres frases lo que hace JDBC:

1. Establece una conexión con una BD, que puede ser remota o no.
2. Envía sentencias SQL a la BD.
3. Procesa los resultados obtenidos de la BD.

❖ JDBC vs. ODBC

ODBC (Open DataBase Connectivity) es la interfase para conectarse con BD relacionales más usada por los programadores de aplicaciones. Las diferencia que presenta respecto de JDBC son:

- ✓ ODBC usa una interfase escrita con el lenguaje de programación C. Por lo tanto, dado que C no es un lenguaje portable, las aplicaciones JAVA perderían también automáticamente su portabilidad.

- ✓ ODBC se debe de instalar manualmente en cada maquina, en cambio, los drivers de JDBC, al estar escritos en JAVA son automáticamente instalables, portables y seguros.

Hay que decir también, que existen drivers puente entre JDBC-ODBC. Estos drivers traducen las llamadas de JDBC a ODBC permitiendo comunicarse con BD propietarias que no conocen de la existencia de JAVA. De esta manera por ejemplo podemos trabajar con una BD Access de Microsoft que usa ODBC, con el lenguaje JAVA.

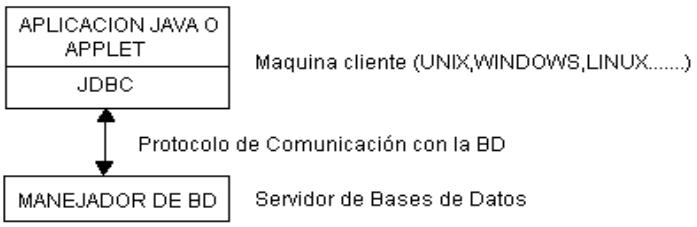
La API JDBC soporta dos modelos distintos de acceso a las BD:

- ✓ Modelo de dos capas.

- ✓ Modelo de tres capas.

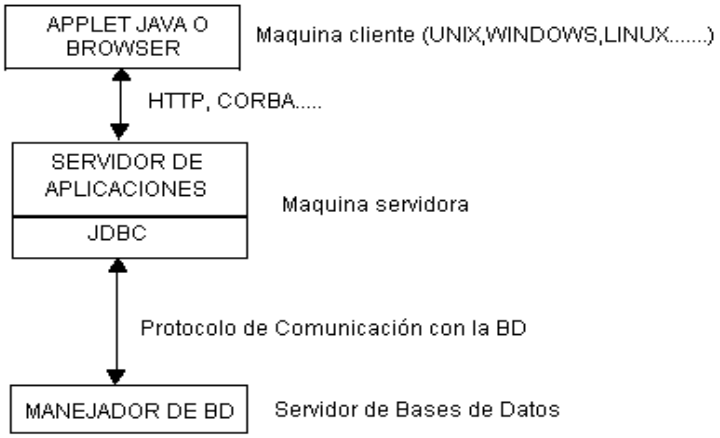
❖ Modelo de dos capas

En este modelo la aplicación JAVA o el Applet, se conectan directamente con la BD. Esto significa que el driver JDBC especifico para conectarse con la BD estará instalado en el sistema local (cliente). La BD puede estar en otra maquina y se accede a ella a través de la red. Esta configuración también se llama Cliente/Servidor. El programa cliente envía instrucciones SQL a la BD, y ésta las procesa y envía los resultados de vuelta al usuario.



❖ Modelo de tres capas

En este modelo, las instrucciones son enviadas a una capa intermedia que se encarga de enviar las sentencias SQL a la BD. El manejador de BD procesa las sentencias y retorna los resultados a la capa intermedia que se encarga de enviarlos al usuario.



Este modelo ofrece diferentes ventajas:

- ✓ El nivel intermedio mantiene el control del tipo de operaciones que se puede hacer en la BD.
- ✓ Los drivers JDBC para conectarse en la BD, no han de residir en la maquina cliente.

En nuestro sistema usamos JDBC como método para conectarnos a la BD. El modelo implementado es el de 3 capas de la API JDBC.

APACHE

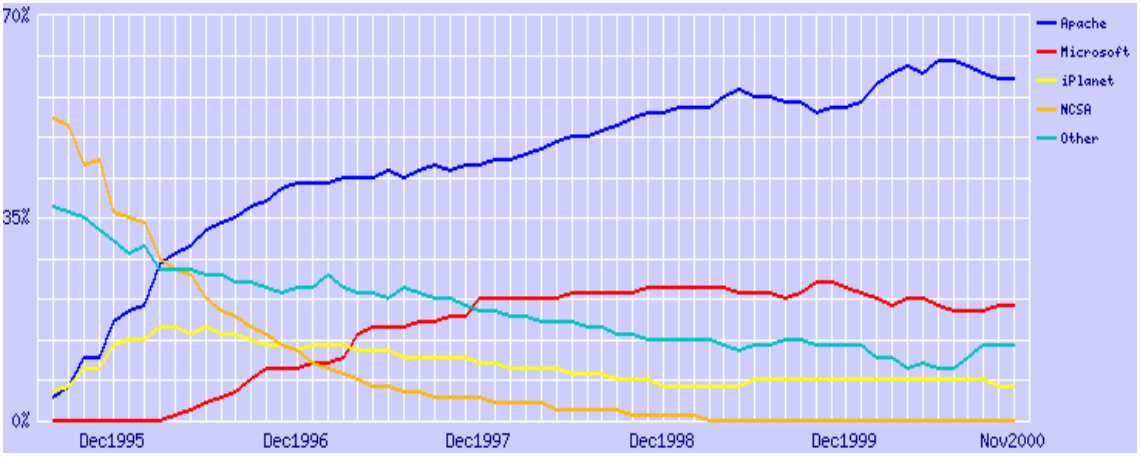
Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1). Entre sus características se destacan:

- ✓ Es multiplataforma.
- ✓ Es un servidor de web conforme al protocolo HTTP/1.1.
- ✓ Es modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- ✓ Es extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.

Apache se basó originalmente en código e ideas del servidor HTTP más popular: NCSA[6] httpd 1.3 (principios de 1995). Desde ese momento se convirtió en un sistema muy superior que puede rivalizar(y probablemente sobrepasar) casi cualquier otro servidor HTTP basado en UNIX en términos de funcionalidad, eficiencia y velocidad.

Desde el principio, ha sido completamente reescrito, e incluye nuevas características. Apache es, desde Enero de 1997, el servidor WWW más popular en Internet, de acuerdo al Netcraft Survey [ref. 4].

[6] NCSA httpd (National Center for Supercomputing Applications): es un servidor compatible con HTTP/1.0 para hacer disponibles hipertexto y otros documentos para navegadores.



Este servidor WEB fue creado para dar seguimiento a las preocupaciones de un grupo de proveedores WWW y programadores de HTTPD debido a que HTTPD no se comportaba como ellos querían que se comportara. Apache es un esfuerzo enteramente voluntario, completamente fundado por sus miembros, y no por ventas comerciales. Se desarrolla en forma abierta e incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.

Apache ha mostrado ser substancialmente más rápido que muchos otros servidores.

Además de correr en la mayoría de los sistemas UNIX del mundo, el servidor WEB Apache es soportado por las plataformas de Servidor de Windows: Windows NT y Windows 2000

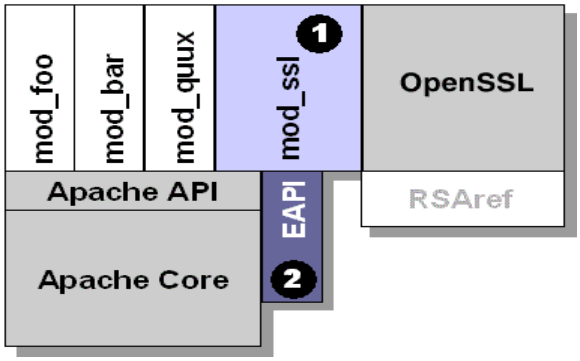
MODSSL

Este módulo provee criptografía fuerte para el servidor de Web Apache vía el protocolo SSL v2/v3 y TLS[7] v1.

Las principales características de mod_ssl son:

- ✓ Es un software de libre distribución.
- ✓ Proveee criptografia de 128 bits.
- ✓ Soporta los protocolos SSL v2/v3 y TLS v1.
- ✓ Soporta cifrados RSA y Diffie-Hellman.
- ✓ Se integra en forma dinámica con Apache a través de un API Extendido (EAPI).
- ✓ Provee comandos para la generación de certificados X.509v3.
- ✓ Soporta listas de revocación de certificados (CRLs).
- ✓ Brinda autenticación basada en certificados X.509 para cliente y servidor.

[7] TLS (Transport Layer Security): El protocolo SSL 3.0 es la base del TLS (Transport Layer Security) desarrollado por el IETF. Se podría decir que es SSLv3.1 ya que es compatible hacia atrás con todas las versiones de SSL y es SSL 3 pero mejorado. Difiere de SSL en que usa un conjunto un poco más amplio de algoritmos criptográficos



Este módulo [ref. 5] incluye al módulo SSL y un conjunto de actualizaciones para Apache agregando la EAPI (API extendida), la cual es un requisito esencial para mod_ssl.

Mod_ssl combina la flexibilidad de Apache y la seguridad de OpenSSL [ref. 6].

APACHE JServ

Para que un Servidor WEB pueda ejecutar servlets JAVA, debe contar con un motor de servlets que brinde el soporte necesario para capturar los requerimientos, enviar las respuestas, etc.

En el caso de Apache, una implementación del 100%JAVA del motor de servlets (servlet engine) es Apache JServ, totalmente compatible con la especificación *JavaSoft Java Servlet APIs 2.0.*

Este motor trabaja sobre cualquier Java Virtual Machine compatible con la versión 1.1 y tiene la capacidad de ejecutar servlets JAVA compatibles con la versión 2.0.

Con el fin de obtener una completa abstracción de los ambientes nativos, Apache JServ fue diseñado como una aplicación servidora que atiende requerimientos realizados usando un protocolo específico, conocido como Apache JServ Protocol[8]. Los módulos incluidos en esta implementación permiten al servidor WEB traducir los requerimientos de servlets y enviar los mismos al motor de servlets.

Actualmente, la distribución de Apache JServ contiene un único módulo para incorporar al Servidor de WEB Apache. Dicho módulo recibe el nombre de mod_jserv.

Algunas características que vale la pena enumerar son las siguientes:

- ✓ Al estar 100% desarrollado en JAVA, el servlet engine es una aplicación portable.
- ✓ Su funcionamiento es totalmente independiente del Servidor WEB, lo cual incrementa la estabilidad de este último y protege los recursos del sistema.
- ✓ Al usar el protocolo específico de red AJP, permite la creación de complejos ambientes distribuidos (los servidores web interactúan con motores de servlets que corren en diferentes máquinas que ellos).
- ✓ Al proveer un mecanismo de redirección de requerimientos permite un eficiente balanceo de carga de trabajo.

Además, los servlets pueden estar separados en diferentes contenedores lógicos, llamados zonas. Esto facilita la administración de los servlets y permite separar los contextos de los diferentes servlets.

También es importante destacar que esta aplicación cuenta con los beneficios inherentes a los productos de libre distribución.

[8] Especificación del Protocolo Apache JServ Protocol. [ref. 7]

Configuración del servidor WEB realizada para nuestra implementación

Para que nuestro servidor WEB pueda trabajar con conexiones seguras, instalamos el paquete mod_ssl, y para contar con soporte para servlets, incorporamos el paquete mod_jserv (estos paquetes están descriptos anteriormente en este capítulo).

Para la instalación del módulo SSL se requieren los siguientes paquetes: Openssl (The Open Source toolkit for SSL/TLS), RSAref (RSA reference implementation), MM (Shared Memory Library), Gzip (Compression Utility) y Perl (The Practical Extration and Reporting Language). Los paquetes: RSAref y MM son opcionales. El primero es utilizado para deshabilitar las librerías de RSA para los habitantes de EEUU, mientras que el segundo es utilizado por las librerías portables de Memoria Compartida en Apache/EAPI.

Los requerimientos para llevar a cabo la instalación del módulo JServ son: Java Runtime Environment, Java Servlet Development Key 2.0, Java Compiler, y ANSI-C Compiler.

Las versiones del software instalado son: Apache 1.3.12, , mod_ssl 2.6.2, OpenSSL 0.9.5, mod_JServ 1.0, JDK 1.2.1, JRE 1.2.1_04 y JSDK 2.0.

La elección de estas versiones se debió en su mayoría a problemas de compatibilidad entre los distintos componentes. En particular la última versión del módulo jserv probada (2.0) no reconoce ciertas variables de ambiente de APACHE-SSL necesarias en nuestro sistema.

Como plataforma se eligió UNIX, específicamente Solaris 2.6. El hardware utilizado es SUN, UltraSPARC 1.

Configuración del módulo mod_ssl

Mod_ssl incluye una serie de directivas [ref. 8] que permiten establecer el comportamiento del Servidor WEB respecto a las conexiones seguras usando SSL.

Para que el servidor pudiera autenticarse, tuvimos que especificar lo siguiente:

En el archivo de configuración del servidor httpd.conf, modificamos las siguientes directivas:

- ✓ *SSLCertificateFile*: especifica el camino completo al archivo correspondiente al certificado del servidor.
- ✓ *SSLCertificateKey*: indica el camino completo al archivo correspondiente a la clave privada del servidor.
- ✓ *SSLCertificateChainFile*: Especifica un archivo que contiene la concatenación de los certificados de las CAs que forman el camino (o la cadena) de certificación para el certificado del servidor.
- ✓ *SSLCACertificateFile*: especifica el camino de certificación indicando los certificados de las CAs que serán usados para autenticar al cliente o alternativamente un único archivo con formato PEM que contenga todos los certificados pertenecientes a la cadena de verificación.
- ✓ *SSLVerifyClient*: Indica si el cliente debe presentar o no un certificado al servidor. Los posibles valores son: *none* (no se requiere autenticación por parte del cliente), *optional* (el cliente puede presentar o no un certificado válido), *require* (el cliente debe presentar o no un certificado válido) y *optional_no_ca* (el cliente puede presentar un certificado válido de una CA en la cual el Servidor no confía).
- ✓ *SSLVerifyDepth*: Especifica la profundidad de la cadena de certificación. En el caso de que los certificados estén firmados directamente por una CA en la cual se confía, se establece el valor 1 para esta directiva.

El módulo SSL incorpora variables de ambiente CGI necesarias para el funcionamiento del protocolo SSL. Algunas de ellas brindan información acerca del certificado del cliente y otras acerca del certificado del servidor.

Las variables usadas en nuestra aplicación fueron las siguientes:

- ✓ *SSL_CLIENT_S_DN*: es el distinguished name del certificado del cliente.

- ✓ *SSL_CLIENT_S_DN_S*: es el subcampo Surname del ditinguished name del certificado del cliente. Nuestra CA determina en su política que el mismo indicará el rol del usuario dentro del sistema. El valor de ese campo es establecido por la CA.
  Esta variable es usada por el Servlet IngresoSistema, el cual determina la interfaz que presentará al usuario de acuerdo al rol que éste cumple en el sistema.
- ✓ *SSL_CLIENT_S_DN_CN*: es el subcampo "Common Name" dentro del distinguished name del certificado del cliente. Cuando se trata del certificado de un alumno, éste contiene el número de alumno mientras que cuando se trata del certificado de un profesor, éste contiene el DNI del profesor.
  El servlet ServletAlumno utiliza esta variable cuando necesita saber un dato del alumno, ya que el número de alumno es el identificador del alumno en la base de datos y a partir de él puede obtenerse cualquier información necesaria.
  Los servlets ServletJuradoResult y ServletProfeJurado usan esta variable para averiguar información respecto al profesor, ya que el DNI es el identificador del mismo en la base de datos.
- ✓ *SSL_CLIENT_CERT*: Contiene el certificado del cliente codificado en base 64. Es usado por el Servlet GuardoFirma para verificar la firma que el cliente efectuó sobre un formulario o sobre un documento.

Para poder usar las variables de ambiente CGI de OpenSSL es necesario incorporar las siguientes directivas: **SSLOptions, StdEnvVars** y **ExportCertData**. Las mismas están presentes en el archivo de configuración de Apache **httpd.conf**.

Configuración del módulo JServ

Luego de haber instalado el módulo JServ deben usarse varios archivos de configuración para que el mismo pueda brindar sus servicios.

En primer lugar, con el fin de incorporar al Servidor WEB el módulo JServ, es necesario incluir en el archivo de configuración de apache **httpd.conf** la siguiente línea:

Include /usr/local/jserv/etc/jserv.conf

Además debimos trabajar con los siguientes archivos de configuración:

- ✓ *JServ.conf:* Aquí se indica el archivo de propiedades usado por Apache JServ a través de la directiva *ApJservProperties***.** Y además debe incluir una línea con la directiva *ApJServMount* para indicar la forma en que cada zona será mapeada a determinada URL. Para ello la sintaxis correcta es la siguiente:
  ApJservMount camino protocolo://máquina:puerto/zona.

- ✓ *jserv.propeties:* Se relaciona con las propiedades que se especifican para el motor de servlets. En dicho archivo debe especificarse el intérprete de la JVM que se utiliza a través de la directiva *wrapper.bin.*

Luego se agregaron en el mismo todas las librerías utilizadas por los servlets a través de las directivas *wrapper.classpath*, como por ejemplo, para poder usar las clases de javamail:

wrapper.classpath=/export/home/root/javamail-1.2/mail.jar

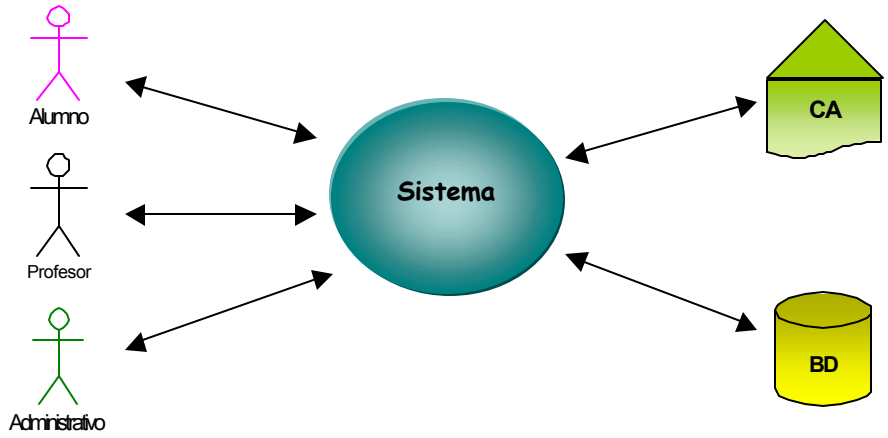
La utilidad principal de esta directiva es que el motor de servlets se cargue automáticamente al inicio. Eso se logra a través de la siguiente línea:

wrapper.classpath=/export/home/root/jsdk/JSDK2.0/lib/jsdk.jar

También pueden establecerse zonas[9] a través de la directiva *zones*. Cada zona tiene un archivo asociado donde se indican las propiedades de la misma.

- ✓ *Zone.properties*: es el archivo de configuración de la zona por defecto. Aquí se indica cuál es el directorio del servidor que contendrá los servlets pertenecientes a la zona. Para ello se usa la directiva *repositories*. Este es el lugar donde pueden establecerse ciertas características de los servlets como ser: alias, valores iniciales de sus parámetros, etc.
  El sistema usa la zona por defecto para contener los servlets.

Funcionalidad del sistema

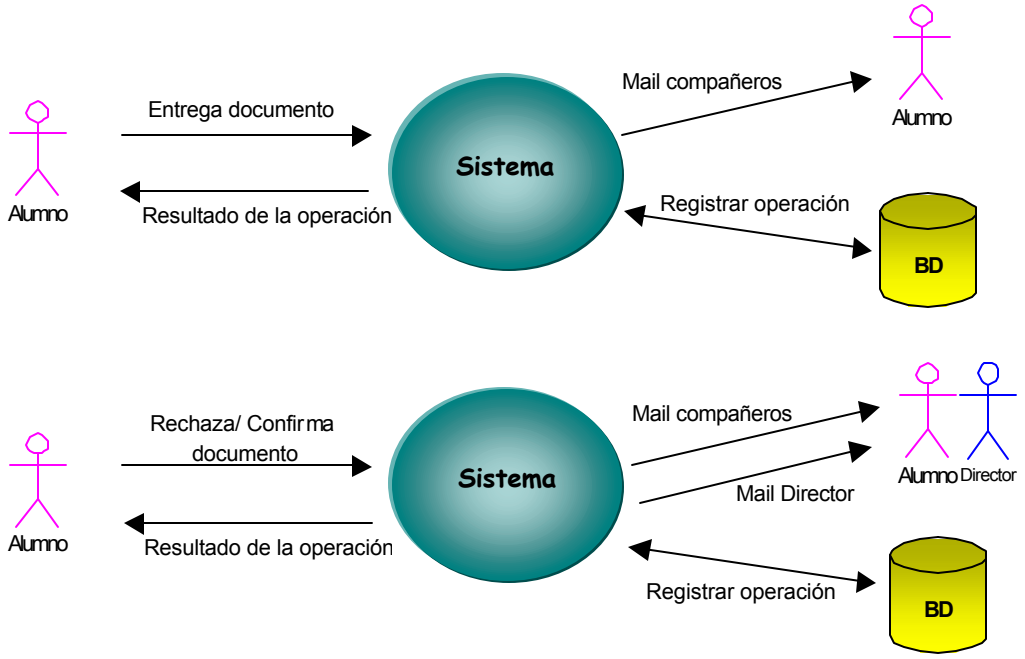


Los usuarios que pueden interactuar con el sistema son: los alumnos en condiciones de realizar el trabajo de grado, los profesores admitidos para dirigir o evaluar trabajos de grado y los administrativos que cumplan alguna función relacionada con la gestión del sistema.

Cada usuario tendrá disponible distintas opciones dentro del sistema de acuerdo al rol que posea.

Inicialmente, el alumno se reúne con su director y deciden el tema de la tesis además de la conformación del grupo. Recién entonces el director puede autorizar la emisión del certificado de los alumnos (previa verificación de los datos por parte del operador de la CA).

[9] Apache Jserv puede ser configurado para correr con una o más zonas y cada servlet puede correr dentro de una de esas zonas. Cada zona tiene asignado un único punto de montaje y un nombre. El punto de montaje corresponde a un directorio virtual. Esto tiene como ventaja el hecho de permitir agrupar servlets por propósitos de seguridad , para correr múltiples JVMs, etc.

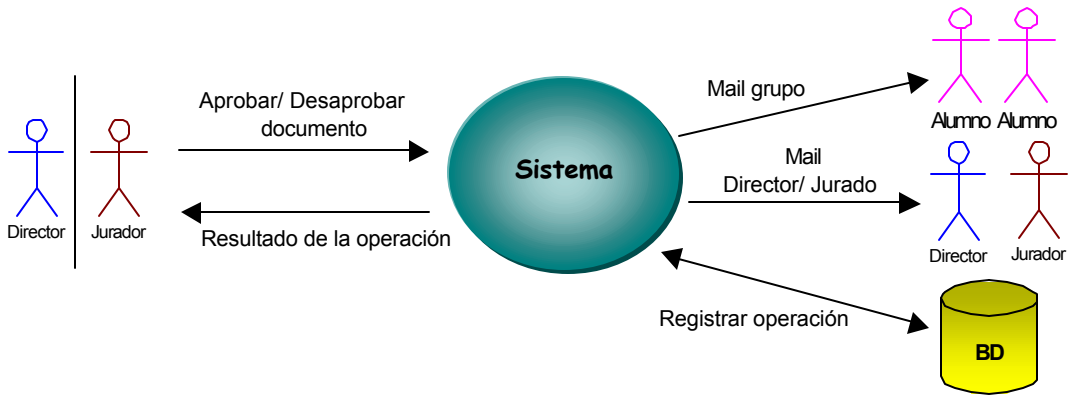


Cuando un alumno ingresa al sistema, según su estado se mostrará la pantalla correspondiente a la acción que podrá realizar en ese momento. Dicha acción puede consistir en realizar una entrega así como en confirmar o rechazar una entrega previamente realizada por alguno de sus compañeros. Cuando un alumno entrega un documento, los demás integrantes deben dar su consentimiento. Estos pasos deben cumplirse para cada entrega a efectuar.

La secuencia de informes a entregar es la siguiente:

- ✓ *Propuesta*
- ✓ *Informe de avance*
- ✓ *Trabajo final*

Recién después de que todos los miembros del grupo hayan consensuado respecto de la entrega, la misma podrá ser revisada por el director.



Si el director aprueba la entrega, entonces la misma estará disponible para la revisión del jurado. En caso contrario, los alumnos se deberán comunicar personalmente con el director para llevar a cabo las correcciones pertinentes.

Los alumnos son notificados a través del correo electrónico de lo que deciden tanto su director como los jurados.

Se debe respetar el mismo circuito para las sucesivas entregas.

Cuando un profesor ingresa al sistema, accede a una página inicial donde puede procesar los distintos trabajos organizados. Según el rol que él desempeñe en dicho trabajo (puede ser jurado o director para determinada tesis). Los trabajos se muestran bajo el siguiente criterio:

- ✓ *Trabajos como Director* son los grupos en los cuales el profesor tiene alguna entrega que corregir como Director.
- ✓ *Trabajos como Jurado* son los grupos en los cuales el profesor tiene alguna entrega que corregir como Jurado.
- ✓ *a Iniciar* esta opción permite al director ingresar un nuevo grupo al sistema. El director decide la conformación de cada grupo y el tema que se le asignará al mismo además de autorizar la emisión de los certificados de los alumnos pertenecientes a este grupo.

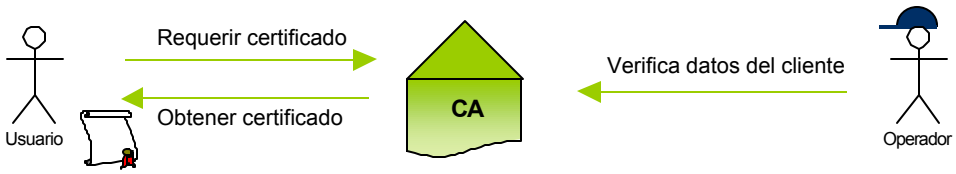
El personal administrativo se encarga de designar los miembros del jurado para cada uno de los trabajos de grado, consultar qué trabajos están en condición de ser defendidos y asignar una fecha y hora de exposición para los mismos).

Configuración de la Infraestructura PKI

Para poder implementar una CA es necesario crear una infraestructura de clave pública que la contenga. Para la definición de dicha infraestructura se utilizó APuN-CA.

Los componentes de nuestra infraestructura PKI son los siguientes:

- ✓ *Una CRA*: ésta incluye las funciones de CA y RA en un único módulo. En nuestro sistema la CA recibe el nombre de Autoridad de Certificación de la Facultad de Informática. Nuestro sistema permite el acceso a los servicios brindados por la misma.
- ✓ *El operador de la CRA* : será representado por personal Administrativo.



Nuestra CA cuenta con un certificado auto firmado lo cual la define como CA Raíz.

Además la Política implementada por la CA determina que ésta únicamente emitirá certificados para profesores, personal administrativo y alumnos. Esta política establece reglas para cada uno de los perfiles antes mencionados:

Perfil de certificado	Reglas
Profesional *(En el sistema este tipo de certificados es emitido para los profesores)*	Datos requeridos: DNI, nombre y apellido, dirección de correo electrónico, teléfono, organización, facultad en la cual ejerce, y cargo que posee. Período de validez: 1 año. Algoritmo de firma: MD5 con RSA. Propósito: Las claves asociadas al certificado pueden ser usadas para: firma digital, cifrado de datos, garantía de no repudio, cifrado de claves y acuerdo de claves.
Alumno *(En el sistema, este tipo de certificado es emitido para los alumnos en condiciones de realizar su trabajo de grado)*	Datos requeridos: Número de alumno, DNI, nombre y apellido, dirección de correo electrónico y domicilio. Período de validez: 1 año. Algoritmo de firma: MD5 con RSA. Propósito: Las claves asociadas al certificado pueden ser usadas para: firma digital, cifrado de datos, garantía de no repudio, cifrado de claves y acuerdo de claves.
Administrativo *(En el sistema, este tipo de certificado es emitido para los administrativos que cumplan con alguna tarea relacionada a las tesis)*	Datos requeridos: DNI, nombre y apellido, dirección, teléfono. Período de validez: 1 año. Algoritmo de firma: MD5 con RSA. Propósito: Las claves asociadas al certificado pueden ser usadas para: firma digital, cifrado de datos, garantía de no repudio, cifrado de claves y acuerdo de claves.
Servidor WEB *(Este tipo de certificado será el que se instalará en el Sitio WEB de nuestro sistema)*	Datos requeridos: Nombre y requerimiento del certificado. Período de validez: 1 año. Algoritmo de firma: MD5 con RSA. Propósito: Las claves asociadas al certificado pueden ser usadas para: firma digital, cifrado de datos, garantía de no repudio, cifrado de claves y acuerdo de claves.

Cada política puede establecer también ciertas reglas con respecto al manejo de los certificados regidos por la misma, como ser:

Política / ***Regla***	**Profesional**	**Alumno**	**Administrativo**	**Servidor WEB**
Es necesario que la CA solicite un token para que el usuario pueda retirar el certificado		***X***		
El usuario es notificado de que su certificado fue emitido a través de un mensaje de correo electrónico	***X***	***X***	***X***	***X***
El operador puede modificar el período de validez del certificado			***X***	
El usuario puede solicitar la revocación de su certificado	***X***	***X***	***X***	***X***
La revocación de un certificado fuerza la emisión de una nueva CRL	***X***			***X***

No se permite más de un certificado con el mismo ***Distinguished Name.***	***X***	***X***	***X***	***X***

Estas políticas son utilizadas en la emisión de certificados, los cuales son usados en nuestro modelo durante la autenticación del cliente y del servidor llevadas a cabo en las transacciones realizadas a través de SSL.

El sistema implementado admite únicamente clientes que utilicen certificados emitidos por la Autoridad de Certificación de la Facultad de Informática. El certificado instalado en el Servidor también ha sido emitido por esta CA.

Es importante mencionar algunos aspectos relacionados con el proceso de autenticación que ocurre entre el cliente y el servidor como ser:

- Cuando un usuario se conecta al Sistema, el sistema puede saber quién ingresó al sistema y cuál es el rol que cumple en el mismo (profesor, alumno o administrativo) a través de los valores de ciertos atributos del certificado que el cliente utiliza.
- Los clientes deben confiar en la CA de la Facultad de Informática, para poder acceder al sistema. Esto significa que el navegador debe tener instalado el certificado de la CA. Si esto no ocurre, cada vez que el cliente se conecta, el navegador presentará distintas ventanas de diálogo (dependiendo del cliente utilizado como ser Nestcape o Explorer) a través de las cuales el usuario decidirá si confiar o no en dicho Servidor.
- Además, para poder autenticarse ante el Servidor, el cliente debe tener instalado el certificado del usuario que debe haber sido previamente solicitado a la Autoridad de Certificación de la Facultad de Informática.

La Interfaz del sistema

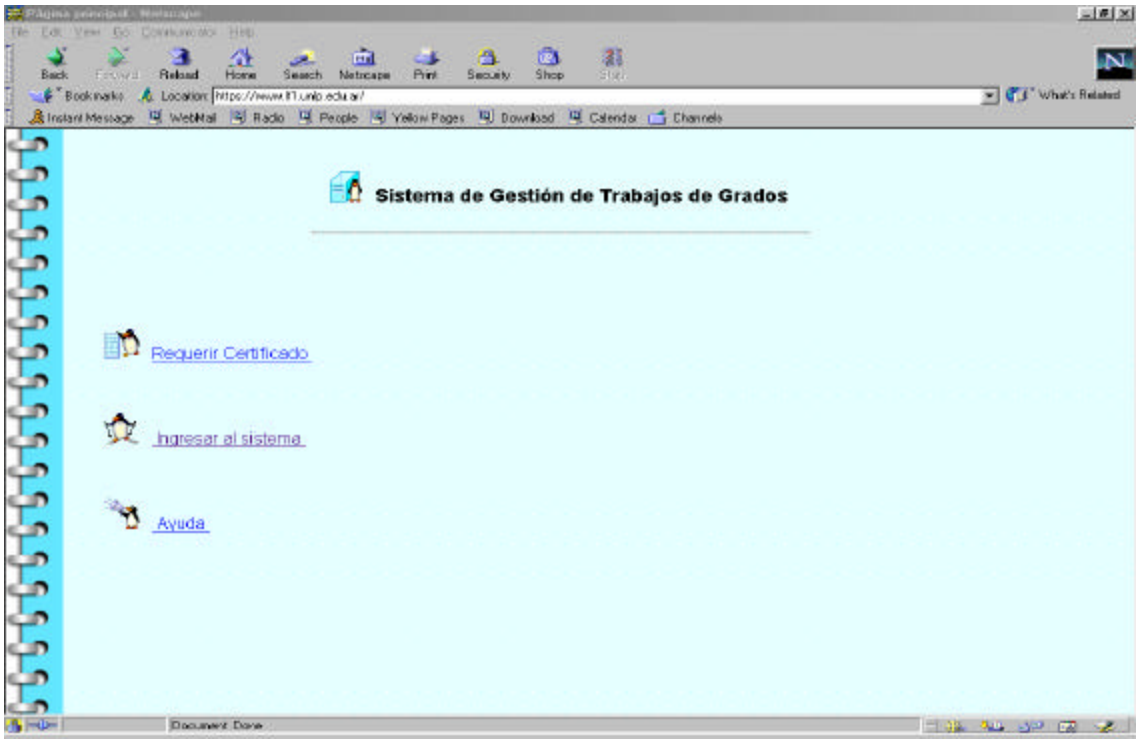
Está implementada a través de páginas HTML. Algunas de ellas son estáticas mientras que otras son construidas dinámicamente por servlets como resultado del procesamiento que éstos realizan.

Algunas páginas contienen un applet que será el encargado de brindar el servicio de *firma digital* a las operaciones.

Para ingresar al sistema, el usuario debe acceder a http://www.lt1.unlp.edu.ar[10]. Las opciones allí disponibles son:

- ✓ *Ingresar al sistema*
- ✓ *Requerir certificado* (Esta opción brinda acceso a la Autoridad de Certificación de la Facultad de Informática).
- ✓ *Ayuda*

[10] El acceso a la página principal no se lleva a cabo usando SSL ya que a partir de esta página también se accede al Sitio de la CA para solicitar el certificado, si el cliente aún no lo posee.

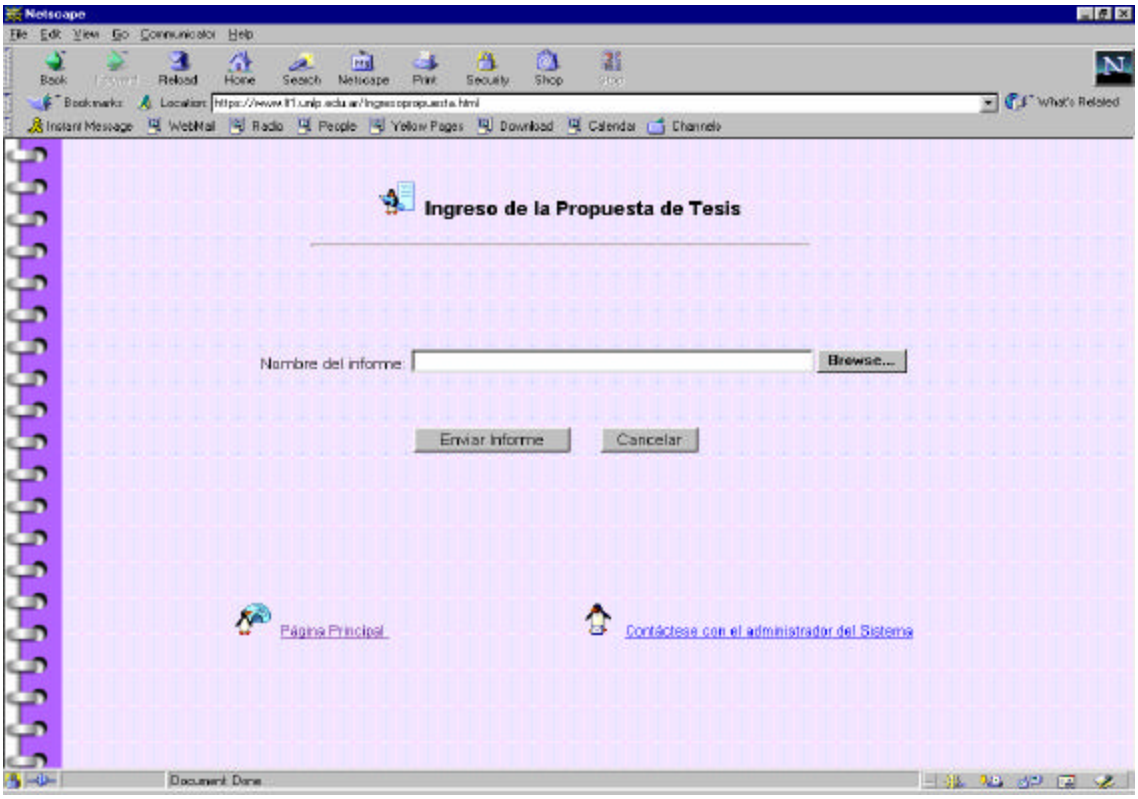


Pantalla inicial del Sistema

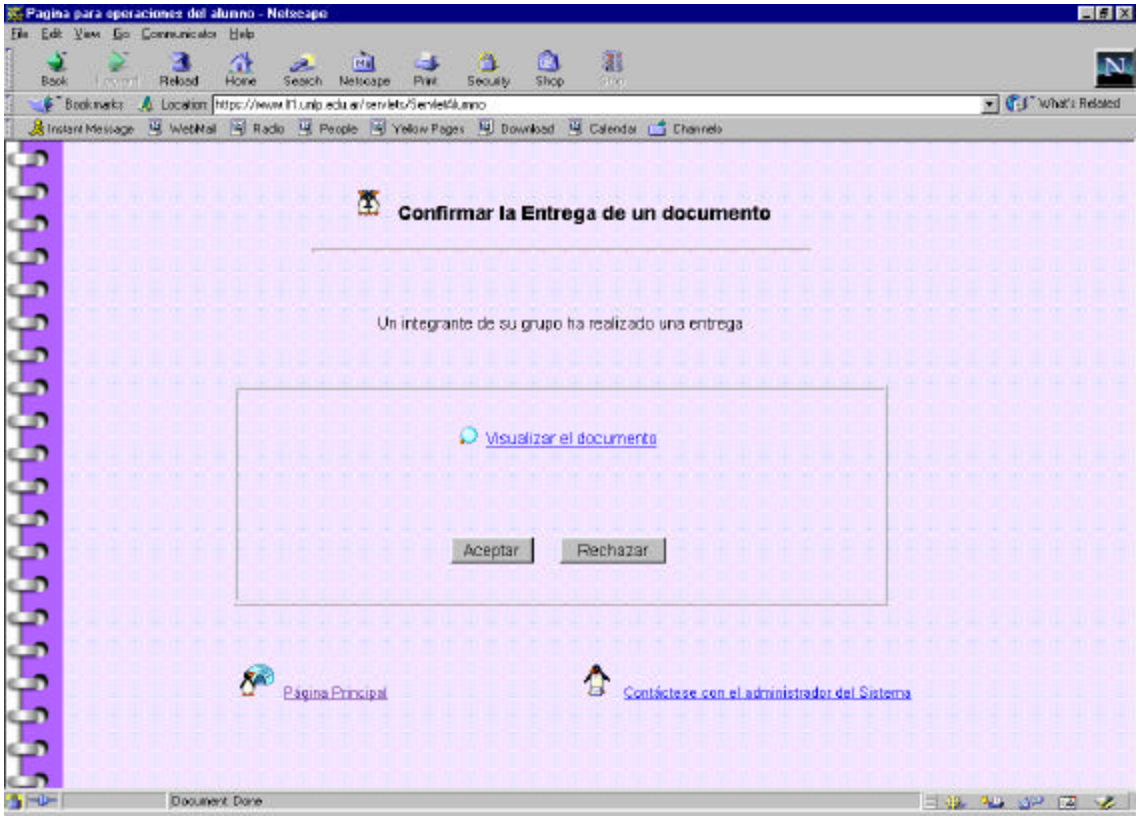
Los servlets que participan cuando un alumno se conecta al sistema son:

- ✓ *ServletAlumno*: cada vez que un alumno se conecta al sistema este servlet se encarga de averiguar a que grupo pertenece el alumno y, en base al estado del mismo, presenta al alumno sus posibles acciones: entregar un informe, confirmar o rechazar un informe entregado por otro alumno o esperar(esto significa que el alumno no puede realizar ninguna acción).

- ✓ *ServletAlumnoABM*: Este servlet tiene como función realizar todo el procesamiento inherente a la entrega de un informe por parte de un alumno, incluyendo el hecho de informar a los demás integrantes del grupo a través del envío de un mail firmado.



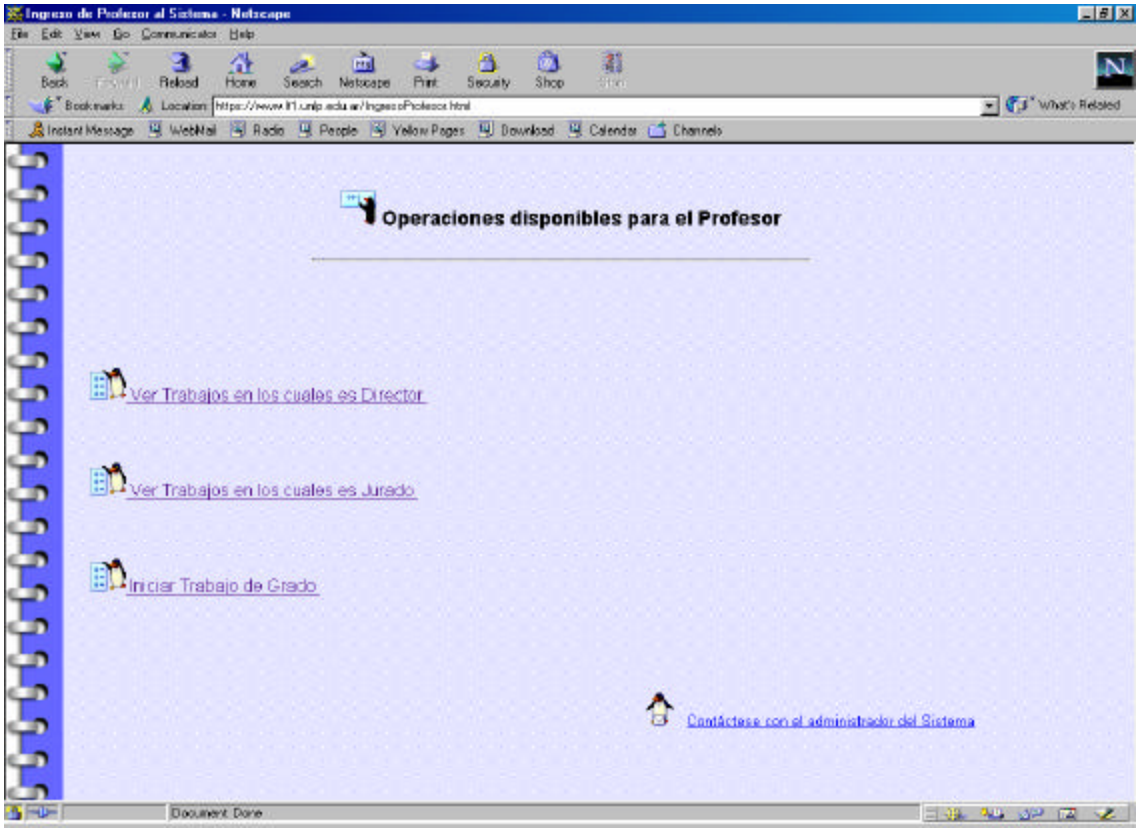
Pantalla del Sistema para realizar una entrega



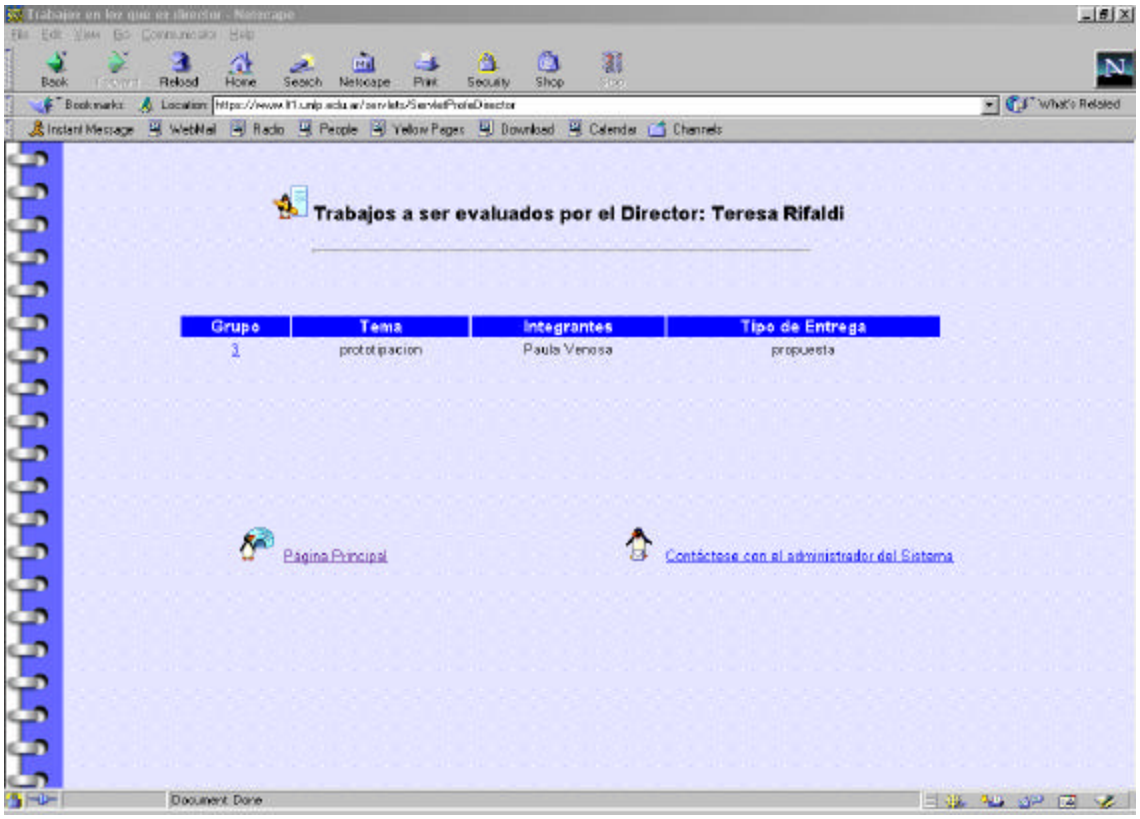
Pantalla del Sistema para confirmar una entrega

Los servlets que intervienen cuando se conecta un profesor al sistema son:

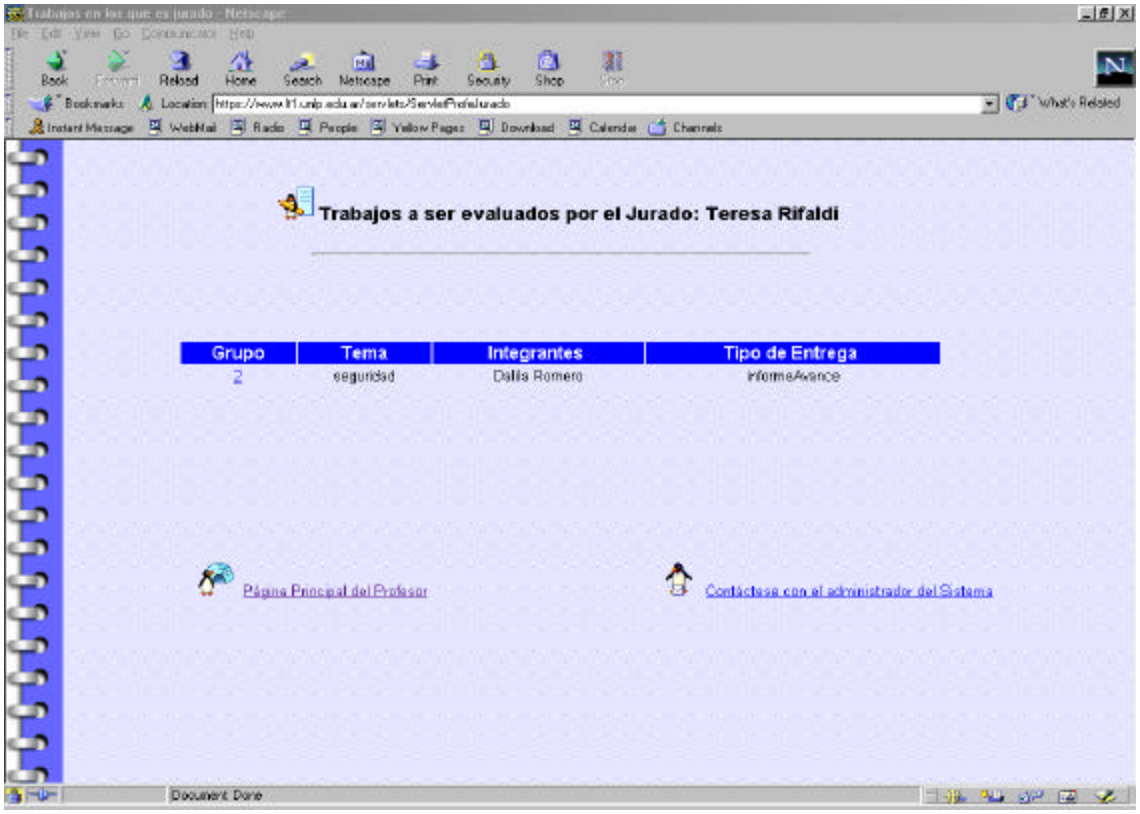
- ✓ *ServletDirector*: este servlet se encarga de armar un listado de todos los grupos de tesistas que son dirigidos por el profesor que está interactuando actualmente con el sistema y tienen algún trabajo a ser corregido por el mismo.
- ✓ *ServletDirectorAprob*: este servlet arma un listado de todas las entregas del grupo que se eligió y le permite al director *aprobar* o *desaprobar* el último trabajo entregado por dicho grupo.
- ✓ *ServletDirectorResult*: este servlet hace las actualizaciones correspondientes según la decisión hecha por el director y devuelve una página que notifica el resultado al usuario.
- ✓ *ServletJurado*: este servlet se encarga de armar un listado de todos los grupos de tesistas en los cuales el profesor conectado al sistema es Jurado y tienen algún trabajo a ser corregido por el mismo.
- ✓ *ServletJuradoAprob*: este servlet arma un listado de todas las entregas del grupo que se eligió y le permite al Jurado *aprobar* o *desaprobar* el último trabajo entregado por dicho grupo.
- ✓ *ServletJuradoResult*: este servlet hace las actualizaciones correspondientes según la decisión tomada por el Jurado y devuelve una página que notifica del resultado al usuario.
- ✓ *ServletAlumnoTemporal*: este servlet se encarga de crear una tabla temporal donde se van agregando los miembros del grupo que inicia una tesis.
- ✓ *ServletAgregarGrupo*: este servlet es el encargado de agregar un nuevo grupo de tesistas al sistema y asignarle un tema. Este utiliza la tabla temporal creada por el servlet anterior.



Pantalla Principal del Profesor



Pantalla del Sistema para que un profesor avale las entregas de las tesis que dirige



Pantalla del Sistema para que un profesor evalúe las tesis de las que es jurado

Otros servlets que forman parte del sistema son:

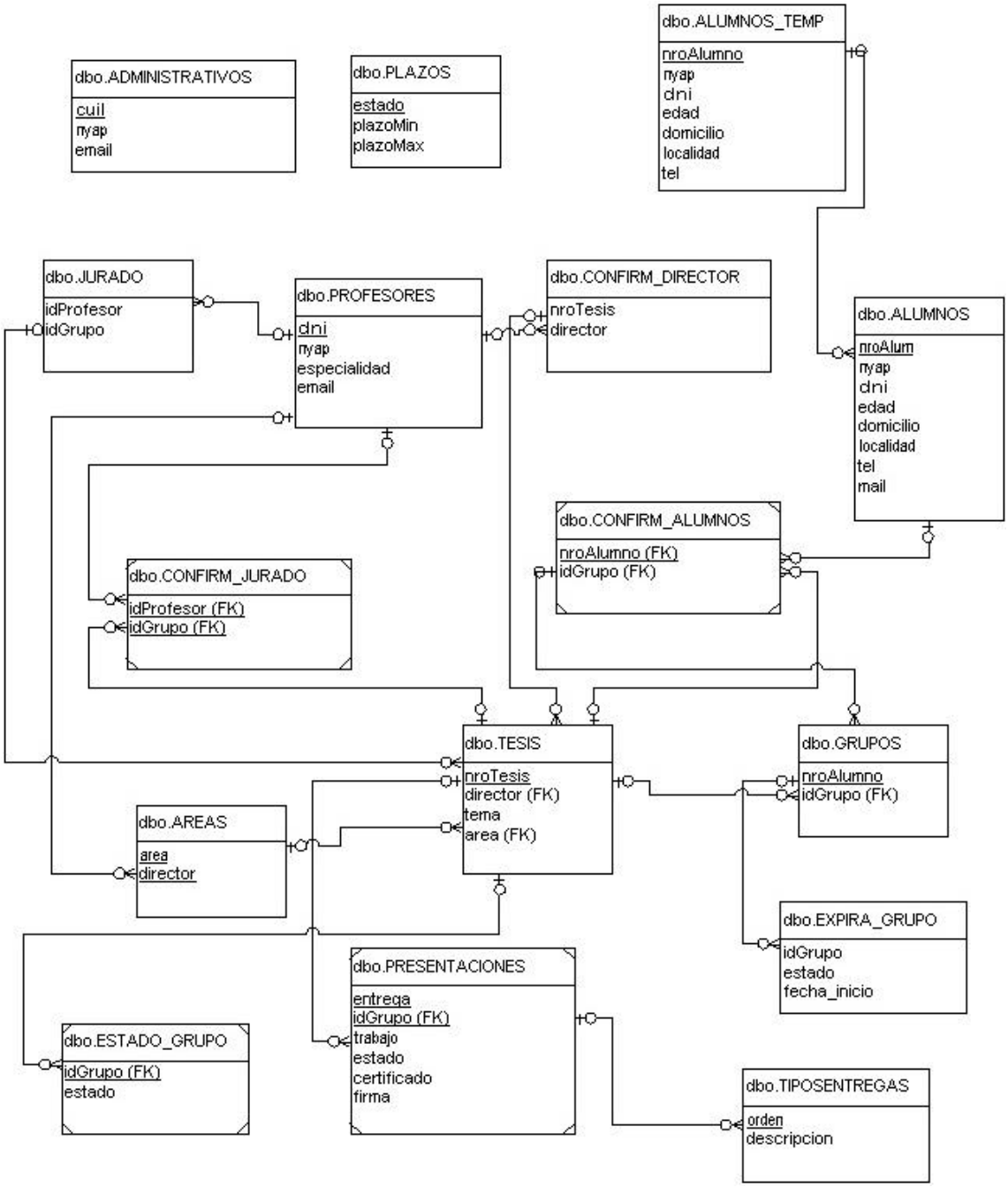
- ✓ *IngresoSistema*: este servlet se encarga de obtener el identificador y rol del usuario de su certificado y, en base a esta información, redirecciona al usuario hacia el servlet o la página correspondiente.

Las operaciones de firma se llevan a cabo a través de dos applets: AppletFirmaStr (el cual firma los formularios enviados por los usuarios) y el AppletFirmaDoc (el cual firma cada entrega realizada por los alumnos.

También fueron implementadas las siguientes clases para el funcionamiento del sistema:

- ✓ *EnvioMailFirmado:* Esta clase se encarga de armar un mensaje firmado para comunicar una acción realizada en el sistema a los usuarios relacionados con el trabajo de grado involucrado (Por ejemplo, cuando un alumno envía un informe, sus compañeros serán notificados de este hecho vía correo electrónico). Si bien la funcionalidad del sistema está completamente accesible a través de una interfaz web, la razón de notificar cada acción a través de mensajes de correo electrónico firmados digitalmente consiste en agilizar el completamiento de cada tarea
- ✓ *ConsultaBD:* Se encarga únicamente de realizar consultas a la base de datos requeridas por el sistema. Utiliza JDBC como mecanismo de conexión a la base de datos.
- ✓ *AbmBD*: Se encarga de llevar a cabo las modificaciones en la base de datos como consecuencia de las acciones de los usuarios en el Sistema. Estas modificaciones incluyen altas, bajas y actualizaciones de registros en la base de datos.

Modelo de datos



❖ Estados de un Grupo

Es un ciclo que se repite tantas veces como entregas tenga la tesis

0 => a enviar entrega (propuesta, informe, trabajo final)

1 => a esperar confirmación de los compañeros

2 => esperar aprobación del director y jurado

❖ Descripción de las Tablas

GRUPOS= (***grupo:*** identificadorG; ***nroAlum:*** identificadorAL)

ALUMNOS= (***nroAlumn***:identificadorAL; ***nombyAp:***string; ***edad:*** number; ***domicilio:*** string; ***localidad:*** string; ***tel:*** number; ***dni:*** string; ***email:*** string)

PROFESORES= *(**profesor:** identificadorP; **nombyAp:**string; **especialidad:** identificadorArea)*

ADMINISTRATIVOS= *(**admin:**identificadorAdm; **nombyAp:** string)*

AREAS= (***area:***identificadorArea; ***director:*** identificadorD)

TIPOS_ENTREGAS = (***orden:***integer; ***descripción:*** tipos[propuesta,informeAvance,trabajoFinal];)

TESIS= (***nroTesis:*** identificadorG; ***director:*** identificadorP; ***tema:***string; ***area:*** identificadorArea)

JURADO= (***nroTesis:*** identificadorG; ***profesor:*** identificadorP)

EXPIRA_GRUPO= (***nroTesis:*** identificadorG; ***estado:*** estados[0,1,2,3...]; ***finicio:*** fecha)

PLAZOS= (***estado:*** estado[0,1,2,3..]; ***plazoMin:***días; ***plazoMax:***días)

CONFIR_ALUMNOS= (***grupo:*** identificadorG; ***nroAlum:*** identificadorAL)

ESTADO_GRUPO= (***grupo:*** identificadorG; ***estado:*** number [0..2])

PRESENTACIONES= (***nroTesis:*** identificadorG; ***entrega:*** orden; ***trabajo:*** documento; ***certificado:*** string; ***firma:*** string; ***estado:***string[pendiente, aprobado])

CONFIR_JURADO=(***nroTesis:*** identificadorG; ***profesor:*** identificadorP)

Firma digital en el Sistema

Todas las operaciones que se realizan dentro de nuestro sistema son firmadas digitalmente.

Mecanismos de firma

La firma de cada operación es realizada en el cliente (Navegador) a través de un applet. El mismo se encarga de firmar determinada información (el objeto a firmar puede ser un documento o un formulario) y luego, la firma junto con la información es enviada al servidor. La implementación de la firma digital dentro del applet se debe a que la clave privada no debe ser transmitida a través de la red, independientemente que sea sobre un canal seguro (https). Finalmente en el servidor, el servlet receptor se encarga de verificar la firma y dependiendo del resultado de dicha verificación, las acciones correspondientes son registradas en la base de datos.

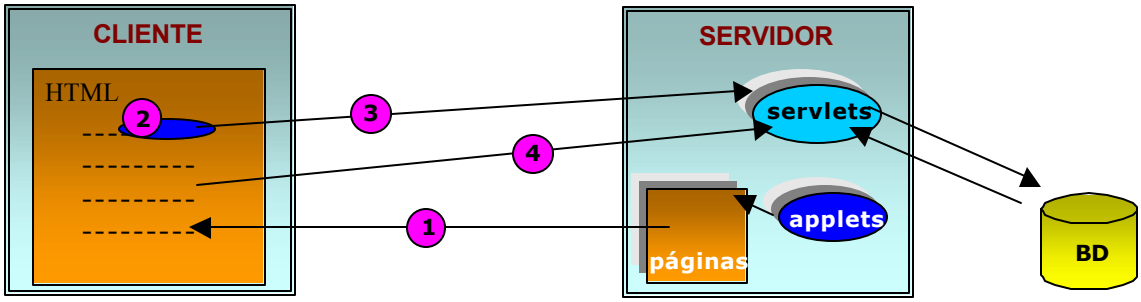
Para ello contamos con dos applets llamados AppletFirmaDoc y AppletFirmaStr.

⇒ *AppletFirmaDoc:* es el applet que se encarga de firmar documentos, por ejemplo: cuando un alumno ingresa al sistema algún informe.

⇒ *AppletFirmaStr:* es el applet que se encarga de firmar formularios, por ejemplo: cuando un director dá de alta un nuevo grupo de tesistas.

También implementamos una clase *VerificarFirma*, que es utilizada por los servlets para realizar la verificación de la firma.

Para lograr un mejor entendimiento acerca de la interacción y el funcionamiento de los distintos componentes que intervienen en la implementación de la firma digital, vamos a ejemplificar la instancia en donde*: "....el alumno ingresa su propuesta de tesis a través del sistema..."*



1. Se carga la página HTML en el cliente junto con el applet (en este caso la página "Ingresopropuesta.html" y el applet "AppletFirmaDoc").
   Utilizamos JavaScript para poder recabar la información que necesita conocer el método *firmar* del applet.
   Dentro de la función javascript *CargarDatos()* que se invoca la momento de enviar el informe, se realizan las siguientes acciones:

a) Pide al usuario su clave privada y la contraseña para desencriptar la clave privada y así poder realizar la firma.
b) Invoca al método firmar del applet pasándole como parámetro los:
   datos que se van a firmar, la clave privada y la contraseña
c) Llama al servlet correspondiente (ServletAlumnoABM) a través del método doPost() del mismo.

Aquí se detalla el código fuente de la página IngresoPropuesta.html:

```
<HTML>
<HEAD>
<script language="JavaScript">
phrompt=prompt;
snarkconf=confirm;

function cargarDatos()
{
      privatekey=phrompt("Ingrese el camino a la clave privada: ","C:");
      keypasswd=phrompt("Ingrese la password de la clave privada: ","");
      if (snarkconf("la clave esta en "+privatekey+" y la password es
          +keypasswd+"?")==true){
              document.appFirma.firmar(document.formu.informe.value,private
              key,keypasswd);
            document.formu.submit();
      }
}

</SCRIPT>
</HEAD>
<BODY>
<APPLET      CODE="AppletFirmaDoc.class"      CODEBASE="https://www.lt1.unlp.edu.ar/"
ARCHIVE="iaik_jce_full_ae.jar" WIDTH="10" HEIGHT="10" NAME="appFirma">
</APPLET>
<H2><CENTER>Ingreso de la Propuesta de Tesis</CENTER></H2>

<FORM ENCTYPE="multipart/form-data"
ACTION="https://www.lt1.unlp.edu.ar/servlets/ServletAlumnoABM" METHOD="POST"
NAME="formu">
<P>Nombre del informe: <INPUT TYPE=file NAME=informe>
<INPUT TYPE="button" NAME="Enviar Informe" ONCLICK="javascript:cargarDatos();">
<INPUT TYPE="reset" VALUE="Cancelar" NAME="Cancelar">
</FORM>
</BODY>
</HTML>
```

2. Cuando el alumno confirma la entrega se invoca al método *CargarDatos()* que se encarga de llamar al método *"firmar"* del applet *AppletFirmaDoc*. Dicho método es el que calcula la firma del documento que recibe como parámetro, con la clave privada que también recibe como parámetro.

   El applet utiliza los métodos de las librerías de IAIK y Java Security explicadas en secciones anteriores para realizar la firma digital.

Utilizamos la clase *iaik.java.security.Signature* para representar la firma, la misma requiere de los siguientes pasos para funcionar:

- ✓ En primer término se crea una instancia del algoritmo de firma deseado, por ejemplo:
  Signature md5_rsa = Signature.getInstance("MD5/RSA","IAIK");

- ✓ Además, el objeto *Signature* creado debe estar apropiadamente inicializado tanto para firmar un mensaje como para verificarlo, usando una clave privada (en el caso de la firma) respectivamente pública (en el caso de la verificación), por ejemplo:

  Inicializar para Firmar: md5_rsa.initSign(clavePrivada);
  Inicializar para Verificar: md5_rsa.initVerify(clavePública);

- ✓ Por último, si el objeto Signature ha sido inicializado para firmar, los datos a ser firmados son provistos al objeto Signature y a continuación se invoca al método *sign* que retorna la firma (como un arreglo de bytes codificado en formato DER) como resultado.
  En cambio, si el objeto Signature fue inicializado para verificar, entonces los datos a ser firmados son provistos al objeto Signature y a continuación se invoca al método *verify* utilizando como parámetro la firma asociada (como un arreglo de bytes codificado en formato DER). Este último método retorna verdadero si la firma calculada con los datos previamente provistos al objeto Signature es igual que la firma dada como parámetro y falso en caso contrario.

  Esta es la forma de cargar los datos a ser firmados y luego firmarlos:

  ```
  md5_rsa.update(data);
  byte[ ] signature = md5_rsa.sign();
  ```

  De esta manera se Cargan los datos a ser verificados y luego se verifican:

  ```
  md5_rsa.update(data);
  boolean resultado = md5_rsa.verify(signature);
  ```

A continuación se presenta el código fuente del AppletFirmaDoc donde se pueden visualizar los pasos explicados anteriormente:

```
public class AppletFirmaDoc extends Applet {
.............
public String firmar(String pathArchi, String pathPrivKey, String realPasswd) {

// Solo para Netscape. Para tener acceso al disco local donde se encuentra guardada la clave privada del usuario
 PrivilegeManager.enablePrivilege("UniversalFileAccess");

// Agrega el proveedor IAIK
iaik.java.security.Security.addProvider(new iaik.security.provider.IAIK());


 PKCS12 archi = null;
 Signature rsa = null;
 byte[] realSig = null;
```

```
 // crea un objeto PKCS12 a partir del archivo con la clave privada ingresado por parámetro
 try {
      archi = new PKCS12( new FileInputStream(pathPrivKey));
 }
 catch (PKCSException xxx) { }
 catch (FileNotFoundException fex) { }
 catch (IOException iox) { }

 // crea una instancia de la firma, utilizado MD5/RSA implementados por IAIK como algoritmo de hash/esquema de encripción
 try {
      rsa = Signature.getInstance("MD5/RSA", "IAIK");
       }
   catch (NoSuchAlgorithmException aaa) { }
   catch (NoSuchProviderException bbb) { }

// Desencripto la clave privada
char[] passwd = realPasswd.toCharArray();
try {
      archi.decrypt(passwd);
}
catch (PKCSException ppp) { }

// Obtengo la clave privada
iaik.pkcs.pkcs12.KeyBag clave=archi.getKeyBag();
iaik.java.security.PrivateKey privKey=clave.getPrivateKey();

// Inicializo la firma con la clave privada
try {
     rsa.initSign(privKey);
}
catch (InvalidKeyException ccc) { }
catch (java.lang.NullPointerException c) { }

PrivilegeManager.enablePrivilege("UniversalFileRead");
FileInputStream fis = null;

// se crea un FileInputStream a partir del archivo que se quiere firmar
try {
      fis = new FileInputStream (pathArchi);
}
catch (FileNotFoundException ffe){}

BufferedInputStream bufin = new BufferedInputStream(fis);
byte[] buffer = new byte[1024];
int len;

// carga el contenido del archivo que se quiere firmar al objeto Signature
try {
      while (bufin.available() != 0) {
           len = bufin.read(buffer);
```

```
        rsa.update(buffer, 0, len);
     };
    bufin.close();

       // luego de haber cargado todos los datos se procede a firmar
    realSig=rsa.sign();
}
catch (SignatureException xxx) { }
catch (IOException iox) { }

// Se crea una conexión desde el Applet hacia el ServletGuardoFirma para enviarle la
firma
URL servletURL=null;

// Se crea un objeto para poder comunicarse con el servlet
try {
    servletURL = new URL("http://www.lt1.unlp.edu.ar/servlets/ServletGuardoFirma");
}
catch (MalformedURLException mue) { }

URLConnection servletConnection = null;
// Se abre una conexión hacia el servlet
try {
     servletConnection = servletURL.openConnection();
}
catch (IOException e) { }

try {
     // para permitir escribir hacia el servlet
     servletConnection.setDoOutput(true);
     servletConnection.setAllowUserInteraction(false);
     servletConnection.setDoInput(true);
     // para asegurarse que se comunica con el servlet y no con la cache del navegador
     servletConnection.setUseCaches(false);

     // setear las propiedades de la información que se envia
     servletConnection.setRequestProperty("Content-type","application/binary");

     // Envia los datos (la firma) al servlet
        DataOutputStream dos = new
          DataOutputStream(servletConnection.getOutputStream());
        dos.writeBytes(lafirma);
        dos.close();
        servletConnection.connect();

} catch (IOException e) { }
.....................
}}
```

3. Una vez firmados los datos (en este caso un documento), el applet se conecta a un servlet llamado *"ServletGuardoFirma"* cuya única función es guardar el flujo de datos enviado por el applet (la firma del documento) en un archivo dentro del servidor.

```
public class ServletGuardoFirma extends HttpServlet {
.....
    // obtiene el mensaje enviado por el applet
    InputStream fromApplet = req.getInputStream();

    ConsultaBD consultaBase = new ConsultaBD();
    int idfirma = -1;
    try {
          idfirma = consultaBase.obtenerIdFirma();
    }
    catch (Exception oif) { }

    String str_idfirma=Integer.toString(idfirma);
    String nombreArchi="//usr//local//firmas//firma-"+str_idfirma+".enc";
    FileOutputStream fos = new FileOutputStream(nombreArchi);

    byte[] buffer4 = new byte[128];
    int len3 = fromApplet.read(buffer4);
    fos.write(buffer4,0,128);
    fos.close();
   fromApplet.close();
.....
} }
```

4. Finalmente, después de que el applet firmó el documento y la firma fue enviada al servidor, se llama al servlet *ServletAlumnoABM*, el cual se encarga de registrar la operación que se quiere efectuar (en este caso: enviar la propuesta de tesis) interactuando con la base de datos. En dicho servlet se instancia la clase *VerificarFirma* que es el que se encarga de afirmar que efectivamente el documento que llegó al servidor no haya sido alterado durante su transferencia desde el cliente.
   A continuación se muestra el código de la clase *VerificarFirma* junto con los métodos que implementa para realizar la verificación de los dos tipos de objetos que se firman en el sistema: documentos y cadenas de caracteres.

```
public class VerificarFirma {


public boolean verificarStr (String infoAfirmar, String clicert) throws Exception
  {
     String firma = null;
     boolean verifies=false;

     //IAIK es agregado como proveedor
     IAIK.addAsProvider(true);

      try  {
        byte[] realSig=null;

        // tomo la firma del archivo porfavor.enc
        FileInputStream fis3 = new FileInputStream("//tmp//files//porfavor.enc");
        BufferedInputStream bufin3 = new BufferedInputStream(fis3);
        byte[] buffer3 = new byte[128];
```

```
        int len3 = bufin3.read(buffer3);
        String str_buffer3=new String(buffer3);

        realSig = str_buffer3.getBytes();

        X509Certificate cert = null;
        try {
             cert = new X509Certificate(clicert.getBytes());
        }
        catch (CertificateException vvv) { }

        // obtener la clave publica
        PublicKey pub = cert.getPublicKey();

        Signature sig = null;
        try {
          sig = Signature.getInstance("MD5/RSA","IAIK");
        }
        catch (NoSuchAlgorithmException eee) { }
        catch (NoSuchProviderException iii) { }

        //cargo la clave pública para poder verificar
        try {
           sig.initVerify(pub);
        }
        catch (InvalidKeyException yyy) { }

        // Se verifica la firma
        try  {
           //cargo la información que quiero verificar
           byte[] buffer = infoAfirmar.getBytes();
           int len = buffer.length;
           sig.update(buffer, 0, len);

           //se verifican los datos en base a la firma que se realizó en el cliente (buffer3)
           verifies = sig.verify(buffer3);
        }
        catch (SignatureException ppp) { }

        return verifies;
}
```

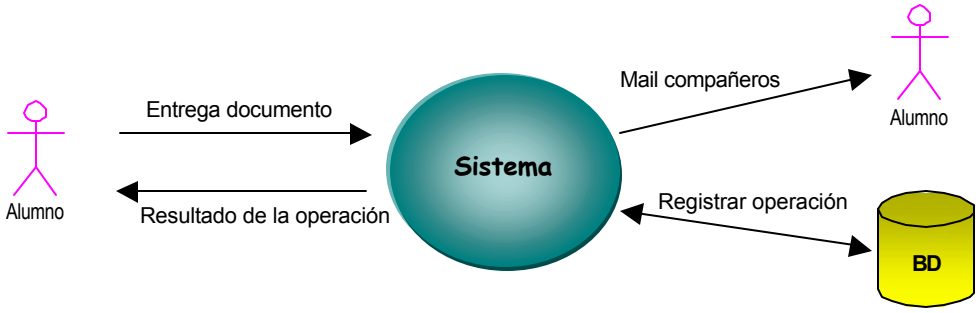
public boolean verificarDoc (String pathDoc, String clicert, String firmaDoc) throws Exception

```
  {

    String firma = null;
    boolean verifies=false;

    //IAIK es agregado como proveedor
    IAIK.addAsProvider(true);
```

```
    X509Certificate cert = null;

    try {
        cert = new X509Certificate(clicert.getBytes());
     }
    catch (CertificateException vvv) { }

    // obtener la clave publica
    PublicKey pub = cert.getPublicKey();

    Signature sig = null;

    try {
           sig = Signature.getInstance("MD5/RSA","IAIK");
    }
    catch (NoSuchAlgorithmException eee) { }
    catch (NoSuchProviderException iii) { }

    try {
        sig.initVerify(pub);
    }
    catch (InvalidKeyException yyy) { }


 // ***************** verifico la firma ********************

     FileInputStream fis = new FileInputStream (pathDoc);
     BufferedInputStream bufin = new BufferedInputStream(fis);
     byte[] buffer = new byte[1024];
     int len = 0;
     try {
          while (bufin.available() != 0) {
                   en = bufin.read(buffer);
                  sig.update(buffer, 0, len);
          };
          bufin.close();
           verifies = sig.verify(firmaDoc.getBytes());
     }
     catch (SignatureException ppp) { }

     return verifies;
   }
}
```

Construcción de mensajes de correo electrónico firmados

El sistema cuenta con un circuito de mensajes electrónicos subyacente cuya función es informar a cada participante del sistema cada vez que ocurre un evento ligado a alguna de las tesis en las cuales dicho participante se encuentra involucrado (Por ejemplo cuando un alumno realiza una entrega el sistema envía automáticamente un correo electrónico a sus compañeros advirtiéndolos de dicha entrega).



A fin de que el sistema pueda proveer esta funcionalidad, se implementaron dos clases:

-*ManejoMail:* El propósito de esta clase es obtener la información necesaria para construir el mensaje que va a enviarse. La clase cuenta con cinco métodos que son invocados según la operación que se realiza en el sistema. Dichos métodos son:

- ✓ *armaMailParaAlumnos* (éste es invocado cuando un alumno efectúa una entrega, confirma o rechaza una entrega realizada por un compañero)
- ✓ *armaMailParaDirector* (el mismo es invocado cuando todos los integrantes de un grupo han confirmado una entrega. Este método tiene como objeto informar al director que tiene un nuevo informe para revisar)
- ✓ *armarMailParaJurado* (este método es llamado una vez que el director de un grupo valida una entrega y la misma debe ser evaluada por los miembros del jurado. Este método informa a cada uno de los miembros del jurado asignado a dicho grupo).
- ✓ *ArmaMailParaGrupo* (éste es invocado cuando el director registra alguna operación relacionada con un grupo (aprobar, rechazar una entrega). El mismo envía un mensaje de correo electrónico a cada uno de los integrantes del grupo informando cuál fue la operación realizada.

-*EnvioMailFirmado*: Esta clase construye un mensaje de correo electrónico a partir de los valores de los campos que recibe como parámetros de la clase ManejoMail. Para llevar a cabo esta tarea se usan las clases: Session, Message, Transport, Header y MessagingException pertenecientes al paquete javax.mail y las clases MimeMessage, MimeBodyPart, InternetHeaders e InternetAddress del paquete javax.mail.internet.

También debimos incorporar otras librerías, como ser:

- ✓ ***IAIK-JCE***: para poder construir un objeto Certificado a partir de un archivo que contiene el certificado en formato P12 y así luego poder obtener información del certificado y extraer su clave privada, entre otras tareas. La clase EnvioMailFirmado usa las siguientes clases del paquete IAIK-JCE: iaik.security.provider.IAIK,iaik.x509.X509Certificate, iaik.pkcs.pkcs12.PKCS12, iaik.pkcs.pkcs12.KeyBag, iaik.pkcs.pkcs12.CertificateBag, iaik.pkcs.PKCSException, iaik.pkcs.PKCSParsingException

- ***IAIK-SMIME***: con el fin de construir mensajes firmados. La clase EnvíoMailFirmado utiliza las clases SmimeMultipart y SignedContent.

También fue necesario agregar y asociar los tipos MIME necesarios para la construcción de mensajes firmados. Para agregar los tipos MIME utilizamos la clase *MimetypesFileTypeMap* perteneciente al paquete javax.activation. La misma se usó de la siguiente forma:

```
MimetypesFileTypeMap mimetypesFileTypeMap = new MimetypesFileTypeMap();
mimetypesFileTypeMap.addMimeTypes("multipart/signed");
FileTypeMap.setDefaultFileTypeMap(mimetypesFileTypeMap);
```

Con el fin de asociar los distintos tipos de contenidos de mensajes a los manejadores correspondientes, usamos la clase *MailcapCommandMap*, perteneciente también al paquete javax.activation. Esta clase implementa la clase abstracta CommandMap cuya configuración se basa en los archivos mailcap especificados en la RFC 1524 (para más información dirigirse a http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1524.html).
La misma se usó de la siguiente forma:

```
MailcapCommandMap mailcapCommandMap =
(MailcapCommandMap)CommandMap.getDefaultCommandMap();
mailcapCommandMap.addMailcap("multipart/signed;;x-java-content-handler =
iaik.security.smime.signed_content");
CommandMap.setDefaultCommandMap(mailcapCommandMap);
```

Los pasos para construir un mensaje electrónico firmado son los siguientes:

1. Obtenemos información de claves y certificados a partir del certificado representado en un archivo con formato .P12 que vamos a necesitar luego para firmar el mensaje de correo (obtener la clave privada, desencriptar dicha clave privada).

2. Agregamos y asociamos los tipos MIME que necesitamos

3. Se construye el mensaje a enviar a partir de los datos recibidos:
   a. Se crea un mensaje MIME (representado por un objeto MimeMessage)con su correspondiente asunto, su correspondiente emisor y sus correspondientes destinatarios
   b. Creamos y completamos la primer parte del mensaje (representado por un objeto MimeBodyPart), para incluir el cuerpo del mensaje en texto plano.
   c. Creamos el objeto Multipart (representado por un objeto SMimeMultipart) y le agregamos la primer parte del mensaje, representada por el objeto MimeBodyPart.
   d. Creamos la firma del mensaje (representada por un objeto SignedContent). Para ello se crea el objeto en cuestión, se le asocia el manejador apropiado para mensajes S/Mime, luego se incluye en el mismo el certificado del firmante y la clave privada correspondiente.
   e. Posteriormente, la firma fue incluida en el mensaje.
   f. Finalmente, el mensaje firmado es enviado.

Este es el código de la clase, en el cual podemos observar la forma en la cual se llevaron a cabo los pasos anteriormente descriptos:

```
public class EnvioMailFirmado{

   //Este servlet siempre firma en nombre de la aplicacion
  //El objetivo es informar el resultado de una operación realiza a los usuarios que
intervinien en la misma.

  String signatureAlgName = "SHA-1/RSA";

  String privateKeyFile = "/usr/local/ssl/bin/administrativoP12.p12";
  String certsFile = null;
  String caCertsFile = "/usr/local/apache/certs/apunca.crt";
  String privKeyPassword = "nqry";

  public void mandoMail(String servidor, String from, String sendTo, String subject,
String bodyText) {

   //Espera recibir los siguientes parámetros para construir el mail:
   //from = la dirección de correo del emisor del mail
   //to = la dirección de correo del receptor del mail
   //subject = el tema del mail
   //bodyText para incluir el contenido del mensaje, el texto...


 //Obtenemos información de claves y certificados que vamos a necesitar     //luego
para firmar el mensaje de correo

      FileInputStream entrada= null;

      RSAPrivateKey rsaPriv= null;
      FileInputStream fis=null;
      try {
          fis = new FileInputStream(privateKeyFile);
          }
      catch (FileNotFoundException iii){ }

      PKCS12 archi = null;

      try {
         archi = new PKCS12(fis);
          }
      catch (IOException x) {}
      catch (PKCSParsingException xxx) {}

      // Desencripto la clave privada
      String realPasswd = "nqry";
      char[] passwd = realPasswd.toCharArray();

      try {
         archi.decrypt(passwd);
          }
      catch (PKCSException ppp) { }
```

```
        // Obtengo la clave privada
        KeyBag clave = archi.getKeyBag();

        PrivateKey priv = null;
        priv = clave.getPrivateKey();
        rsaPriv = (RSAPrivateKey)priv;

        //Creamos el arreglo de certificados que será incluido en el
        //mensaje firmado
        X509Certificate[] myCerts =
CertificateBag.getCertificates(archi.getCertificateBags());

        //Agregamos y asociamos los tipos MIME que necesitamos
        MimetypesFileTypeMap mimetypesFileTypeMap = new MimetypesFileTypeMap();
        mimetypesFileTypeMap.addMimeTypes("multipart/signed");
        mimetypesFileTypeMap.addMimeTypes("multipart/mixed");
        mimetypesFileTypeMap.addMimeTypes("application/x-pkcs7-mime");
        mimetypesFileTypeMap.addMimeTypes("application/x-pkcs7-signature");
        FileTypeMap.setDefaultFileTypeMap(mimetypesFileTypeMap);

       MailcapCommandMap mailcapCommandMap =
(MailcapCommandMap)CommandMap.getDefaultCommandMap();
        mailcapCommandMap.addMailcap("multipart/signed; ; x-java-content-
handler=iaik.security.smime.signed_content");
        mailcapCommandMap.addMailcap("multipart/mixed; ; x-java-content-
handler=com.sun.mail.handlers.multipart_mixed");
        mailcapCommandMap.addMailcap("application/x-pkcs7-signature; ; x-java-
content-handler=iaik.security.smime.signed_content");
        mailcapCommandMap.addMailcap("application/x-pkcs7-mime; ; x-java-content-
handler=iaik.security.smime.encrypted_content");
        CommandMap.setDefaultCommandMap(mailcapCommandMap);

//construimos el mensaje, que luego firmaremos

      // variables del mensaje
        Properties props;

        //Usamos ADA como SMTP Server
        String    mailHost  = "ada.info.unlp.edu.ar";
        Session    session;

        // creamos algunas propiedades y toma una instancia de sesión
        props = System.getProperties();
        props.put("mail.smtp.host", mailHost);

        session = Session.getDefaultInstance(props, null);

        // creamos un mensaje mime

        MimeMessage mail = null;
        try {
          mail = new MimeMessage(session);
```

```
        mail.setFrom(new InternetAddress(from));
        InternetAddress[] address = {new InternetAddress(sendTo)};
        mail.setRecipients(Message.RecipientType.TO, address);
        mail.setSubject(subject);
        }
      catch ( Exception ex2) { }

// creamos y llenamos la primer parte del mensaje (bodyText), para el mensaje
      //en texto plano
        MimeBodyPart bodyPart1 = new MimeBodyPart();
       try {
           bodyPart1.setText(bodyText);
           }
       catch ( MessagingException ex2) {}

         // creamos el objeto Multipart y le agrega la primer parte del mensaje

        SMimeMultipart multiPart = new SMimeMultipart();
       try {
           multiPart.addBodyPart(bodyPart1);
           }
       catch ( MessagingException ex2) {}

      //Creamos el SignedContent
      SignedContent sc = new SignedContent(true);
      sc.setDataHandler(new
                    DataHandler(multiPart,multiPart.getContentType()));
      String tipocont=multiPart.getContentType();
      try {
         sc.setCertificates(myCerts);
         //Firmamos el mensaje
         sc.setSigner(rsaPriv,myCerts[0]);
          }
      catch (InvalidKeyException aei) { }
      catch (SignatureException aeio) { }

      String tiposmime=sc.getSMimeType();

      //Seteamos el contenido del mensaje
      try{
        mail.setContent(sc,sc.getContentType());
         }
      catch (MessagingException mex) { }

      //Enviamos el mensaje
      try{
          Transport.send(mail);
         }
      catch (Exception mex) { }
      catch (java.lang.NoSuchFieldError mex) { }

  }}
```

Experiencia adquirida

Durante la etapa en la que realizamos este trabajo nos enfrentamos ante determinadas dificultades, las cuales fueron superadas más allá de las demoras que significaron. Ellas nos llevaron a tomar ciertas decisiones que fueron marcando nuestro camino. También nos ayudaron a comprender las dificultades prácticas inherentes a la puesta en marcha de una aplicación que trabaja junto con una infraestructura PKI.

Es importante destacar los siguientes puntos:

- ✓ Al intentar buscar información respecto a otros proyectos de estas características descubrimos que se trataba de un tema que recién comenzaba a explorarse, fundamentalmente en los puntos relativos a la implementación.
- ✓ Dado que no se contó con una versión final de APuN-CA, tuvimos que enfrentarnos con algunos problemas propios de este producto, como ser conflictos relacionados a la autenticación entre los módulos cuando éstos utilizaban certificados creados para nuestra arquitectura
- ✓ Existen problemas relacionados con la incompatibilidad entre las representaciones de determinados objetos (como ser datos firmados, datos encriptados, etc) presentes en las distintas librerías usadas.
- ✓ Se descubrió la falta de compatibilidad entre las distintas versiones de las librerías utilizadas para el manejo de funciones criptográficas y para la construcción de mensajes firmados.

Conclusiones

De acuerdo a la experiencia adquirida a partir de nuestro trabajo, resulta interesante destacar los puntos más relevantes relacionados con la puesta en marcha de un sistema que utiliza la PKI como solución de seguridad.

Actualmente se están realizando distribuciones de PKI en todo el mundo, preparándose para el nuevo milenio de negocios digitales. Las numerosas organizaciones que operan con sistemas PKI reflejan el hecho de que se trata de un marco fundamental para permitir aplicaciones de comercio electrónico que repercutirá en todos nosotros. La PKI pronto será tan habitual que las organizaciones emitirán certificados digitales y tarjetas inteligentes como práctica usual.

A través del uso de esta tecnología, las organizaciones deben volver a diseñar sus prácticas de trabajo, e implantar sistemas para llevar a cabo sus actividades electrónicas con total seguridad. Es posible que, así como comienza a suceder en los clientes de correo electrónico, a medida que la firma digital se incorpore en las distintas aplicaciones existentes, el usuario final adquiera el hábito de utilizarla a través de los mecanismos simples que los desarrolladores creemos para ello. En realidad, las posibilidades tecnológicas no son más que una oportunidad. Su aplicación y absorción por parte de los usuarios es lo que las harán ser válidas y útiles.

El presente informe, fruto de la investigación realizada, constituye una pieza fundamental para *comprender* todos los conceptos relacionados con la infraestructura PKI.

Nuestro principal aporte consiste en haber desarrollado una *aplicación específica*, diferente de las clásicas en las cuales se utiliza firma digital como son el correo electrónico y la WEB (si bien la firma digital es muy importante en estos casos, existen un sinnúmero de aplicaciones que se pueden beneficiar al introducir un *esquema de seguridad que tenga como eje la firma digital* con el fin de garantizar, por ejemplo, autenticidad y no repudio).

Respecto a los trabajos futuros que se desprenden de este desarrollo podemos mencionar que resulta factible que la Cátedra de Trabajo de Grado ponga en funcionamiento este sistema a fin de realizar el manejo administrativo de los trabajos de grado.

Nuestro trabajo abre la puerta a otras aplicaciones que usen la misma arquitectura de PKI. Dichas aplicaciones pueden también beneficiarse del uso de las bibliotecas que en él intervienen para la implementación de determinados aspectos en cuanto a la seguridad respecta.

Glosario

Estos son términos usados frecuentemente durante el desarrollo del presente informe.

+ 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 (en especial certificados X.509) y avala la relación entre los datos contenidos. Debe gozar de la confianza de los usuarios, usualmente por depender de un gobierno, corporación u otra organización. Es responsable de administrar el ciclo de vida de los certificados y, posiblemente, el de las claves asociadas.

+ Autoridad de registración - registration authority (RA)

Entidad opcional en una PKI, separada de las CAs, que no firma certificados digitales pero es responsable de registrar o verificar alguna o toda la información necesaria (en particular, la identidad) para que una CA emita certificados y efectúe otras funciones administrativas.

+ ASN.1 (Abstract Syntax Notation One)

Lenguaje formal para describir en forma abstracta los mensajes a ser intercambiados entre Sistemas distribuidos.

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

Secuencia ordenada de certificados que 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; para ello, se debe contar con la clave pública del objeto inicial de la secuencia.

+ Cifrado - encrypt

Transformación criptográfica de datos (texto llano) en una forma que oculta el significado original (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. Usualmente, el texto llano que ingresa a una operación de cifrado es texto en claro, pero en algunos casos puede ser texto cifrado que proviene de la salida de otra operación de cifrado.

+ 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.

+ Clave privada - private key

Componente secreto del par de claves criptográficas usadas en criptografía asimétrica, conocido solamente por el usuario titular.

+ Clave pública - public key

Componente publicable del par de claves criptográficas usadas en criptografía asimétrica.

+ 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 inteligible.

+ Criptografía asimétrica - asymmetric cryptography

Moderna rama de la criptografía, también conocida 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; ésto incrementa el costo y los riesgos de la distribución segura de la clave, asunto en el que la criptografía asimétrica tiene ventaja.

+ Criptosistema - cryptographic system

Conjunto de algoritmos criptográficos junto con los procesos de administración de claves, que soporta el uso de los algoritmos en algún contexto de aplicación.

+ Descifrado - decrypt

Proceso de restablecer criptográficamente el texto cifrado a la forma de texto llano 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.

+ Firma digital - digital signature

Valor calculado con un algoritmo criptográfico y añadido a un objeto de datos de forma tal que cualquier receptor de los datos pueda usar la firma para verificar el origen de los 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, usualmente de tamaño variable y posiblemente grande) y así mapea el objeto a otro más pequeño, el resultado, normalmente de tamaño fijo.

+ 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) para una comunidad de usuarios en aplicaciones de criptografía asimétrica.

+ Integridad - data integrity - integridad de los datos

La propiedad de que los datos no hayan sido cambiados, destruídos o perdidos en forma no autorizada o accidental; no se refiere al significado de los datos ni a la verosimilitud de la fuente.

+ Lista de certificados revocados - certificate revocation list (CRL)

Estructura de datos que enumera certificados digitales que su emisor ha invalidado con anticipación a la fecha de vencimiento.

+ Modelo de seguridad - security model

Descripción esquemática de un conjunto de entidades y relaciones por las cuales se proporciona dentro de un sistema un conjunto especificado de servicios de seguridad.

+ 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, una organización), construída según las convenciones culturales asociadas; ejemplos: "Dr. Felipe J. Vázquez", "Naciones Unidas", "impresora laser del 3er. piso".

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

Serie de especificaciones publicadas por RSA Laboratories 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

Contracción de "Infraestructura de Clave Pública X.509", 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.

+ 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, no un tipo de servicio de seguridad.

+ Repudio - repudiation

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

+ 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 - plaintext

Datos de entrada que serán transformados por un proceso de cifrado, o datos de salida de un proceso descifrador. Usualmente, la entrada al cifrador también es texto en claro pero, en algunos casos, puede ser texto cifrado proveniente de la salida de otra operación de cifrado.

Referencias

Capítulo 2 - Criptografía

[1] http://www.itl.nist.gov/fipspubs/fip46-2.htm
[2] http://www.cs.nps.navy.mil/curricula/tracks/security/notes/chap04_43.html
[3] http://www.cis.ohio-state.edu/htbin/rfc/rfc2268.html
[4] http://csrc.nist.gov/publications/nistpubs/800-20/800-20.pdf
[5] http://www.ansi.org

Capítulo 3 - Estándares PKC

[1] http://as400bks.rochester.ibm.com/pubs/html/as400/v4r4/ic2924/info/RZAHURZAHU0NMDISTNAMECO.HTM

Capítulo 4 - Infraestructura de Clave Pública (PKI)

❖ Certificados

[1] http://www.w3.org/Protocols/HTTP-NG/asn1.html.

❖ Autoridades de Certificación

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

❖ Repositorios de Datos

[1] http://www.nexor.com/x500frame.htm.
[2] http://www.umich.edu/~dirsvcs/ldap/doc/rfc/rfc1777.txt.

❖ Autoridad de Registración

[1] http://as400bks.rochester.ibm.com/pubs/html/as400/v4r4/ic2924/info/RZAHURZAHU0NMDISTNAMECO.HTM

❖ Listas de revocación

[1] http://www.ietf.org/internet-drafts/draft-ietf-pkix-ocspv2-02.txt.

❖ PKIX

[1] http://www.ietf.org/html.charters/pkix-charter.html
[2] http://www.ietf.org/internet-drafts/draft-ietf-pkix-ldap-v3-03.txt.

Capítulo 6 - Firma Digital

[1] http://www.onnet.es/06041008.htm.
[2] http://www.ispo.cec.be/eif/policy/com98297.html.
[3] http://www.darmstadt.gmd.de/ice-tel/ice-home.html.

Capítulo 7 - Implementación de seguridad en Servicios de Internet

[1] http://www.imc.org/rfc1847.
[2] http://www.imc.org/rfc1521.
[3] http://speth08.wu-wien.ac.at/manuals/rfc-pem.html.
[4] http://afs.wu-wien.ac.at/manuals/rfc/rfc1991.txt.
[5] http://www.faqs.org/rfcs/rfc1848.html
[6] http://www.faqs.org/rfcs/rfc821.html

Capítulo 8 - Implementación

[1] http://java.sun.com
[2] http://java.sun.com/products/jdbc/
[3] http://www.microsoft.com/data/odbc
[4] http://www.netcraft.com/survey/
[5] http://www.modssl.org.
[6] http://www.openssl.org.
[7] http://java.apache.org/old/doc/ajpv2-spec.html
[8] http://www.apache-ssl.org/docs.

Bibliografía

Capítulo 1 - Introducción

Libro: “Electronic Commerce - On-line Odering Digital Money” Segunda edición
Pete Loshin/ Paul Murphy
Charles River Media, INC.
http://www.modernizacion.com.cl/utic/firma/index.htm
http://www.w3.org/Security/Faq/www-security-faq.html
http://www.sans.org/newlook/publications/roadmap.htm
http://netsecurity.about.com/compute/netsecurity/cs/generalfaqs/index.htm

Capítulo 2 - Criptografía

http://www.rsa.com/
http://world.std.com/~franl/crypto.html
http://www.jjtc.com/Security/crypto.htm
http://www.nwfusion.com/netresources/crypto.html

Capítulo 3 - Estándares PKC

http://www.rsalabs.com/pkcs/

Capítulo 4 - Infraestructura de Clave Pública (PKI)

Libro: “Digital Certificates – Applied Internet Security”
Jalal Feghhi - Jalil Feghhi - Peter Williams”
Addison Wesley
Libro: “Secure Electronic Commerce – Building the infraestructore for Digital Signatures and Encryption”
Warwick Ford – Michael S. Baum
Prentice Hall PTR
http://www.entrust.com/downloads/docs/protocols_pki.htm
http://www.entrust.com/downloads/docs/protocols_pki.htm
http://www.elock.com/WHITE-P/architecture.htm
http://www.counterpane.com/pki-risks.html
http://www.nwfusion.com/netresources/crypto.html
http://www.entrust.com/resourcecenter/pdf/pki.pdf

❖ Certificados

Libro: “Digital Certificates – Applied Internet Security”
Jalal Feghhi - Jalil Feghhi - Peter Williams”
Addison Wesley
Libro: “Secure Electronic Commerce – Building the infraestructore for Digital Signatures and Encryption”
Warwick Ford – Michael S. Baum
http:://www.zdnet.com/pcweek/reviews/0428/28cert.html

http://psych.psy.uq.oz.au/~ftp/Crypto/certs.htm
http://www.quasars.com/services/certificates.htm
http://home.netscape.com/security/techbriefs/certificates/

❖ Autoridades de Certificación

Libro: "Digital Certificates – Applied Internet Security"
Jalal Feghhi - Jalil Feghhi - Peter Williams"
Addison Wesley
Libro: "Certification Authority Guidelines 1.0 de ECOM (Electronic Commerce Promotion Council of Japan) – Certification Authotity Working Group"
http://www.baltimore.ie/clintonvisit98/techback.html
http://www.bankgate.com/agree.htm
http://www.verisign.com/repository/CPS1.1/CPSCH13.HTM
http://www.anl.gov/ECT/certify/CA-FAQ.html
http://www.ec.gov.sg/ETBpart8.html
http://fw4.iti.salford.CA.uk/ice-tel/trust/ieee/

❖ Repositorios de Datos

http://www.ust.hk/itsc/pki/model/cr.html
http://www.faqs.org/rfcs/rfc2587.html

❖ Autoridad de Registración

Libro: "Digital Certificates – Applied Internet Security"
Jaal Feghhi - Jalil Feghhi - Peter Williams"
Addison Wesley
http://www.imc.org/draft-ietf-pkix-roadmap

❖ Listas de revocación

Libro: "Digital Certificates – Applied Internet Security"
Jalal Feghhi - Jalil Feghhi - Peter Williams"
Addison Wesley

❖ PKIX

Libro: "Digital Certificates – Applied Internet Security"
Jalal Feghhi - Jalil Feghhi - Peter Williams"
Addison Wesley
http://www.imc.org/draft-ietf-pkix-roadmap

Capítulo 5 - APuN-CA - Ejemplo de una Infraestructura PKI

Documentación del producto ApuN-CA.
http://www.pki.gov.ar

Capítulo 6 - Firma Digital

http://www.baltimore.ie/clintonvisit98/techback.html
http://www.ilpf.org/work/ca/app6.htm
http://www.abanet.org/scitech/ec/isc/dgs-tutorial.html

http://www.baltimore.ie/clintonvisit98/techback.html
http://www.imc.org/draft-ietf-pkix-roadmap
http://www.abanet.org/scitech/ec/isc/dgs-tutorial.html
http://www.pkilaw.com/surety_time_long_3.htm
http://www.lafirmadigital.com
http://www.pki.gov.ar
http://www.ncsc.dni.us/NCSC/TIS/ELECFILE/Digsig1.htm

Capítulo 7 - Implementación de seguridad en Servicios de Internet

Libro: "Digital Certificates – Applied Internet Security"
Jalal Feghhi - Jalil Feghhi - Peter Williams"
Addison Wesley
Libro: "Secure Electronic Commerce – Building the infraestructore for Digital Signatures and Encryption"
Warwick Ford – Michael S. Baum
Prentice Hall PTR
http://www-mat.upc.es/~jforne/jarne.pdf
http://www.iec.csic.es/criptonomicon/ssl.html
http://www.se-com.com/wp/jgyandyke-smime.html
http://www.iec.csic.es/criptonomicon/shttp.html

Capítulo 8 - Implementación

Libro: "The Java tutorial - Second Edition – Object-Orientented Programming for the Internet"
Mary Campione-Kathy Walrath
Addison-Wesley
Libro: "The Java Tutorial Continued – The rest of the JSDK"
Campione- Walrath-Huml
Addison-Wesley
http://www.openresources.com/es/magazine/tutoriales/apache/htm/x31.htm
http://www.teledatos.com/page23.html
http://www.linti.unlp.edu.ar
http://java.sun.com
http://www.webdeveloper.com/
http://www.programacion.com/java/jdbc1.html
http://java.apache.org/jserv/install/howto.unix_install.html
http://www.apache.org
http://jcewww.iaik.at/