UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CREACIÓN DE UN MODELO DE IDENTIFICACIÓN DE ALUMNOS E INTEGRACIÓN DE SERVICIOS UTILIZANDO SMARTCARDS Y TECNOLOGÍA JAVA.

CASO DE ESTUDIO: FACULTAD DE INGENIERÍA

EL PROYECTO IDI

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A N :
CARLOS ALEGRÍA GALICIA
SAMUEL FLORES SANDOVAL
AUGUSTO DOBESLAO HERNÁNDEZ LÓPEZ

DIRECTOR DE TESIS:
ING. ALBERTO GONZÁLEZ GUÍZAR

MÉXICO, D.F. ABRIL, 2005

Creación de un modelo de identificación de alumnos e integración de servicios utilizando smartcards y tecnología java. Caso de estudio: Facultad de Ingeniería.

El proyecto IDI

Carlos Alegría Galicía <calegria@gmail.com>
Samuel Flores Sandoval <samuelflores@gmail.com>
Augusto Dobeslao Hernández López <dobeslao@gmail.com>

Creación de un modelo de identificación de alumnos e integración de servicios utilizando smartcards y tecnología java. Caso de estudio: Facultad de Ingeniería. : El proyecto IDI

por Carlos Alegría Galicía, Samuel Flores Sandoval, y Augusto Dobeslao Hernández López

publicado en Febrero de 2005

Índice General

ción de alumnos e integración de ser-

Lista de figuras

Lista de tablas

Lista de ejemplos

Introducción

Para el buen funcionamiento de cualquier organización es indispensable un sistema de identificación de personas que sea efectivo y seguro. La importancia de contar con un sistema de identificación radica principalmente en dos aspectos: el primero es que se trata de un requisito para tener la capacidad de decidir quién tiene acceso a la información y a otros bienes con los que cuenta la organización. El segundo es que los sistemas de identificación facilitan la operación de los procesos y permiten ahorrar ciertos recursos mediante la integración de diversos servicios.

Actualmente la Facultad de Ingeniería de la UNAM carece de un sistema de identificación de personas que satisfaga la características mencionadas anteriormente. Los procedimientos actuales de identificación son funcionales, pero no permiten la integración de servicios ni garantizan la seguridad de los bienes que pretenden proteger.

En los siguientes capítulos mostraremos el desarrollo y los resultados de una investigación realizada con el objetivo de proponer la creación de un sistema adecuado para la identificación de personas de la Facultad de Ingeniería de la UNAM.

Debido a que el problema de la identificación de todas las personas que forman parte de la Facultad de Ingeniería es muy complejo, limitamos nuestra investigación a la identificación de alumnos, puesto que se trata de un caso que conocemos bien.

En la búsqueda de medios para lograr nuestro objetivo, la utilización de tarjetas inteligentes nos pareció la solución más adecuada por las razones que se expondrán con detalle en los siguientes capítulos.

Así pues, nuestro objetivo concreto fue evaluar la posibilidad de crear un sistema de identificación de alumnos en nuestra facultad, basado en tarjetas inteligentes, que solucione algunos problemas existentes en los procedimientos actuales.

A grandes rasgos, la estructura de este reporte es la siguiente: En el capítulo uno, expondremos los procedimientos actuales de identificación de alumnos en la Facultad de Ingeniería, sus problemas y el Estado actual del desarrollo de sistemas de identificación personal. En el segundo capítulo presentaremos un análisis de las necesidades de identificación de alumnos en la Facultad y una justificación del uso de

tarjetas inteligentes para resolver el problema. El tercer capítulo es una explicación de los conceptos básicos que son necesarios para el entendimiento de la tecnología utilizada en la solución que proponemos. En el capítulo cuatro mostramos a detalle el modelo propuesto, mientras que en los capítulos cinco y seis evaluamos su factibilidad con la creación de dos prototipos a pequeña escala. Finalmente, presentamos nuestras conclusiones.

Capítulo 1.

Antecedentes

1.1. Procedimientos de identificación actuales en la Facultad de Ingeniería

Actualmente la Facultad de Ingeniería carece de un sistema de identificación rápido, eficiente y seguro para todos sus alumnos. Existen diversos medios de identificación como la credencial de alumno y contraseñas de acceso para cada uno de los laboratorios de cómputo; sin embargo, todas ellas funcionan sólo en un ámbito reducido y manejan la información del alumno a diferentes niveles y sobre diversos sistemas informáticos. Un caso similar se da con los servicios proporcionados por el resto de las entidades o dependencias de la Universidad, como el préstamo de libros en la biblioteca central y los servicios de actividades deportivas, para los cuales se requiere una identificación diferente.

Para utilizar la mayoría de los servicios que ofrece la Facultad de Ingeniería (y en general, en toda la Universidad) se deben realizar trámites repetitivos que podrían ser automatizados.

El servicio de préstamo de libros las bibliotecas de la Facultad de Ingeniería requiere que el alumno cuente con un medio de identificación, además de estar registrado en su sistema, por lo que debe ser dado de alta antes de poder realizar cualquier trámite.

Para poder acceder a los laboratorios de cómputo, se requiere que el alumno sea estudiante de la facultad y que cuente con una identificación oficial (p.e. credencial de la facultad o credencial especial para laboratorio). En algunos casos, para po-

der hacer uso de las computadoras, al inicio del semestre el alumno tiene que demostrar que se encuentra inscrito en el semestre actual y darse de alta en el sistema.

Si se requiere demostrar que el alumno efectivamente está inscrito en el semestre actual y que es estudiante de la Facultad, debe presentar la credencial de alumno y además proporcionar la tira de materias actualizada.

En todos estos trámites se maneja prácticamente la misma información sobre el alumno (p.e el nombre, número de cuenta, dirección, teléfono, correo electrónico, etc.) y se llega a tener tanto redundancia como inconsistencia entre los registros de cada una de las dependencias provocadas tanto por las personas encargadas de realizar las altas en los sistemas, como por los mismos alumnos al proporcionar sus datos.

Para darnos cuenta de que existe una gran cantidad de servicios de este tipo que se proporcionan a la comunidad estudiantil a través de las diversas direcciones generales y dependencias universitarias, basta con citar algunos de los más comunes, que continuamente son utilizados por alumnos: los servicios de control escolar, servicios bibliotecarios, deportivos, culturales y recreativos. Para obtener estos servicios generalmente el alumno tiene que darse de alta en cada uno de los sitemas que ellos manejan, lo cual provoca que se dé de alta la misma información por cada alumno en cada uno de los servicios, con los inconvenientes que ya se indicaron.

Una mención especial merece el hecho de que cada una de las dependencias que otorgan algún tipo de servicio, expide su propia identificación *del servicio* (biblioteca, facultad, alberca, centro de cómputo, mediateca, etc.)

1.2. Estado Actual del desarrollo de sistemas de identificación personal

Los sistemas de identificación personal se utilizan para el manejo de información relativa a personas y objetos. Para tal propósito son utilizados mecanismos de registro magnético, óptico, acustico e impreso.

En estos sistemas se requiere de dos componentes fundamentales: un elemento codificado que contiene la información al respecto (p.e datos procesados o un patrón establecido) y un elemento con la capacidad de recuperar esa información.

Cotidianamente, los sistemas de identificación personal pueden ser utilizados de forma diversa, como por ejemplo en el acceso a una cuenta bancaria, permitir la entrada a un área restringida, ingreso a los recursos lógicos de una computadora, utilización de algún equipo electrónico (como en el caso de algunas fotocopiadoras), acceso a las oficinas de alguna empresa, apertura de puertas y utilización de tarjetas de crédito, entre otras [MEDINA]. En la actualidad encontramos cada vez más sistemas automátizados que permiten que los procesos se agilicen, se cometan cada vez menos errores y en consecuencia incrementen tanto la confiabilidad como la eficiencia de los procesos que realizan.

Estos sistemas no son solamente aplicados a control de personal, sino también al control de objetos que necesitan ser reconocidos o clasificados, sobre todo cuando se destinan a usos comerciales. Cuando el número de artículos rebasa la capacidad de clasificación humana es cuando se necesita una identificación automatizada del producto. Además, la clasificación debe ser lo suficientemente exacta como para poder reconocer características extras a las propias del producto, como su lugar de origen, ubicación, destino, costo, precio de venta, verificación, control, contabilidad, estadísticas e inventarios.

Un buen ejemplo de este tipo de sistemas lo podemos encontrar en empresas de paquetería y envío de mensajes, que entre sus servicios ofrecen el rastreo de paquetes, permitiendo al cliente conocer la ubicación de su paquete y estimar el tiempo de recepción del mismo. Esto se puede lograr gracias a la identificación individual de cada paquete cuando se realiza su traslado entre terminales aéreas.

1.2.1. Tarjetas de identidad

Las tarjetas de identidad se utilizan cotidianamente para diversos propósitos y trámites. Un uso típico de estas tarjetas es durante el proceso electoral, para identificar a cada uno de los electores cuando se registran en la casilla para realizar su voto.

Las tarjetas de identidad se pueden dividir en dos grandes grupos: aquellas que incluyen información digitalizada (en una banda magnética, código, chip) y las que solamente tienen información visual. Ambos tipos de tarjeta pueden incluir alguna fotografía del poseedor.

Cuando se incluye la información digitalizada, las tarjetas de identidad se pueden clasificadar en las que son sólo para lectura y en las que son de lectura y escritu-

ra. Las primeras contienen información de la persona que no puede ser alterada una vez que ha sido emitida. Las segundas contienen datos que además de ser leídos pueden ser actualizados con nueva información. Éstas últimas son conocidas como tarjetas inteligentes.

Debido a que las tarjetas de identidad son utilizadas para verificar los derechos de un persona durante eventos sensibles, la mayoría contienen rasgos orientados a minimizar la posibilidad de fraude. La inclusión de una fotografía, firma o huella digital sirve como mecanismo ideal de verificación visual de la identidad. Los dispositivos de seguridad impresos como los hologramas o diseños a color difíciles de reproducir pueden ser empleados para prevenir la alteración o falsificación de tarjetas. Los elementos de identificación integrados en las tarjetas también pueden servir para prevenir el fraude.

1.2.2. Número de identificación personal (NIPs)

Los NIPs se utilizan extensamente en muchos países y cada vez más servicios son ofrecidos a través de sistemas automatizados, como los cajeros automáticos de los sistemas bancarios o sistemas de transferencia electrónica de fondos.

Sirven como identificadores únicos cuando las personas utilizan servicios automatizados a fin de verificar que se tiene derecho de ingresar a determinado servicio, en nombre de esa persona. Para que este proceso sea efectivo, se debe tener el cuidado en la forma en que se generan, almacenan y distribuyen estos NIPs.

En el momento que un banco emite un NIP, normalmente se realiza por correo tradicional o solicitando que la persona acuda a una oficina bancaria para que se le asigne uno después de acreditar su identidad. Los NIPs utilizados en Internet pueden ser distribuidos a través de correo electrónico.

Las autoridades responsables deben supervisar los sistemas de distribución de los NIPs para que sean entregados a las personas correctas. Distribuirlos por correo tradicional o electrónico plantea algunos riesgos, ya que puede ser difícil o poco práctico exigir que los NIPs sean entregados de manera personal.

Una estrategia que puede evitar el mal uso de cualquier NIP es el utilizar una autentificación dual, es decir, exigir una segunda identificación, como una tarjeta de identidad o un número de identidad personal distinto, como el del seguro social.

Si un NIP llega a ser robado, puede ser mal utilizado si el sistema al que está ligado no exige alguna otra prueba de identidad. Para mayor seguridad, el uso del NIP se puede restringirse para que solo sean utilizados de manera conjunta con otro medio, como en el caso de una tarjeta inteligente. Requiriéndose ambos para tener acceso al sistema, utilizando el factor dual de *algo que se sabe* (NIP) y *algo que se tiene* (Tarjeta Inteligente), con lo cuál cualquier atacante tiene menos posibilidades de obtener los dos beneficios.

1.2.3. Sistemas de identificación biométrica

Estos sistemas utilizan dispositivos electrónicos biométricos, con la finalidad de medir alguna característica o cualidad de una persona para ser identificada y proporcionar acceso. Estos sistemas pueden ser divididos en dos grandes tipos : visuales y electrónicos.

Los métodos biométricos de identificación visual incluyen el uso de fotografías, firmas o huellas digitales en las tarjetas de identidad. Su costo de aplicación y administración es relativamente económico. Muchos de los sistemas de emisión de tarjetas de identidad incluyen una foto y firma. Este sistema se complementa cuando el responsable o custodio compara los elementos con la persona que los presenta.

Este procedimiento tiene sus debilidades. Las comparaciones de firmas y huellas digitales son actividades que requieren de grandes habilidades, por lo que no se puede esperar que el personal encargado las domine por completo. Incluso con las fotografías, la fisionomía de una persona puede en ocasiones cambiar significativamente respecto a la incluida en la tarjeta de identidad, sobre todo si no se actualiza de forma periódica. Sin embargo, esta clase de sistema de identidad probablemente sea suficiente en la mayoría de los casos donde el riesgo de fraude es relativamente bajo.

Si los sistemas biométricos de identificación visual no se consideran suficientemente seguros, se pueden usar los electrónicos. Estos pueden incluir voz, huellas de la mano o dactilares e incluso imágenes digitalizadas de la retina del usuario. En el caso de que se utilicen estos sistemas, los registros digitales de la voz o rasgos físicos de una persona son almacenados y comparados con los rasgos reales de la persona utilizando un lector adecuado. Solo se autoriza el acceso cuando se logra realizar una correspondencia plena.

Estos sistemas son la opción ideal y factible para ofrecer seguridad en sitios de alta sensibilidad.

1.2.4. Fotografía digitalizada

Las fotografías digitalizadas de los rostros de personas pueden ser comparadas con otras que se encuentren almacenadas. A través de programas de cómputo se puede determinar si una persona se ha encontrado en cierto lugar en un momento determinado.

1.2.5. Código de barras

El código de barras puede ser utilizado como un medio para identificar objetos que requieren ser clasificados. El sistema consiste en una serie de líneas codificadas y espacios que pueden ser leídas por un equipo decodificador láser y convertidas en una cadena de dígitos legibles por una máquina.

Es un método muy simple y eficiente para asignar una clave única de identificación a casi cualquier objeto. Son utilizados ampliamente en comercios, donde pueden identificar y marcar el precio de cada artículo, así como para registrar niveles de ventas y reservas. Pueden ser utilizados para efectos de inventario.

Los códigos de barras son establecidos por la Asociación Internacional de Numeración de Artículos (EAN) en común acuerdo con las Asociaciones Nacionales. La EAN proporciona un número de identificación conocido como FLAG de dos o tres dígitos, para el país de origen del producto.

Tabla 1.1. Códigos EAN según el país

País	EAN FLAG
Brasil	789
Canadá y EEUU	00-09
Francia	30-37
México	750

Posteriormente la Asociación Nacional proporciona un número de identificación

para el fabricante y también servirá para todos sus productos. Podrá asignar otros conjuntos numéricos para cada producto o forma de presentación del mismo, integrando de esta manera una serie única de números para cada uno de sus productos que se conocerá como *código* y que incluye el país, empresa, producto y control.

Este código se conforma con dos elementos, una serie de lineas verticales y un conjunto de números. Gracias a este código se pueden reconocer los productos, facilitando el manejo operativo y administrativo de ellos.

Cuando se aproxima el código de barras de un objeto determinado a un escáner, éste es capaz de detectar la presencia del objeto en su campo visual y activa un mecanismo electrónico de seguridad que dura mientras este permanezca en esa zona, evitando por lo general que el producto pueda ser registrado varias veces mientras es explorado por la unidad lectora.

Con la información capturada, se puede obtener el nombre, precio, tamaño u otras características registradas para ese objeto. Una ventaja que se obtiene con este sistema, es que la información que cambia constantemente (p.e. el precio del artículo) no se encuentra impresa en el producto, sino que se encuentra almacenada en el sistema de cómputo donde puede ser actualizado de acuerdo a las necesidades y frecuencia requerida.

Existe una variedad de este código que tan sólo emplea ocho carácteres (EAN8), su estructura es la que se presenta en la Tabla 1.2, "Descripción del Código EAN8".

Tabla 1.2. Descripción del Código EAN8

Posición	Descripción
1	Dígito de verificación del código.
2-6	Identificación del producto.
7-8	Identificación del país.

Código UPC (Universal Product Code). Existe en dos modalidades el código UPC-A y el UPC-E. El código UPC-A emplea 12 caracteres numéricos exclusivamente y se presenta en la Tabla 1.3, "Descripción del Código UPC-A".

Tabla 1.3. Descripción del Código UPC-A

Carácter	Descripción
1	Dígito de verificación del código (check character).
2-6	Identificación del producto.
7-11	Identificación del fabricante.
12	Tipo de producto.

El código UPC-E (código reducido) se conoce como "cero suprimido", ya que elimina por lo menos 4 ceros del código. No siempre es factible utilizarlo ya que depende del número asignado al fabricante y el número de productos correspondiente. Para establecer el código UPC-E es necesario observar las 4 normas de supresión de ceros siguientes.

1. Si el número del fabricante termina en 00, precedido por 0,1 o 2; 1000 productos podrán ser codificados por con UPC-E.
2. Si el número del fabricante termina en 00 precedido por 3 al 9; 100 productos podrán codificarse.
3. Si el número del fabricante termina en 0; 10 números de productos podrán asignarse.
4. Si el número del fabricante no termina en 0 tan solo 5 artículos podrán utilizar la versión reducida.

1.2.6. Infraestructura de Llave Pública

Es conocida como PKI por sus siglas en inglés *Public Key Infraestructure*. La PKI es una arquitectura de seguridad que ha sido desarrollada para proveer el nivel más alto de seguridad al intercambiar información.

De manera general, una PKI es el conjunto de hardware, software, personas especializadas, políticas y procedimientos necesarios para crear, administrar, almacenar, distribuir y revocar certificados de llave pública.

Los sistemas actuales de seguridad son muy vulnerables a ataques malintencio-

nados y en consecuencia, no satisfacen las necesidades de seguridad de los usuarios. Una buena PKI ofrece un nivel muy alto de seguridad para todas las actividades realizadas.

Para lograr obtener una PKI robusta estos son algunos de los componentes más importantes que debe incluir:

- Autoridades Certificadoras (CA)
- Autoridades de Registro (RA)
- Protocolos de administración de Certificados (CMP)
- Repositorio de los certificados
- Tarjetas inteligentes
- Mecanismos biométricos
- Reconocimiento de patrones

Tabla 1.4. Características de algunos componentes PKI

Autoridad Certificadora	• Entidad confiable encargada de gestionar la emisión y generación de certificados y listas revocación. • Pieza clave de una PKI. • Las tareas de una CA son críticas ya que toda la responsabilidad de la seguridad recae en ellas. • En el momento en que se confía en una CA, se debe tener en cuenta cuál es la política de seguridad (Política de Certificación). • Expedición de certificados. • Revocación de certificados digitales. • Almacenamiento y publicación de los certificados

	emitidos. • Establecer la fecha de revocación de los certificados.
Repositorio de Certificados	• El repositorio generalmente se asocia con un directorio, pero no necesariamente es el caso. • Base de Datos con información de certificados disponibles. • Servidores de directorio basados en X.500 con un cliente de acceso vía LDAP (Lightweight Directory Access Protocol). • Información almacenada jerárquicamente mediante una estructura de árbol. • Trabaja en HTTP (Hyper Text Transfer Protocol) o FTP (File Transfer Protocol).

1.2.7. Certificados digitales

Un certificado es información con respecto a la clave pública que ha sido firmada por una Autoridad Certificadora (CA). La información encontrada en el certificado se basa en el estándar X.599 V3

Los certificados dentro de este estándar incluyen información asociada a la identidad del propietario, así como el periodo de validez del certificado.

1.2.8. Firma electrónica

Una firma electrónica o firma digital es un conjunto de datos asociados que permiten asegurar la identidad de una entidad, así como la integridad de la información emitida por ésta última. Se consiguen bajo criterios específicos de autenticidad, integridad y confidencialidad.

A medida que son cada vez más las transacciones gubernamentales que pueden realizarse a través de Internet, ha surgido la necesidad de un formato electrónico

para la prueba de la identidad, o firma electrónica.

PKI o infraestructura de llave pública es el nombre que se le da al proceso de transferir información cifrada electrónicamente utilizando una llave pública para cifrar la información y una llave privada para descifrarla una vez recibida. El uso de la llave privada también funciona para identificar al usuario al darle una identidad electrónica exclusiva, debido a que la llave privada es única.

Para aumentar la seguridad del uso de estas llaves privadas, serán emitidas por una autoridad responsable, que requiere que la persona se identifique plenamente antes de generar sus llaves.

Los beneficios que se obtienen con el establecimiento de una firma electrónica son los siguientes:

- Confirmación de la integridad y calidad de la información enviada/recibida electrónicamente.
- Confirmación de la fuente y destino de esa información.
- Seguridad en tiempo de la información.
- Confirmación de la privacidad de la información.

1.2.9. Contraseñas

Las contraseñas son utilizadas casi exclusivamente para identificar a los usuarios de sistemas de cómputo. Para lograr una mejor seguridad todos los sistemas de un centro de cómputo deberían ser protegidos mediante contraseñas para evitar accesos no autorizados.

Existen algunas reglas básicas aplicables al uso de contraseñas, orientadas a asegurar que los usuarios no autorizados en el sistema no puedan descubrirlas.

- Las contraseñas no deben dejarse escritas a la vista en sitios públicos.
- No deben de ser demasiado cortas (algunos sistemas imponen una longitud míni-

ma).

- No deben de ser compartidas con colegas, familiares o amigos.
- En el caso de sistemas sensibles es deseable llevar un registro de cuales contraseñas son usadas, para no volverlas a repetir.
- Limitar el número de intentos fallidos de utilización de la clave en una misma sesión.
- Cuando a una persona se le asignó una contraseña para ingresar a un sistema y esta persona deja de ser un usuario válido, la contraseña debe revocarse.
- Los administradores del sistema necesitan tener la capacidad de actualizar la contraseña de los usuarios que la han olvidado.
- Las personas con contraseña de acceso a un equipo de cómputo (incluyendo los consultores externos o administradores del sistema) deben tener límitación sobre la información a la que pueden tener acceso.

Capítulo

2.

Análisis del Problema

2.1. Discusión de los métodos existentes de identificación Personal.

2.1.1. Introducción

Dadas las necesidades de proteger la privacidad y la integridad de la información digital que se almacena en computadoras o que se transmite por redes públicas, es necesario implementar métodos de autentificación que contribuyan a éstos objetivos.

La autentificación (también llamada autenticación), es el hecho o el proceso de asegurase de que una entidad (una persona, computadora o un programa de computadora) es en realidad quien declara ser.

La invención de las redes de computadoras constituyó uno de los factores que más complicaron el aseguramiento de la información, la cual se vio aun en más problemas con la introducción de redes públicas de datos (p.e. Internet, redes de teléfonos móviles, etc.). El acceso seguro a información de entidades remotas conectadas a redes públicas implica todavía hoy uno de los problemas fundamentales de la ingeniería de computadoras y la ciencia computacional.

En la última década, las aplicaciones de la criptografía se extendieron a muchas actividades de uso común, entre ellas podemos mencionar las comunicaciones cifradas para uso civil y comercial (PGP [1], S/MIME [2]), el acceso a equipos remotos en comunicaciones (por ejemplo de correo electrónico) se hace preferentemente con los

datos cifrados (SSH , SSL, IPSec [3]) e incluso, en algunos ámbitos se habla de implementar la legalidad de firmas electrónicas o sistemas electorales electrónicos.

2.1.2. Problemática del uso de contraseñas

2.1.2.1. Orígenes

Actualmente existen muchos tipos de mecanismos de autentificación, la mayor parte de ellos basados en contraseñas. Una contraseña* no es más que una cadena de caracteres que el usuario o programa debe mantener en secreto para identificarse ante otra entidad, que comparte el secreto. En este tipo de mecanismos, se asume que sólo la persona autorizada para conocer o para modificar la información posee la contraseña, y de esta manera, se supone que el conocimiento de dicha contraseña garantiza que el usuario es auténtico. Así, todos los mecanismos basados en contraseñas confían en un sólo factor* para la autentificación: lo que alguien sabe.

En la Facultad de Ingeniería, específicamente en los servicios que implican manejeo de computadoras, los sistemas utilizan únicamente la autentificación por contraseñas para asegurar la información. No tenemos conocimiento de otros mecanismos (exceptuando por supuesto la seguridad física) que protejan la información de posibles ataques. Ésto cobra aún más importancia cuando la tendencia es justamente a automatizar procesos y a utilizar nuevas tecnologías para el préstamo de servicios (Educación a distancia, portales en Internet, Internet2, trámites en línea, etc)

Los sistemas de autentificación basados en contraseñas fueron los primeros en utilizarse y continúan siendo ampliamente utilizados en todos los campos de la computación. Sin embargo, a través del tiempo, múltiples problemas han surgido con el uso de estos sistemas. Las soluciones propuestas a éstos muchas veces resultaron ser insuficientes o causar nuevos problemas. Con el tiempo, los protocolos y algoritmos fueron mejorando, resolviendo uno a uno los problemas y volviéndose más complicados.

---

[1]) *Pretty Good Privacy*. Sofware utilizado para cifrar y descifrar correo electrónico, y firmar documentos digitalmente [PGP].

[2] *Secure Multi-Purpose Internet Mail Extensions*. Es una forma de enviar correo electrónico cifrado utilizando el sistema RSA.

[3] Conjunto de protocolos de seguridad en la capa de red (nivel de procesamiento de paquetes) dentro de una red de datos.

2.1.2.2. Problemas técnicos

Por otra parte, sin importar que tan complejos sean los sistemas de autentificación por contraseñas, existen algunas características deseables cuya implementación ha representado durante años un gran obstáculo para ellos. La descripción detallada de estos problemas esta fuera de nuestro objetivo, sin embargo consideramos necesario mencionar al menos algunos de los más importantes, que se presentarán a continuación.

2.1.2.2.1. Inmunidad a ataques de diccionario

Un ataque de diccionario es básicamente el intento de adivinar una contraseña usando un espacio reducido de palabras comunes. Por ejemplo, en los sistemas operativos UNIX, una contraseña en `/etc/passwd` es típicamente una cadena de 64 bits de largo generada con una llave de 56 bits. Para buscar entre todas las llaves posibles, se tendrían que hacer 256 intentos. Sin embargo, dado que la llave no es más que una función simple de la contraseña, un atacante podría crear un diccionario de contraseñas comunes e intentar con ellas primero. Incluso con un diccionario de un millón de palabras, buscar en todo el diccionario no tomaría ni una trillonésima parte de lo que tomaría un ataque de fuerza bruta (en el que se busca la llave entre todas sus posibles combinaciones). Dado que los usuarios tienden a escoger contraseñas que puedan memorizar fácilmente, los ataques de diccionario permiten a los atacantes ahorrarse mucho tiempo de procesamiento.

Un ataque de diccionario requiere entonces 1) que los usuarios utilicen contraseñas débiles, esto es, fáciles de adivinar, basadas en alguna palabra existente en un diccionario; y 2) que el atacante posea una copia del archivo de contraseñas cifradas. Desgraciadamente, también se han implementado ataques de diccionario basados en red, los cuales ni siquiera requieren la posesión de tal archivo. Para ilustrar ésto, analizamos la vulnerabilidad de un protocolo desafío-respuesta muy simple, donde Anita trata de convencer a Beto de que conoce la contraseña C:

- Anita se identifica ante Beto.
- Beto manda a Anita el desafío R.
- Anita calcula H(R,C) y se la envía a Beto.
- Beto compara lo recibido con H(R,P) calculado con su propia copia de P.

Todo esto donde C es la contraseña que comparten ambos y H(x,y) es una función hash segura y unidireccional.

Una atacante Eva que ha estado observando y copiando toda esta información podría verificar la veracidad de una C' adivinada, calculando H(R,C') y comparándola con el valor que capturó de H(R,C) durante una autentificación exitosa de Anita. Si los valores coinciden, la C' que adivinó, es idéntica a C, y ahora puede hacerse pasar por Anita.

2.1.2.2.2. No Equivalencia a texto puro

Se dice que un dato es equivalente al texto puro de una contraseña cuando con él, podemos obtener el mismo nivel de acceso que con la contraseña misma. Por ejemplo, consideremos el siguiente intento ingenuo de mejorar un protocolo de contraseñas:

- Anita ingresa su contraseña directamente en el software cliente.
- En vez de mandar la contraseña directamente a través de la red, el software cliente la cambia por la función hash estándar de Unix.
- Beto, el servidor, compara directamente la cadena enviada a través de la red con el hash de la contraseña de Anita. Si coinciden, la autentificación es exitosa.

A pesar de que el texto puro de la contraseña nunca se envió ni se guardó en el sistema, un atacante podría perpetrar el siguiente ataque: después de obtener el hash de la contraseña de Anita - ya sea desde el archivo local o valiéndose de monitoreo de la red - inicia una conexión con Beto, y envía el hash de la contraseña cuando el servidor se la pide. Sin saber la contraseña propiamente, y sin siquiera haber tenido que invertir tiempo para encontrarla en claro, el intruso ha obtenido acceso a la cuenta de Anita en Beto.

Este problema persiste incuso en protocolos más complejos, como el de desafío-respuesta, donde una función hash unidireccional se utiliza para esconder las contraseñas almacenadas. El protocolo de autentificación funciona como sigue:

- Anita se identifica ante Beto.

- Beto manda a Anita la cadena de desafío R.
- Anita calcula H(R,f(C)) y la manda a Beto.
- Beto calcula H(R,f(C)) con su valor almacenado de f(C) y lo compara con lo que recibió de Anita. Si los valores coinciden, la autentificación fue exitosa.

De nuevo, la contraseña C nunca se almacenó en el sistema en texto puro, pero un atacante puede utilizar la equivalencia a texto puro de f(C), buscando éste valor en la base de datos de contraseñas, y hacerse pasar por Anita. Este fenómeno se da porque Anita siempre utiliza f(p) para calcular H.

En general, es bastante sencillo diseñar protocolos de contraseñas donde ambas partes comparten un secreto, y donde se realizan operaciones simétricamente basándose en este secreto y en el intercambio de mensajes. Incluso cuando las operaciones son muy complejas, el riesgo de causar que el protocolo deje de funcionar correctamente es muy bajo, pero al mismo tiempo la equivalencia a texto puro es muy difícil de evitar. Así, evitar la equivalencia a texto puro se requiere seleccionar operaciones matemáticas con mucho cuidado, que en la mayoría de los casos resultan ser operaciones que actúan asimétricamente entre cliente y servidor. De aquí se puede explicar la relativa abundancia de protocolos de cifrado simétricos en comparación a los pocos que existen que son asimétricos.

En muchos sistemas utilizados actualmente, la protección a las bases de datos que contienen las contraseñas se limitan a los mecanismos inherentes al sistema operativo. Los archivos que contienen contraseñas tendrán que ser leídos en algún momento ya que el servidor está obligado a hacerlo para realizar la autentificación. En algún momento, un acceso no autorizado a estas bases de datos se puede dar, comprometiendo la seguridad de todo el sistema. Sólo los nuevos protocolos de contraseñas solucionan este problema, haciendo que el archivo de contraseñas sea completamente inútil para el atacante, incluso haciéndolo leíble para todo el mundo. Así, la seguridad reside en el algoritmo mismo y no en el sistema operativo.

2.1.2.2.3. Resistencia serial (forward secrecy)

La característica de ser resistente de manera serial o traducido literalmente "secreto hacia adelante", cuando hablamos de sistemas de autentificación basados en contraseñas, nos indica el grado de resistencia que tiene un sistema a comprometer información secreta si una parte del sistema ha sido ya comprometida. Para ilustrar la im-

portancia de ésta característica, citemos por ejemplo a los mecanismos de autentificación que usan llaves de sesión para cifrar todo el tráfico de información además de la información inherente a la autentificación:

- Anita y Beto intercambian llaves aleatorias, Anita manda R a Beto y Beto manda S a Anita.
- Ambos calculan la llave de sesión K = H(R,S,C) donde H es una función hash y C es la contraseña compartida.
- Ambos verifican que su K es la misma, y comienzan a cifrar los mensajes de su sesión con ella.

Si la contraseña C fuera comprometida por alguna razón, (los usuarios suelen anotarlas, compartirlas por teléfono, o por mensajeros instantáneos, dejar abiertas sesiones de correo o incluso escoger una que se adivina fácilmente.) nuestra intruso Eva, que con regularidad monitorea y almacena las transmisiones entre Anita y Beto, podría descifrar todas las transmisiones mientras C sea usada como contraseña. El protocolo no es secreto hacia adelante, porque, dado que la manera más fácil de cambiar la contraseña de Anita es usando el comando apropiado, Eva podría descifrar también ese comando, revelándose la nueva contraseña, o averiguar una vieja contraseña y descifrar el tráfico que había almacenado antes, hasta encontrar la nueva.

Este ataque también trabaja con una llave de sesión comprometida: si nuestra atacante Eva descubriera un valor específico de K para una sesión, podría iniciar un ataque de diccionario contra C, calculando H(R,S,C') con C' posibles y comparando el resultado contra el valor de K capturado. A este tipo especial de ataque de diccionario se le llama ataque de Denning-Sacco, descrito en una publicación de la ACM [4] en 1981. Muchos protocolos existentes mostraron ser vulnerables a dichos ataques.

2.1.2.2.4. Conclusión

No fue sino hasta la década de 1990 cuando los primeros algoritmos y protocolos seguros para mantener e intercambiar contraseñas fueron desarrollados [WU-1998]. Entre estos algoritmos podemos mencionar a EKE (Encrypted Key Exchange, 1992), SPEKE (Strong Password Exponential Key Exchange, 1996), SRP (Secure Remote

[4] Association for Computing Machinery [ACM].

Password, 1998).

2.1.2.3. Problemas humanos

Si bien estos algoritmos solucionan problemas muy complejos a nivel lógico como los tres descritos en esta sección, existen otros problemas inherentes a la naturaleza de los usuarios que los mejores algoritmos nunca podrán resolver. La siguiente es una lista de conductas comunes entre usuarios, las cuales representan serios riesgos de seguridad en un sistema de autentificación basado en contraseñas:

- Anotar una contraseña en un papel que después dejan pegado frente al monitor de su estación de trabajo.
- Utilizar siempre la misma contraseña para sus cuentas en sistemas que no están relacionados (correo electrónico personal, cuenta del sistema de la empresa en que labora, páginas de comercio electrónico, etc.)
- Compartir la contraseña con otros usuarios.
- Enviar la contraseña por un canal inseguro (teléfono, mensajeros instantáneos, correos electrónicos)
- Elegir una contraseña tan débil que se puede adivinar sin siquiera realizar una ataque de diccionario. (fecha de nacimiento, apodo, nombre de la mascota, etc.)
- Introducir sin la precaución de asegurarse que nadie esta viendo lo que se escribe.

Recordemos que aún tratándose de un usuario sin muchos privilegios, una intrusión al sistema es un gran riesgo de seguridad, pues se tiene acceso a información clasificada y además se sabe que muchos ataques exitosos se realizan de manera local, no remota [STRUCK].

2.1.3. Esquemas de factor dual: tokens criptográficos

2.1.3.1. Otros factores de autentificación

Ante la imposibilidad de los mejores algoritmos de autentificación basados en contraseñas para asegurar la información ante los riesgos del mal manejo de un secreto, surgen esquemas de seguridad donde la autentificación depende de varios factores,

y no sólo de lo que alguien sabe. En general, entre más factores tenga un sistema de autentificación, más seguro será éste [KIRCH-2003]. Además del secreto que representa una contraseña, un factor de autentificación adicional podría ser, por ejemplo:

2.1.3.1.1. Lo que alguien es

"Lo que alguien es" se refiere a la existencia de cierto patrón único en la naturaleza física de los usuarios, por ejemplo las huellas digitales, formas y colores en el iris o la forma de la cara. Un sistema que autentifica a las personas basándose en alguno de éstos patrones, se llama sistema de autentificación biométrico*. Un ejemplo del uso de éste factor de autentificación es cuando alguien nos llama por teléfono y lo reconocemos inmediatamente por su voz. El uso del procesamiento digital de imágenes y el reconocimiento de patrones es extensivo en estos sistemas. Las aplicaciones de biometría en materia de autentificación son relativamente nuevas, y hasta el día de hoy no son perfectas. Los inconvenientes del uso de éste factor son el alto costo, la imprecisión de los métodos y la poca flexibilidad que tienen (las características físicas no pueden cambiarse o escogerse).

2.1.3.1.2. Lo que alguien tiene

Este factor exige a la entidad que intenta autentificarse la posesión de alguna cosa que demuestra su autenticidad. Un ejemplo de esto es cuando un oficial de policía nos muestra su placa, o cuando alguien se autentifica ante un servidor remoto ejecutando *ssh* usando llaves guardadas en un archivo de su estación de trabajo.

2.1.3.2. Ventajas de tener dos factores

Por la experiencia sabemos que los sistemas más seguros usan más de un factor de autentificación. El ejemplo más común es el sistema bancario de cajeros automáticos: para acceder exitosamente a una cuenta bancaria, es necesario conocer un secreto (el NIP) pero también se necesita poseer una tarjeta al mismo tiempo. Esto da tiempo al dueño legítimo de la cuenta bancaria de actuar en caso de una intrusión (p.e. reportar la pérdida o robo de la tarjeta).

Otro ejemplo es un archivo de llaves criptográficas protegido con algún algoritmo simétrico que requiere de algún secreto (comúnmente llamado passphrase).

Las ventajas de los sistemas de autentificación de factor dual sobre aquellos de un solo factor son obvias:

1. El sistema es más seguro dado que es menos probable que un intruso logre poseer ambos factores de autentificación al mismo tiempo.

2. El sistema entero posee una resistencia serial mayor (forward secrecy); al no comprometer a todo el sistema si una parte de él ha sido violada (de nada nos sirve tener sólo uno de los factores de autentificación). Esto proporciona además, más tiempo de actuar ante una contingencia.

2.1.3.3. Tokens criptográficos: Tarjetas inteligentes

Una token criptográfico (también conocido como token de seguridad o token de autentificación) es un dispositivo de hardware de tamaño pequeño, que su dueño porta para obtener acceso a un recurso informático o a algún recurso de red. Los tokens criptográficos implementan la autentificación de factor dual al requerir un número de identificación personal (NIP) o algún otro secreto para hacer uso de él. Las tarjetas inteligentes (descritas en el capítulo 3) que usan microprocesadores y coprocesadores criptográficos son el tipo de tokens de autentificación más usados, aunque no los únicos: también existen tokens en otras formas y con estándares diferentes, por ejemplo con formas de llaveros o anillos.

Muchas tarjetas inteligentes han sido especialmente diseñadas para funcionar como dispositivos de autentificación, y en términos muy generales funcionan albergando una o más llaves criptográficas que se usan para computar una firma digital y de esa manera autentificar al portador de la tarjeta, pero antes de poder hacer uso de éstas llaves, será necesario presentarle a la tarjeta un secreto (NIP) [STRUCK].

2.1.3.4. Ventajas de usar Tarjetas Inteligentes

Además de todas las ventajas de un sistema de autentificación de factor dual, las tarjetas inteligentes tienen algunas características especiales que las hacen una excelente opción para usarse en sistemas de autentificación.

Primero, las llaves criptográficas se guardan sólo en un lugar, lo que ayuda a reducir el riesgo que implica que el usuario tenga varias copias de éstas en diferentes máquinas (una computadora portátil y una estación de trabajo, por ejemplo).

La característica de destruir la información que contienen después de algunos intentos fallidos de utilización (generalmente 5 o 10) hace de las tarjetas inteligentes dispositivos resistentes a ataques de diccionario. Esto aporta una gran ventaja sobre

implementaciones de autentificación de factor dual donde las llaves criptográficas se almacenan en archivos cifrados por algoritmos simétricos, ya que la potencial obtención de estos archivos podría permitir a un atacante llevar a cabo un ataque de este tipo sobre ellos, mientras que en el caso de una tarjeta inteligente que almacena llaves, éstas simplemente se destruirán ante un intento de esta naturaleza [KIRCH-2003].

2.1.3.5. Seguridad Física

Por último, las tarjetas inteligentes poseen características físicas que las hacen muy resistentes a violaciones para la mayor parte de los individuos. Se han realizado diversos estudios que indican que violar físicamente una tarjeta inteligente relativamente moderna requiere de una inversión de dinero bastante grande [ANDERSON-KHUN-1996], probablemente sólo accesible para organizaciones con fondos grandes, capaces de reunir grupos de especialistas, y con un interés tan grande que justifique dicha inversión.

Las tarjetas inteligentes no son inviolables y probablemente nunca lo serán, pero los fabricantes han puesto atención en imponer retrasos y dificultades a los atacantes, de manera que dicho ataque sea costoso [ANDERSON-KHUN-1997], [CHAUMETTE-HATCHONDO].

Así, las tarjetas inteligentes con microprocesadores criptográficos se presentan como un medio para almacenar llaves mucho más seguro que los archivos comunes ya que son inmunes a ataques de diccionario, reducen en riesgo de ataques, y son prácticamente inviolables excepto por organizaciones con fondos muy grandes.

2.1.3.6. Desventajas de usar tarjetas inteligentes

Por su puesto, también existen inconvenientes del uso de tarjetas inteligentes. El principal es el costo adicional que implica la inversión inicial: si se quiere hacer autentificación con tarjetas inteligentes, hará falta la infraestructura que genere las llaves y los certificados para los usuarios y algún mecanismo como LDAP* para guardarlos. Además habrá que capacitar a los usuarios para usar adecuadamente las tarjetas y por último, los costos de los lectores y las tarjetas por sí mismas. Como ejemplo de estos costos, mencionamos los precios a los que adquirimos el equipo que utilizamos para la presente investigación: Aproximadamente 30 USD por cada lector y 20 dólares por cada tarjeta.

2.2. Recopilación de requerimientos

Una vez analizados los mecanismos de identificación existentes en la Facultad de Ingeniería, y conociendo someramente los diferentes mecanismos de identificación que existen en el mercado, exponemos lo que a nuestro parecer debería ser la forma de identificar alumnos en nuestra facultad. La Facultad de Ingeniería requiere de un modelo de identificación con las siguientes características:

- **Universal.**

  Que permita identificar a un alumno en todas las situaciones donde es necesario. El modelo debe permitir a **todos** los prestadores de servicios en la facultad de ingeniería identificar a cualquier alumno en el momento que el alumno requiera cualquiera de éstos servicios.

- **Seguro.**

  El modelo provisto debe ser seguro. Por seguridad entendemos que ninguna persona pueda cambiar la identidad de ningún alumno ante los sistemas que implementen al modelo. También se entiende que ningún alumno debería tener acceso a servicio alguno sin haber pasado por un proceso de identificación previo. Mecanismos adicionales deberían ser agregados para garantizar éstas dos premisas. El análisis de registros y la puesta en marcha de sistemas de detección de intrusos (*IDS*) son ejemplos de éstos mecanismos.

- **Privado.**

  En el sentido de la información privada de los alumnos deberá ser manejada únicamente bajo su conocimiento y su consentimiento. Excepciones a este principio sólo deberían concederse bajo condiciones muy restringidas, por ejemplo, para los administradores de sistemas.

- **Independiente de la plataforma.**

  Los sistemas que implementen al modelo, deberían ser capaces de operar en el mayor número posible de plataformas de cómputo, entendiéndose por éstas las combinaciones de hardware (arquitecturas de computadoras, redes de datos) y software (sistemas operativos, lenguajes de programación, bibliotecas de procedimientos) que existen en el mercado y que son susceptibles de ser usadas por servicios de la Facultad de Ingeniería.

- **Escalable.**

  Las tecnologías usadas para la implementación del modelo de identificación deberían ser capaces de corregir rápidamente errores que comprometan su seguridad, de integrar cambios con el objetivo de adecuarse a cambios en el modelo.

- **Cómodo.**

  Los usuarios y los prestadores de servicios requieren de mecanismos de identificación que sean cómodos y no impliquen una excesiva pérdida de tiempo.

- **Propio.**

  Puesto que consideramos la creación de un modelo de éste tipo una tarea con un valor ampliamente de ingeniería, el modelo propuesto y los sistemas que lo implementen deberían estar creados (al menos en una gran parte) por los miembros de la comunidad de la propia Facultad de Ingeniería.

- **Económicamente Viable.**

  Es decir, que sea posible implementar el modelo rápidamente, haciendo una inversión de dinero que se pueda recuperar con los ahorros que el mismo sistema de identificación produzca.

- **Libre.**

  En el sentido de que el conocimiento generado por una institución de educación publica como la UNAM debería estar disponible para su libre uso por toda aquella persona que pueda resolver un problema usando este conocimiento. Sería muy deseable que el modelo de identificación pudiera llevarse a otras instituciones con características similares.

2.3. Definición

Después de haber analizado un problema que nosotros mismos padecimos durante nuestra estancia en la Facultad de Ingeniería de la UNAM y tomando en cuenta las tendencias tecnológicas y comerciales que existen en el rubro de la identificación de personas, nos hemos planteado la siguientes preguntas:

- ¿Es posible crear un modelo de identificación para los alumnos de la Facultad de Ingeniería, que sea implementable con la tecnología actual de tarjetas inteligentes?

- ¿Qué tanto nos podemos acercar al modelo ideal establecido en la recopilación de requerimientos?

Capítulo

3.

Conceptos básicos

« And above all society needs to encourage the spirit of voluntary cooperation in its citizens. When software owners tell us that helping our neighbors in a naturai way is "piracy", they pollute our society's civic spirit »

-Richard M. Stallman, FSF

3.1. Tarjetas inteligentes

3.1.1. Introducción

Este documento tiene como objetivo proporcionar una panorámica general acerca de las Tarjetas Inteligentes; de manera que se revisará la historia de su desarrollo, los tipos de tarjetas que existen actualmente y sus aplicaciones. Además se incluye una explicación breve de sus características físicas y de las funciones que realiza su sistema operativo.

Este documento no pretende servir como referencia para el desarrollo de ninguna aplicación que involucre la utilización de tarjetas inteligentes, si no como una introducción que facilite la investigación posterior de documentos más especializados.

3.1.2. Evolución de las tarjetas de identificación personal

Los inicios de las Tarjetas Inteligentes que conocemos hoy en día se dieron en Estados Unidos a principios de los años 50's, cuando la cadena de restaurantes Diners Club produjo la primer tarjeta hecha completamente de plástico para utilizarse en su sistema de pagos. Para construir las tarjetas utilizó PVC, un material sintético que proporciona un tiempo de vida mucho mayor en comparación con las tarjetas de papel que se utilizaban en esos días. En este sistema sólo era necesario que el cliente recibiera una tarjeta Diners Club para que pudiera realizar pagos al presentarla en lugar de efectivo. La tarjeta identificaba a cada cliente como miembro de un grupo selecto de personas, y era aceptada por algunos restaurantes y hoteles que reconocían a dicho grupo. El único tipo de verificación que se llevaba a cabo era que la tarjeta sólo era expedida a miembros, por lo que se suponía que el portador de la tarjeta correspondía con el nombre impreso en ella.

VISA y MasterCard entraron entonces al mercado, pero los costos generados por fraude, falsificación, cargos bancarios y manipulación de las tarjetas hicieron necesaria la creación de una tarjeta cuya validación pudiera ser efectuada automáticamente por algún dispositivo. Fue entonces cuando se crearon las tarjetas de cinta magnética, lo que permitió introducir más información en la tarjeta además del nombre del usuario en un formato digital que pudiera ser leído por un dispositivo electrónico. Actualmente este tipo de tarjetas son las más utilizadas como método de pago. Sin embargo, las tarjetas de cinta magnética tienen una gran debilidad: cualquier persona que tenga acceso al dispositivo adecuado puede leer, reescribir o borrar la información contenida en ellas. Así, una tarjeta de cinta magnética no puede utilizarse para guardar información de manera confiable, sin mencionar que requiere una gran infraestructura centralizada (y en la mayoría de los casos remota) para realizar la verificación y el procesamiento de la información.

A pesar de las desventajas mencionadas anteriormente, el uso de tarjetas de cinta magnética se hizo más común, y la infraestructura requerida para su funcionamiento poco a poco estuvo disponible con un crecimiento mayor en Estados Unidos.

En cualquier arquitectura cliente - servidor existen dos soluciones para la falta de poder de procesamiento: la primera es aumentar la capacidad de procesamiento del lado del servidor, y la segunda es aumentar el poder de procesamiento de las terminales, de manera que parte del trabajo lo realiza el cliente. Al parecer, los países europeos se decidieron por la solución del lado del cliente, y realizaron un avance enorme sobre la tecnología de cinta magnética al crear las llamadas Integrated Circuit

Cards (ICC), o Tarjetas basadas en Circuitos Integrados.

En 1968, los inventores alemanes Jürgen Dethloff y Helmut Grötrupp hicieron una solicitud para obtener la primera patente relacionada con ICC. Solicitudes similares fueron hechas en Japón en 1970 y en Francia en 1974. En 1984, los servicios Postales y de Telecomunicaciones de Francia hicieron un gran avance al desarrollar las primeras tarjetas telefónicas. Para 1986, varios millones de tarjetas telefónicas hechas en Francia se encontraban en circulación.

Como la criptografía tuvo un gran progreso en los años 60's al grado de poder evaluar matemáticamente los mecanismos de seguridad, las tarjetas inteligentes se convirtieron en un medio ideal para almacenar de manera confiable llaves y algoritmos criptográficos. Los bancos franceses fueron el primer campo de aplicación de este tipo de tarjetas al utilizar una tarjeta bancaria que incluía un chip, en 1984. Después fueron los bancos alemanes, al utilizar las tarjetas bancarias cerca del año de 1997. Otra aplicación establecida en Alemania fueron las más de 70 millones de tarjetas inteligentes utilizadas para almacenar información sobre seguros de vida.

3.1.3. Tipos de tarjetas de identificación

El estándar ISO 7810 ("Identification Cards - Physical Characteristics") define propiedades físicas como la flexibilidad, resistencia a la temperatura, y las dimensiones para tres diferentes formatos de tarjetas (ID -1, ID - 2 y ID - 3). El estándar para las tarjetas inteligentes, ISO 7816, está basado en el formato ID - 1 [ISOOverview].

Para tener una mejor perspectiva de las tecnologías existentes, a continuación se mostrarán varios tipos de tarjetas que obedecen el estándar ID - 1.

3.1.3.1. Tarjetas de relieve (embossed cards)

Este tipo de tarjetas tiene la información en forma de relieve sobre ella, lo que permite que ésta sea fácilmente transferible al papel utilizando un dispositivo muy sencillo. El estándar ISO 7811 especifica las formas de las marcas de relieve, su tamaño y su posición en la tarjeta.

3.1.3.2. Tarjetas de cinta magnética (magstrip cards)

La ventaja principal que ofrece la tecnología de cinta magnética sobre la de relieve es

que disminuye la cantidad de documentos de papel. Las secciones 2, 4 y 5 del estándar ISO 7811 especifican las características de la cinta magnética, las técnicas de codificación y la posición de la cinta dentro de la tarjeta. La capacidad de almacenamiento de la cinta magnética es de aproximadamente 1000 bits. Cualquier persona que posea un dispositivo de lectura y escritura puede tener acceso a la información en la tarjeta.

3.1.3.3. Tarjetas inteligentes (smart cards).

Las tarjetas basadas en Circuitos Integrados (ICC) son conocidas comúnmente como "Tarjetas Inteligentes". Este tipo de tarjetas son la más nueva adición a la familia de tarjetas ID - 1 , y siguen además las especificaciones del estándar ISO 7816 [ISOOverview] [ISO7816 part[1-3]][ISO7816 part[4]].

Las tarjetas inteligentes tienen una capacidad de almacenamiento mucho mayor a las basadas en tecnología de cinta magnética, con un máximo aproximado de hasta 64 Kbytes en ROM y EEPROM en las tarjetas que existen actualmente y tienen un tiempo de vida mucho mayor; sin embargo, la ventaja más importante sobre las tarjetas de cinta magnética es que la información almacenada puede ser protegida contra accesos no autorizados y / o falsificación. Las operaciones de lectura y escritura realizadas en la memoria pueden ser ligadas a condiciones específicas controladas tanto por software como por hardware.

3.1.3.4. Tarjetas de memoria (memory cards)

Aunque también son conocidas como Tarjetas Inteligentes, este tipo de tarjetas generalmente ofrecen una funcionalidad mucho menor que las tarjetas que cuentan con un microprocesador, por lo que su precio es mucho menor a tal punto que conviene comprar una nueva antes que tratar de actualizarla. Contienen memoria EEPROM y ROM y circuitos que implementan la lógica necesaria para direccionamiento y seguridad.

Los diseños más austeros implementan solamente lógica para protección de escritura sobre la información contenida en la tarjeta. Diseños más completos implementan además lógica para proteger tanto la escritura como la lectura sobre la memoria. Una aplicación típica para este tipo de tarjetas son las tarjetas telefónicas de prepago.

Este tipo de tarjeta funciona como un simple almacén de información que el usuario modifica cuando realiza una transacción con ella. Es así por ejemplo, como en el caso de telefonía la tarjeta viene de fábrica con el contenido de minutos que el usuario puede ocupar. Al hacer una llamada con la tarjeta, la máquina en cada minuto va descontando uno de los minutos que trae la tarjeta, de esa manera se evita que el usuario se sobregire.

3.1.3.5. Tarjetas con microprocesador (Integrated Circuit Cards, ICC)

Este tipo de tarjetas contienen un CPU y memorias RAM, ROM y EEPROM. El sistema operativo de la tarjeta típicamente es almacenado en memoria ROM; el CPU utiliza la memoria RAM para realizar operaciones, y la memoria EEPROM para almacenar información.

Una arquitectura típica para una tarjeta inteligente tiene las características que se muestran a continuación:

Tabla 3.1. Características según el tipo de tarjeta inteligente

Tipo de tarjeta	Características
RAM	De 256 bytes a 2 Kbytes
EEPROM	De 8 Kbytes a 32 Kbytes
ROM	De 8 Kbytes a 64 Kbytes
Microprocesador	8 bits de 1 a 5 MHz
Interfaz de comunicación	9600 bps mínimo, half duplex

La interfaz serial de comunicación de entrada - salida generalmente consiste de un solo registro a través del cual se transmite la información bit a bit. Aunque el chip contenido en la tarjeta es un microprocesador completo; tanto el voltaje como la referencia a tierra y la señal de reloj deben ser proporcionadas por la terminal externa. Actualmente existen tarjetas con procesadores de hasta 32 bits, aunque su utilización se reserva sólo para aquellas que realizan operaciones que requieren de un poder de cómputo mayor (como las tarjetas con funciones criptográficas).

3.1.3.6. Tarjetas con coprocesador criptográfico

Estrictamente hablando, este tipo de tarjetas también son tarjetas con microprocesa-

dor (ya que contienen uno), sin embargo es conveniente tratarlas por separado debido a las diferencias en su funcionalidad y costo.

Debido a que los algoritmos criptográficos simétricos que existen en la actualidad requieren de cálculos matemáticos que utilizan números enteros muy grandes, a un microprocesador de 8 bits con RAM limitada le tomaría varios minutos realizar una codificación sobre una llave privada de 1024 bits; sin embargo, al agregar un coprocesador criptográfico a la arquitectura el tiempo se reduce a unos cuantos microsegundos.

El coprocesador criptográfico incluye unidades aritméticas especialmente desarrolladas para realizar exponenciales y operaciones con números enteros de gran tamaño de forma más rápida y eficiente.

La mayor desventaja de este tipo de tarjetas es que el costo sobre una tarjeta normal con microprocesador aumenta de un 50% a un 100%; sin embargo. la posibilidad de realizar operaciones de autentificación, firma digital y no repudio sin que la llave privada del usuario deje la tarjeta hace que el aumento de precio valga la pena.

3.1.3.7. Tarjetas inteligentes sin contacto (contactless smart cards)

Aunque la confiabilidad de las tarjetas inteligentes de contacto a aumentado a niveles aceptables a través de los años, el contacto entre los componentes es una de las causas más comunes de falla no sólo en esta tecnología, si no en cualquier tipo de sistema electromecánico, principalmente debido al polvo y al desgaste por el uso continuo.

Las tarjetas sin contacto resuelven éste problema al utilizar el cuerpo de la tarjeta como una antena, de manera que ya no es necesario insertarla en ningún lector, debido a que este también cuenta con una. La mayoría de las tarjetas inteligentes sin contacto obtienen la energía para el circuito integrado interno a través de la señal electromagnética que se establece entre la tarjeta y el lector.

Son ideales para aplicaciones en las que se requiera una interfaz muy rápida, como por ejemplo una gran masa de clientes que requieren un acceso dinámico, boletería electrónica, identificación de paquetes. etc.

3.1.3.8. Tarjetas de memoria óptica (optical memory cards)

Este tipo de tarjetas contienen una memoria implementada utilizando la misma tecnología que se utiliza para construir discos compactos. La ventaja de utilizar este tipo de almacenamiento es que permite una mayor densidad de información, aunque actualmente sólo existen tarjetas con memoria PROM, es decir, que se escribe una sola vez y puede ser leída muchas veces.

Al igual que con las tarjetas basadas en circuitos integrados, existen principalmente dos tipos de éstas tarjetas: tarjetas de memoria óptica (optical memory cards), y tarjetas de memoria óptica con microprocesador (optical smart cards). La diferencia entre las dos radica en que la primera sólo sirve como medio de almacenamiento, y la otra es una tarjeta inteligente (con todas las características mencionadas anteriormente) que tiene una memoria óptica como medio de almacenamiento. Actualmente, las tarjetas con memoria óptica tienen una capacidad máxima de hasta 4.9 Mbytes, y las tarjetas de memoria óptica con procesador tienen una capacidad aproximada de 1 Mbyte, aunque éste valor varía según el fabricante.

Debido a su capacidad de almacenamiento actualmente son utilizadas como medio de identificación utilizando alguna medida biométrica, ya que los archivos que describen un característica biométrica generalmente son de gran tamaño. La especificación para éstas tarjetas se encuentra en los estándares ISO/IEC 11693 y 11694.

3.1.4. Tarjetas inteligentes

A grandes rasgos, una tarjeta inteligente es una tarjeta de plástico con una pequeña computadora contenida en un chip que se encuentra incrustada en ella [SCBasics]. Como se explicará más adelante, las características de esta computadora proporcionan funcionalidades diferentes a cada tarjeta, de manera que existen tarjetas con procesadores especiales que implementan funciones criptográficas de una forma más eficiente, tarjetas que sólo contienen memoria para almacenar información, etc.

3.1.4.1. Características eléctricas y físicas

Como se mencionó anteriormente, las tarjetas inteligentes pertenecen a la familia ID - 1 del estándar ISO - 7810. Las dimensiones para las tarjetas de esta familia son de 85.6 mm por 54 mm, con un radio de curvatura en las esquinas de 3.18 mm y un grosor de 0.76 mm. Aunque el ISO - 7810 fue creado en 1985, éste no especifica la posición del chip dentro de la tarjeta, sino la posición de marcas de relieve y de cintas

magnéticas. La posición del microprocesador dentro de la tarjeta se encuentra especificado en el estándar ISO - 7816-2, creado en el año de 1988 [ISO7816 part[1-3]].

Otras características físicas como la radiación UV, la radiación por rayos X, el perfil físico de la superficie de la tarjeta, la susceptibilidad electromagnética, su resistencia a la temperatura, etc., se encuentran definidas además en los estándares ISO - 7810, ISO - 7813 e ISO - 7816 sección 1.

Las características eléctricas para las tarjetas inteligentes se encuentran definidas en los estándares ISO / IEC 7816 secciones 2 y 3, y GSM 11.11 [ISO7816 part[1-3]]. La mayoría de las tarjetas inteligentes tienen ocho contactos; sin embargo, como dos de ellos se encuentran reservados para su utilización en el futuro, muchos fabricantes prefieren construir las tarjetas sólo con seis contactos con el objetivo de reducir los costos de producción [PETRI]. Los contactos eléctricos están nombrados del C1 al C8 desde el contacto superior izquierdo hasta el inferior derecho. En la Figura 3.1, "Especificaciones ISO de la tarjeta inteligente " se muestran las especificaciones de las tarjetas inteligentes así como también los contactos eléctricos los cuales se describen en la Tabla 3.2, "Contactos físicos en tarjetas inteligentes".

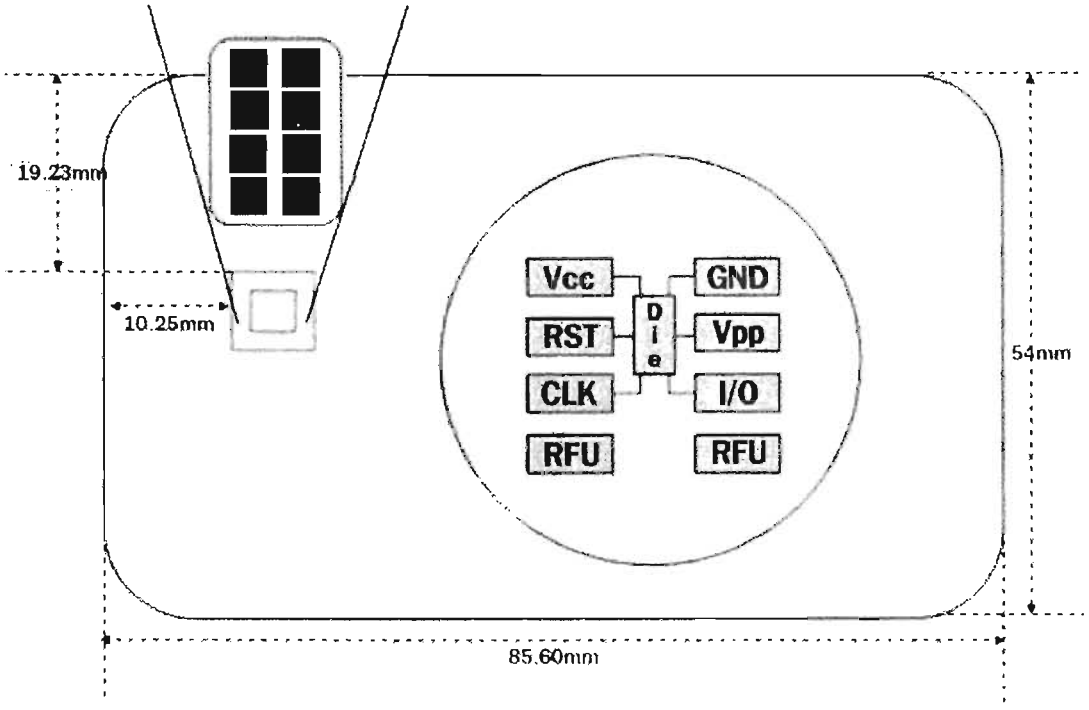

Figura 3.1. Especificaciones ISO de la tarjeta inteligente [SMARTCARD-TECH]

Tabla 3.2. Contactos físicos en tarjetas inteligentes

Contacto	Nombre	Función
C1	Vcc	Voltaje de alimentación (5 [V] +- 10%)
C2	RST	Reset
C3	CLK	Señal de reloj
C4	RFU	Reservado para su uso en el futuro (Reserved for Future Use)
C5	GND	Referencia a tierra
C6	Vpp	Voltaje de programación de memoria
C7	I/O	Comunicación serial de entrada y salida
C8	RFU	Reservado para su uso en el futuro (Reserved for Future Use)

3.1.4.2. Características generales del sistema operativo

El sistema operativo contenido en la tarjeta inteligente se encarga de las siguientes tareas:

- Transmisión de información a través de la interfaz de comunicación serial
- Carga, operación y administración de aplicaciones
- Procesamiento y control de ejecución de instrucciones
- Administración de la memoria (acceso a la información y manipulación de archivos)
- Administración y ejecución de algoritmos criptográficos

El tamaño típico de un sistema operativo se encuentra entre los 3 y los 24 Kbytes. Esta variación en el tamaño es debida a que algunas tarjetas son fabricadas para aplicaciones especificas, mientras que otras tienen la capacidad de almacenar diferentes aplicaciones.

Ya que la memoria en este tipo de dispositivos es un recurso muy limitado, no todas las instrucciones y las estructuras de datos se encuentran implementadas en todos los sistemas operativos. Por esta razón, en los estándares ISO 7816-4 [ISO7816 part[4]] y END 726-3 se crearon los llamados "perfiles", que especifican los

requerimientos mínimos de estructuras de datos e instrucciones que un sistema operativo para una tarjeta inteligente debe implementar.

Como ejemplo podemos tomar el perfil O que se encuentra definido en el estándar ISO 7816-4 y que se muestra a continuación:

Tabla 3.3. Perfil 0 del del estándar ISO 7816-4

Estructuras de datos	• Transparent • Linear fixed • Linear variable • Cyclic
Instrucciones	• READ BINARY, UPDATE BINARY • READ RECORD, UPDATE RECORD • APPEND RECORD • SELECT FILE • VERIFY • INTERNAL AUTHENTICATE • EXTERNAL AUTHENTICATE • GET CHALLENGE

3.1.4.3. Capacidades criptográficas

Actualmente, las tarjetas inteligentes tienen el suficiente poder criptográfico como para ser utilizadas con las aplicaciones y protocolos de seguridad más utilizados.

Las firmas RSA se encuentran disponibles con la capacidad de elegir entre 512, 768 o 1024 bits de longitud para la llave privada [RSASecurity]. Los algoritmos gene-

ralmente utilizan CRT (Chinese Remainder Theorem) para acelerar el procesamiento, de manera que incluso utilizando una llave de 1024 bits, el tiempo de procesamiento se mantiene generalmente debajo del segundo. El archivo que contiene la llave privada se mantiene en EEPROM, y el sistema esta diseñado para que la llave nunca deje la tarjeta, de manera que incluso el dueño de la tarjeta no tiene acceso a ella. Además, el uso de la llave privada se encuentra protegido por un PIN (Personal Identification Number) que sólo conoce el dueño de la tarjeta, por lo que la posesión de la tarjeta no otorga la capacidad de realizar firmas digitales con la llave que se encuentra en ella.

Aunque las tarjetas inteligentes tienen la habilidad de generar pares de llaves con el algoritmo RSA, esta operación puede llegar a ser bastante lenta. El tiempo típico para generar un par de llaves de 1024 bits puede variar desde 8 segundos hasta 3 minutos [PETRI]. Los casos más lentos violan las especificaciones ISO, por lo que se requiere software o hardware especializado. Además, la calidad de los pares de llaves generalmente no es muy alta debido a que las limitaciones de poder de cómputo provocan una fuente débil de números aleatorios e impiden la utilización de números primos de gran tamaño.

Las tarjetas inteligentes tienen la capacidad de almacenar y configurar varios PINs para propósitos diferentes. Uno de los PINs puede ser configurado por las aplicaciones para desbloquear el PIN del usuario después de ser bloqueado por sobrepasar el número de intentos permitidos o para reiniciar la tarjeta. Otros PINs pueden ser configurados para controlar el acceso a archivos o para su utilización en aplicaciones de monedero electrónico.

Pueden encontrarse DES y triple DES en las tarjetas más modernas. Generalmente tienen la opción de utilizar estos algoritmos para construir códigos MAC (Message Authentication Code), sin embargo, como la interfaz de comunicación serial de las tarjetas Inteligentes actuales tienen un ancho de banda lento, realizar operaciones de cifrado simétrico puede llegar a ser muy lento.

Aunque extraer información acerca del procesador o el sistema de archivos de una tarjeta inteligente de por sí es difícil por el esfuerzo que requiere, las tarjetas inteligentes más actuales cuentan además con varios métodos físicos de monitoreo y seguridad. Uno de los más comunes es un fusible que deshabilita la capacidad de ejecutar cualquier código cargado en la EEPROM. Una vez que el fusible ha sido deshabilitado, la tarjeta no puede regresar a su estado anterior. Para evitar la clonación de tarjetas, un número de serie inalterable es grabado en ROM. Además, las tarjetas es-

tán diseñadas para reiniciarse si detectan alguna fluctuación en el voltaje, la temperatura o la frecuencia de la señal de reloj. Generalmente la escritura y lectura de la memoria ROM no se encuentra disponible. Aunque las medidas mencionadas son las más comunes, éstas pueden variar según el fabricante.

La generación de números aleatorios varía según el fabricante. Algunos implementan números seudo-aleatorios al incluir en cada una de las tarjetas una semilla diferente. Para estos casos los números generados se encuentrarı dentro de un rango que depende de la semilla y el algoritmo utilizado. Algunas tarjetas implementan en hardware un generador real de números aleatorios que utiliza alguna propiedad física del silicio del que está construido el procesador.

Los protocolos de comunicación en las tarjetas inteligentes contienen la mayoría de la veces un protocolo de seguridad implementado al nivel de comandos que generalmente utilizan cifrado simétrico y permiten que la tarjeta inteligente valide el dispositivo de lectura (lector) o viceversa. Sin embargo, los criptogramas y algoritmos para estos protocolos de seguridad generalmente cambian con la aplicación y la terminal.

3.1.4.4. Comunicación con la Tarjeta

Toda la comunicación desde y hacia la tarjeta inteligente se lleva cabo por el contacto C7 (figura #) de manera half-duplex. Esto significa que sólo puede existir comunicación en un sentido a la vez, ya sea de parte de la tarjeta o de parte de la terminal. La comunicación siempre es iniciada por la terminal, lo que significa que la interacción entre la tarjeta y la terminal es del tipo cliente-servidor.

Después de que la tarjeta es insertada en la terminal, ésta le proporciona la corriente y el voltaje para funcionar. La tarjeta entonces se reinicia (power-on-reset), y en vía un ATR (Answer To Reset) a la terminal [GemplusSCBasics]. La terminal realiza un análisis sintáctico del ATR, obtiene varios parámetros y envía a la tarjeta una instrucción inicial. La tarjeta genera entonces una respuesta y la envía a la terminal. La interacción cliente-servidor continúa de la misma manera hasta que los procesos terminan y la tarjeta es removida de la terminal.

La capa física para el proceso de transmisión está especificado en el estándar ISO / IEC 7916-3 [ISO7816 part[1-3]]. En ese estándar se encuentran definidos los niveles de voltaje para los cuales la señal es interpretada como un 1 o como un 0 lógicos.

Existen varios protocolos diferentes para el intercambio de información entre la tarjeta y la terminal. Estos protocolos están identificados mediante los caracteres "T=" más un número, como se muestra en la tabla siguiente:

Tabla 3.4. Protocolos de comunicación entre ICC y CCID

Protocolo	Descripción
T0	Comunicación asíncrona, half-duplex, orientada a bytes
T1	Comunicación asíncrona, half-duplex, orientada a bloques
T2	Comunicación asíncrona, full-duplex, orientada a bloques
T3	Full-duplex
T4	Comunicación asíncrona, half-duplex, orientada a bytes (expansión del protocolo T = 0)
T5 - T13	Reservados para su uso en el futuro
T14	No es un estándar ISO.
T15	Reservado para su uso en el futuro

Los dos protocolos más comunes actualmente son T=0 y T=1, de los cuales el más utilizado es T=0.

3.1.4.5. Conjuntos de instrucciones

Existen cuatro estándares internacionales que definen los conjunto de instrucciones más comunes para tarjetas inteligentes. En esos estándares se encuentran definidas más de cincuenta instrucciones con los parámetros correspondientes. Aunque se encuentran definidas en cuatro estándares diferentes, todas las instrucciones son compatibles entre si, lo que significa que en una misma tarjeta pueden implementarse instrucciones que pertenecen a más de uno de ellos. Los cuatro estándares mencionados son el GSM 11.11 (prETS 300608), EN 726-3, ISO / IEC 7816 - 4, y CEN (psEN 1546).

A grandes rasgos, todas las instrucciones pueden clasificarse por su funcionalidad en una de las siguientes categorías:

- Selección de archivos

- Lectura y escritura de archivos
- Búsqueda de archivos
- Administración de archivos
- Operaciones con archivos
- Identificación
- Autentificación
- Funciones criptográficas
- Instrucciones relacionadas con aplicaciones de monedero electrónico o para tarjetas de crédito
- Complementos del sistema operativo
- Pruebas de hardware
- Funciones de soporte para protocolos de transmisión
- Instrucciones especiales para aplicaciones específicas

3.1.4.6. Lectores de tarjetas inteligentes

Aunque comúnmente son conocidos como "lectores", todas las terminales construidas para ser utilizadas con tarjetas inteligentes tienen la capacidad de leer y escribir en la tarjeta; siempre y cuando la tarjeta esté habilitada para realizar estas operaciones y se hayan cumplido las condiciones de seguridad necesarias [OpenSC].

A diferencia de las tarjetas inteligentes que son similares en su estructura, las terminales pueden tener una gran variedad de formas y niveles de sofisticación física y lógica. Como ejemplo podemos mencionar los lectores incluidos en PDAs, en teléfonos celulares GSM y en computadoras personales. Además, los lectores ofrecen varias opciones mecánicas entre las que se incluyen que el usuario inserte y remueva la tarjeta por sí mismo o que algún mecanismo lo haga por él automáticamente, contactos eléctricos de deslizamiento o de superposición y diversas maneras para realizar una comunicación entre el usuario y las aplicaciones de la tarjeta (teclados, pantallas LCD, etc). Sin embargo, todos los lectores deben cumplir con las especificaciones eléctricas incluidas en el estándar ISO/IEC 7816 - 3.

3.1.4.7. Problemas de seguridad

Una de las características más importantes (o tal ves la más importante) de las tarjetas inteligentes es la capacidad de aumentar el grado de seguridad en sistemas de cómputo al proporcionar un método más confiable de autentificación [STRUCK][SauveronWeb].

Sin embargo, aún con las medidas de seguridad incluidas en las tarjetas inteligentes, es posible que personas no autorizadas tengan acceso a la información contenida en ellas.

Como es bien sabido, ningún sistema es cien por ciento seguro, y el caso de las tarjetas inteligentes no es la excepción. La importancia de las tarjetas inteligentes no radica en que sean inviolables, si no en que aumentan en un gran medida el esfuerzo y tecnología requeridos para obtener la información que se guarda en ellas. Todo diseñador de un sistema de seguridad toma en cuenta la relación entre el valor de la información y el esfuerzo que se requiere para romper las barreras de seguridad establecidas, y en la mayoría de los casos, las tarjetas inteligentes establecen una barrera lo suficientemente fuerte como para hacer desistir a la mayoría de los atacantes.

Los ataques a tarjetas inteligentes generalmente caen en alguna de las siguiente categorías:

- *Ataques lógicos* Ocurre cuando la tarjeta se encuentra trabajando bajo condiciones físicas normales. La información se obtiene al examinar los bytes que entran y salen de la tarjeta. Un ejemplo es el ataque conocido como "timing attack" [DPA]. En este ataque se envían varios patrones de bytes a la tarjeta para que ésta los cifre utilizando su llave privada. Mediante información como el tiempo que tarda la tarjeta en realizar el cifrado y el número de ceros y unos obtenidos, eventualmente se obtiene la llave privada de la tarjeta.

  Aunque ya existen medidas para prevenir este tipo de ataques, actualmente no todas las tarjetas las implementan

  Este ataque requiere el conocimiento del PIN de la tarjeta para poder realizar varias peticiones de cifrado.

- *Ataques físicos* Ocurren cuando las condiciones físicas normales, como la temperatura, la frecuencia de la señal de reloj, el voltaje, etc son alteradas para tener acceso a la información contenida en la tarjeta inteligente [ANDERSON-KHUN-1997]. La mayoría de los sistemas operativos almacenan la información importante de forma cifrada en la EEPROM, de manera que si se obtiene información al atacar directamente la memoria, ésta no sea útil [KÖMMERLING-KHUN-1999].

  Otro tipo de ataques físicos exitosos son en los que se lleva a cabo una fluctuación muy grande en las condiciones físicas de funcionamiento de la tarjeta justo en el momento en que la tarjeta realiza la verificación del PIN del usuario. Al hacer esto se han podido ejecutar funciones sobre la información de la tarjeta sin conocer el PIN correcto. Este ataque puede combinarse con ataques lógicos para obtener la llave privada de la tarjeta. Para realizar este ataque se requiere de equipo especial.

- *Troyanos* Este ataque se lleva a cabo mediante el uso de un programa troyano instalado en la estación de trabajo en la que se encuentra conectado el lector. El programa espera a que el usuario teclee el PIN en una aplicación confiable para él. Cuando esto sucede, y ya que el usuario de ha autentificado, el programa envía información para ser cifrado por la tarjeta sin que el usuario se entere de ello.

- *Ingeniería Social* En seguridad de sistemas de cómputo, este tipo de ataques es usualmente el más exitoso, se presenta especialmente cuando la seguridad tecnológica esta propiamente implementada y configurada. Usualmente estos ataques recaen en los errores cometidos por las personas. Un ejemplo de ataque de ingeniería social cuando un hacker suplanta la identidad del servicio técnico de red, aprovechando la confianza de empleados de menor nivel y solicita sus contraseñas con el supuesto propósito de un mantenimiento de red. Con las tarjetas inteligentes este tipo de ataques es mas difícil de realizar. La mayoría de la gente no confiaría en ninguna persona que deseara tener su tarjeta inteligente y el PIN con propósitos de otorgar un servicio.

Una solución a este problema es que el sistema operativo restrinja la utilización del lector, de manera que sólo pueda ser utilizado por una sola aplicación al mismo tiempo; sin embargo, mediante este método también se le quita funcionalidad a la tarjeta, ya que varias aplicaciones no podrían utilizar los servicios de la tarjeta al mismo tiempo. Otra solución es pedir el PIN del usuario cada vez que la tarjeta realice una operación de cifrado, de modo que un programa troyano no pueda utilizar la tarjeta sin que el usuario se dé cuenta de ello.

3.1.5. Servicios Proporcionados por Tarjetas Inteligentes

La tarjeta inteligente es una de las ultimas adiciones en el mundo de las tecnologías de información. Son del tamaño de una tarjeta de crédito, tiene un microprocesador embebido en ella, cuando se complementa con un lector obtiene el poder de procesamiento para utilizarse por diferentes aplicaciones.

Actualmente existen múltiples aplicaciones que nos ayudan a realizar nuestras labores cotidianas de una forma mucho más sencilla, con la introducción gradual del uso de tarjetas inteligentes se pretende proporcionar estos servicios utilizando una misma tarjeta. Además se pretende que estos servicios sean personalizados de acuerdo al portador de la tarjeta [SCBasics].

Inicialmente la tarjeta inteligente fue concebida como el reemplazo de la tarjeta de crédito de cinta magnética, debido a sus características mejoradas en seguridad física y procesamiento de información. Bajo este concepto los servicios proporcionados fueron enfocados hacía esquemas donde se realizaran transacciones monetarias.

Gracias a que la tarjeta inteligente cuenta con memoria y capacidad de procesamiento ofrece muchas más aplicaciones que las tarjetas de crédito tradicionales de banda magnética y puede proporcionar múltiples aplicaciones de diferente tipo en una misma tarjeta.

A medida que las tarjetas inteligentes han aumentado su capacidad de memoria e instrucciones de procesador, han ido aumentando la cantidad de servicios que pueden ofrecer, estos servicios están típicamente enfocados a cubrir tres aspectos:

- **Financieros:** La tarjeta puede ser utilizada para el manejo de transacciones como por ejemplo el reemplazo de dinero en efectivo.
- **Identificación:** La tarjeta proporciona una forma segura de identificar al poseedor, para permitirle el acceso a zonas restringidas (p.e. Autorización de acceso a computadoras).
- **Soporte:** La tarjeta es utilizada como un medio portátil y proporciona una forma confiable para almacenar de información (e.j. Registros Médicos).

Cada uno de estos aspectos se enfocan a resolver actividades muy particulares, por lo que el nivel de detalle al que pueden llegar dependerá de la capacidad de diseñar una aplicación que cubra una determinada demanda. En la Figura 3.2, "Algunas de las aplicaciones para tarjetas inteligentes " se muestran algunas de las múltiples aplicaciones que se pueden realizar con tarjetas inteligentes.

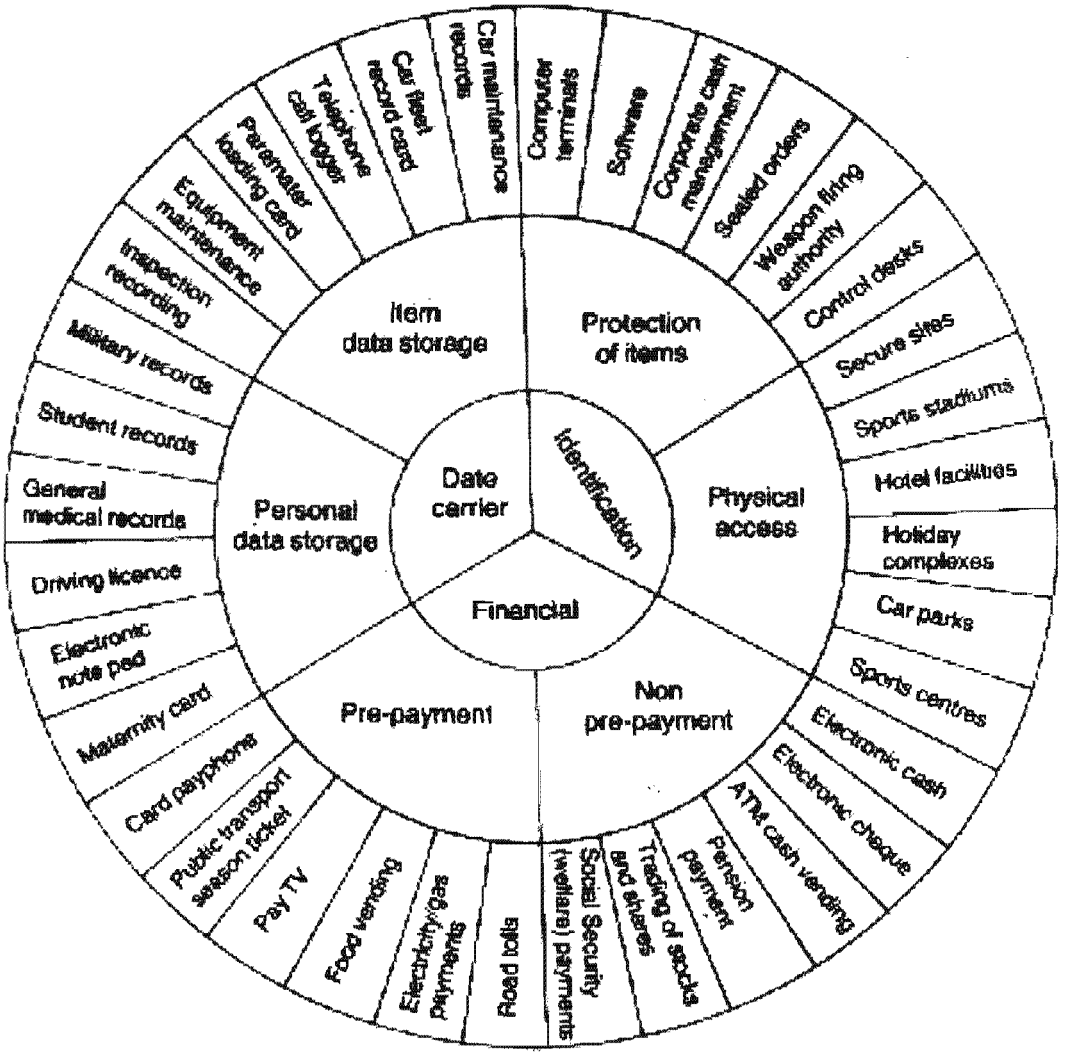

Figura 3.2. Algunas de las aplicaciones para tarjetas inteligentes [JAVACARD-GROUP]

La finalidad de utilizar una tarjeta inteligente es poder ofrecer al usuario un mecanismo sencillo para coleccionar diferentes servicios en un dispositivo único y no encontrarse en la necesidad de utilizar diversos medios para hacer uso de los servicios.

A medida que la tecnología se abarate y las tarjetas inteligentes sigan evolucionando, se lograra agregar mayores y mejores servicios, con la facilidad de poder disponer de ellos en cualquier lugar y con una sola tarjeta. Todos los servicios estarán automatizados y todo lo que se refiera a información personal se encontrara dentro de la tarjeta.

3.1.5.1. Posibles aplicaciones

- **Navegadores de Internet** Los Navegadores de Internet utilizan tecnologías como SSL (Secure Sockets Layer) y TLS (Transport Layer Security) para proporcionar seguridad mientras se está navegando por Internet. Mediante estas tecnologías se puede hacer una autentificación entre el cliente y el servidor, y proporcionar un canal cifrado para el intercambio de información. Con la utilización de una tarjeta inteligente la seguridad es incrementada debido a que la llave privada del usuario se encuentra almacenada de forma segura dentro ella.

- **Correo electrónico seguro** Mediante S/MIME (Secure Multi-Purpose Internet Mail Extensions) y OpenPGP (Open Pretty Good Privacy) el correo electrónico puede ser enviado ya sea solamente cifrado o firmado digitalmente. Las operaciones de cifrado pueden realizarse entonces con la llave privada almacenada en la tarjeta inteligente [PIETING-2003].

- **Firma digital de formas HTML** El firmar digitalmente una forma HTML permite que las aplicaciones de negocios basadas en Web, puedan tener todos sus documentos almacenados en un servidor, de manera que cualquier usuario tenga acceso a ellos a través Internet. Las tarjetas inteligentes proporcionan una mayor seguridad mediante el almacenamiento y portabilidad de la llave privada.

- **Firma de objetos** Si una organización produce código que puede ser descargado a través de Internet y ejecutado en la computadora del cliente, la firma digital del código le daría al cliente la seguridad de que el archivo descargado es seguro, y que de hecho pertenece a la organización.

- **Kioscos** Mediante la utilización de tarjetas inteligentes, las aplicaciones utilizadas en kioscos pueden guardar las preferencias del usuario en la memoria de la tarjeta, de manera que el cliente lleve su configuración personal con él.

- **Cifrado de archivos** Aunque generalmente no se utilizan las capacidades criptográficas de las tarjetas inteligentes para realizar el cifrado completo de un archivo, debido a que el poco poder de computo de las tarjetas actuales y la poca rapidez de transferencia entre la tarjeta y la terminal hacen que este proceso no sea lo su-

ficientemente rápido, éstas pueden ser utilizadas para mejorar este proceso [PETRI].

Si se genera una llave aleatoria por sesión, ésta puede ser utilizada por una computadora más rápida (como una PC) para realizar el proceso de cifrado. Después, la llave utilizada puede cifrarse con la llave privada contenida en la tarjeta, de manera que sólo la persona que tiene la tarjeta inteligente puede descifrar la llave de sesión, y por lo tanto el archivo.

- **Conexión a estaciones de trabajo** Una tarjeta inteligente es un medio perfecto para guardar nombres de usuario y contraseñas utilizadas para tener acceso a una estación de trabajo. Debido a que la información utilizada para la autentificación se encuentra contenida en la tarjeta, no es necesario que el usuario la conozca, lo que fomenta la generación de contraseñas más seguras, que pueden ser asignadas por el mismo sistema.

- **Acceso a estaciones de trabajo** Las credenciales de acceso pueden ser almacenadas de forma segura en una tarjeta inteligente. Los mecanismos tradicionales de acceso a las estaciones de trabajo, que usualmente preguntan por un nombre de usuario y contraseña, pueden ser remplazados con uno que se comunique directamente con la tarjeta.

- **Monedero digital** Las tarjetas inteligentes pueden ser utilizadas para implementar aplicaciones de monederos digitales. En estas aplicaciones, las llaves utilizadas para autentificación son intercambiadas exclusivamente por el hardware, de manera que los usuarios no las conocen.

- **Acceso en áreas restringidas** Las tarjetas inteligentes proporcionan grandes ventajas en cuanto a seguridad física se refiere, ya que, a diferencia de las cintas magnéticas, por ejemplo, una tarjeta inteligente tiene en sí misma métodos de seguridad que dificultan su falsificación. En caso de que la inserción de tarjetas en una terminal pueda llegar a ser poco práctica, pueden utilizarse tarjetas sin contacto en lugar de las tarjetas convencionales.

- **Compatibilidad con PKI** Puede ser utilizada como portadora de certificados digitales bajo el estándar X.509, permitiendo la autentificación en una variedad muy grande de redes de comunicación corporativas o dedicadas a la seguridad de la información. Las tarjetas inteligentes permiten el uso de estándares especiales como el PKSC#11 y PKSC#15, utilizados ampliamente en aplicaciones PKI [RSALabs-PKCS#11][RSALabs-PKCS#15].

3.1.5.2. Beneficios de las tarjetas inteligentes

Una de las principales ventajas de las tarjetas inteligentes, es la capacidad de cubrir totalmente los siguientes conceptos de seguridad:

- **Autentificación.** (tarjeta y terminal están seguros de la identidad del otro). Se realiza, gracias a la existencia de procesadores criptográficos simétricos (3DES) y/o asimétricos (PKI).
- **Identificación.** (Verificación de la identidad de la tarjeta). La validación del PIN se realiza en la propia tarjeta.
- **Integridad.** (Asegurar que el mensaje no ha sido alterado). Utilizando algoritmos simétricos de criptografía (hashing).
- **No repudio.** (Evitar la denegación de una transacción). Existiendo la posibilidad de realización de firmas con la llave privada.

3.1.6. Integración de los sistemas basados en tarjetas inteligentes

Cualquier estándar diseñado para facilitar la integración de tarjetas inteligentes debe cumplir con ciertos principios para que sea útil y llegue a ser aceptado. Algunos de éstos principios son:

- **Multiplataforma** El estándar debe ser aplicable a los sistemas operativos y arquitecturas más utilizadas en el momento de su creación (p.e. Windows, Unix, Mac, x86,Sparc, etc.)
- **Participación abierta** El estándar debe aceptar contribuciones y revisiones de miembros de la industria, del gobierno y la educación.
- **Interoperabilidad** El estándar debe funcionar en conjunto con otros estándares y protocolos.
- **Real y funcional** El estándar debe estar enfocado a resolver problemas reales y cumplir adecuadamente con sus requerimientos.
- **Experiencia y productos** El estándar debe ser creado por un grupo de personas con experiencia en el desarrollo de productos relacionados con seguridad y en la creación de estándares anteriores.

- **Extensibilidad** El estándar debe facilitar su expansión a nuevas aplicaciones, protocolos y aumentar las capacidades de tarjetas inteligentes que no existían en el momento de su creación.

3.1.6.1. Estándares actuales más importantes

3.1.6.1.1. PKCS#11: Cryptografic Token Interface Standard

Este estándar incluye la especificación de una API, llamada Cryptoki (Cryptographic Token Interface), para dispositivos que almacenan información criptográfica y realizan operaciones de cifrado [RSASecurity][RSALabs-PKCS#11]. Cryptoki implementa una solución orientada a objetos, logrando así una interfaz independiente del dispositivo mediante la cual se pueden desarrollar aplicaciones que compartan recursos entre si, es decir, que pueden existir varias aplicaciones accediendo a varios dispositivos diferentes. PKSC#11 fue creado en 1994 por RSA con contribuciones de miembros de la industria, del gobierno y la educación.

3.1.6.1.2. PC/SC: Personal Computer / Smart Card

La organización PC/SC (PC/SC Workgroup) fue creado en Mayo de 1997 con los siguientes objetivos:

- Promover una especificación estándar que asegure que las tarjetas inteligentes, los lectores de tarjetas y las computadoras personales desarrolladas por distintos fabricantes puedan trabajar en conjunto.
- Facilitar el desarrollo de aplicaciones con tarjetas inteligentes para computadoras personales y otras plataformas de cómputo.

Los miembros principales son Gemplus, Infineon Technologies, Ingenico, Microsoft, Philips, Schulmberger y Toshiba.

La especificación PC/SC constituye una capa sobre los estándares ISO 7816 y EMV, además de complementarlos al definir interfaces de bajo nivel y una serie de APIs independientes del hardware, así como una especificación para la administración de recursos de manera que varias aplicaciones puedan compartir las tarjetas inteligentes conectadas a un sistema [PCSC-WORKGROUP]. La segunda versión de la especificación PC/SC fue lanzada en Junio del 2004.

3.1.6.1.3. OpenCard Framework

OpenCard fue creado por un conjunto de compañías como Gemplus, Siemens, Sun Microsystems y Toshiba Corporation entre otras, con el objetivo de proporcionar una manera de desarrollar aplicaciones para tarjetas inteligentes que pudieran ser utilizadas en distintas plataformas.

OpenCard Framework es un estándar que proporciona una arquitectura y un conjunto de APIs que permiten a los desarrolladores de aplicaciones y a los proveedores de servicios desarrollar soluciones basadas en tarjetas inteligentes que puedan utilizarse en cualquier ambiente que cumpla con las especificaciones que establece la arquitectura OpenCard [OCF].

3.1.6.1.4. Java Card

Java Card es una tecnología que permite a las tarjetas inteligentes y a otros dispositivos con memoria limitada ejecutar aplicaciones (llamadas applets) que utilizan tecnología Java [SUN-JAVACARD-INFO]. Proporciona a los fabricantes de tarjetas inteligentes una plataforma de ejecución segura e interoperable capaz de almacenar y actualizar varias aplicaciones en un mismo dispositivo. La tecnología Java Card es compatible con los estándares actuales relacionados con las tarjetas inteligentes.

Java Card API define la manera en que un applet tiene acceso al Java Card Runtime Enviroment (Java Card RE). Además permite que las aplicaciones desarrolladas para un dispositivo compatible con Java Card puedan ser ejecutado en cualquier otro dispositivos compatible con Java Card.

3.1.6.1.5. Common Data Security Architecture

Desarrollado por Intel, el Common Data Security Architecture (CDSA) proporciona un framework de características abierta, interoperable, extensible y de plataforma cruzada que permite a las diferentes plataformas de cómputo incrementar la seguridad en todas las aplicaciones incluyendo el comercio electrónico, comunicaciones y contenido digital. La especificación CDSA 2.0 fue adoptada por The Open Group en diciembre de 1997.

3.1.6.1.6. Microsoft Cryptographic API

Microsoft Cryptographic API (CryptoAPI) permite agregar funciones de cifrado y ma-

nejo de certificados (firma digital) en aplicaciones hechas para Win32 [MICROSOFT-TECHNET]. Las aplicaciones pueden utilizar funciones del CryptoAPI sin tener que saber absolutamente nada acerca de las capas más bajas de implementación, casi de la misma forma en que una aplicación puede utilizar una librería gráfica sin tener que conocer nada acerca la configuración de la tarjeta gráfica.

3.1.7. Tarjetas inteligentes multiaplicación

La mayoría de los sistemas que hacen uso de tarjetas inteligentes cumplen con un solo propósito y están relacionados a un solo proceso [SCBasics][JAVACARD-GROUP]. En este caso se encuentran las tarjetas telefónicas públicas, las tarjetas medicas que almacenan el historial medico. Estas aplicaciones se guardan en diferentes tarjetas inteligentes y conducen a la misma situación y problema que con el sistema tradicional de tarjetas de banda magnética que requiere que el usuario cuente con una tarjeta para cada aplicación.

De hecho, las tarjetas inteligentes tienen la capacidad de integrar todas esas aplicaciones juntas para formar una tarjeta multiaplicación utilizando el procesador y los espacios de almacenamiento de memoria. Sin embargo este tipo de integración siempre esta limitado más por algunos elementos logisticos que por las capacidades técnicas. Por ejemplo, en una aplicación sencilla, los datos almacenados siempre pertenecen al emisor de la tarjeta. En el caso de más de una aplicación residiendo en una sola tarjeta, esto se vuelve impráctico.

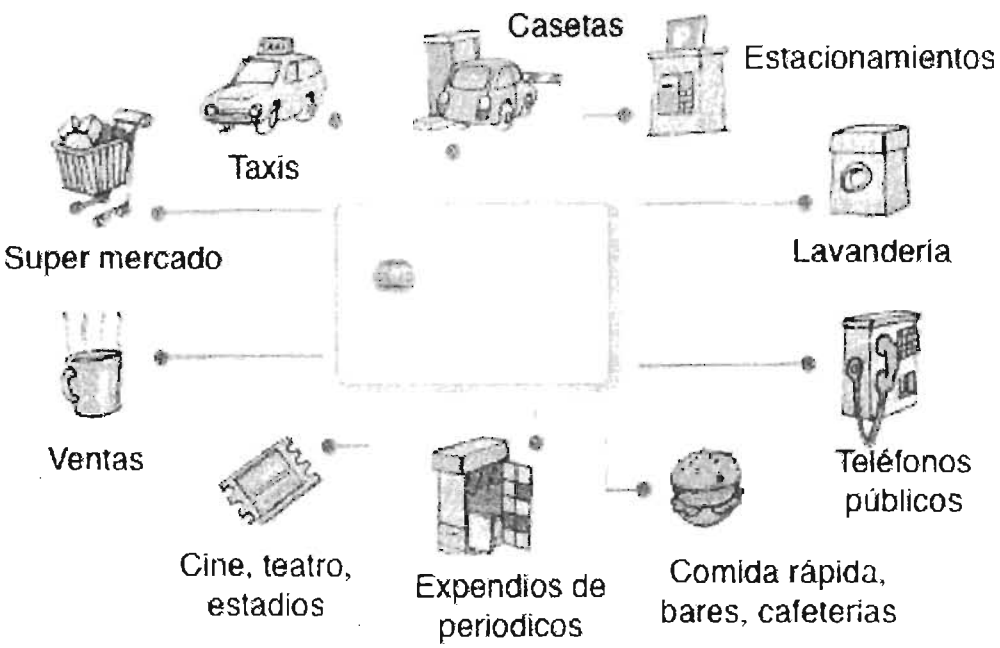

Figura 3.3. Tarjeta multiplataforma

Hasta el momento, el estándar 7816 esta incompleto en todos su niveles. En el nivel de tarjeta no se tienen especificado completamente una interfase con el sistema operativo, no se tiene propiamente especificado una estructura de directorio y las estructuras de datos no están completamente especificadas. A nivel Aplicación, los estándares están incompletos y en muchos casos no han sido iniciados.

3.1.7.1. Sistema operativo genérico

Para lograr la integración de todas las aplicaciones dentro de una tarjeta, los fabricantes buscan desarrollar un sistema operativo genérico y utilizando Java, a pesar de la falta de estandarización. Muchas compañías ya han comenzado a generar sus tarjetas basadas en la especificación Java Card.

Las tarjetas inteligentes genéricas son como computadoras. No tienen ninguna función orientada a aplicación en sus funcionalidades básicas. En consecuencia, el programa de la tarjeta es un sistema operativo real el cual administra los recursos de hardware para las aplicaciones. En cuanto a las aplicaciones, estas no están predefinidas, pueden ser dinámicamente cargadas. El sistema operativo de la tarjeta reserva la memoria para colocar los datos de las aplicaciones y activar las funciones en co-

mandos de recepción. Por razones inherentes a la seguridad de las tarjetas, las funciones se ejecutan en una *máquina virtual* por un interprete seguro en lugar de hacerlo directamente en el lenguaje nativo. Por supuesto, otras cuestiones de seguridad como el intercambio de archivos a través de aplicaciones deben ser tratadas. Estas cuestiones son integradas en la especificación del Java Card.

3.1.8. Uso de tarjetas inteligentes en los negocios

Una encuesta realizada por *Card Technology* Magazine [CardTechnology] indica que la industria ha gastado más de 1.5 billones en tarjetas inteligentes alrededor del mundo desde finales de 1999.

Dentro de los siguientes cinco años, la experiencia de la industria crecerá, particularmente en tarjetas y dispositivos que se encaminen al comercio electrónico y habilitar el acceso seguro a las computadoras en red [EuroSmart].

En el año 2000 se predijo que se comprarían al menos 20 millones de tarjetas inteligentes (microprocesadores y memoria) en los EEUU [DataQuest], de acuerdo a este estudio, un crecimiento anual del 60% es esperado para la compra de tarjetas inteligentes entre 1998 y 2003.

3.2. Tarjetas java

Cada vez más las tarjetas inteligentes son una tecnología de uso común, sin embargo, esta tecnología apenas comienza a despegar y se prevé que su uso se extenderá aún más en un futuro cercano [Schlumberger-1996], debido principalmente a la demanda de aplicaciones en seguridad de la información y en el comercio electrónico. Las tarjetas inteligentes representan un avance cualitativo en el camino hacia una seguridad de la información práctica

Las tarjetas son dispositivos informáticos seguros, portátiles y resistentes a ataques físicos que permiten almacenar información importante y realizar los cómputos necesarios para producir firmas digitales y realizar otras operaciones criptográficas básicas.

Existen tarjetas programables que incorporan la tecnología Java Card, y hacen posible el desarrollo de aplicaciones específicas basadas en certificados digitales que no pueden ser soportadas por otras tecnologías o protocolos criptográficos. Gracias a

la seguridad física que ofrecen, así como al hecho de que sea el emisor de la tarjeta y no el poseedor de la misma quien controle el software que se ejecuta, este tipo de tarjetas permite el control sobre el uso de la información que transportan y posibilita el desarrollo de sistemas que ofrecen propiedades de seguridad difíciles de garantizar sin ellas.

A las tarjetas inteligentes capaces de poder ejecutar código java se les conoce como *Java Card*, la tecnología empleada no solo hace referencia a la tarjeta sino también a las herramientas utilizadas para el desarrollo de aplicaciones por medio del lenguaje Java. Java Card esta basado en el enfoque de que una sola tarjeta inteligente puede contener diferentes aplicaciones.

3.2.1. ¿Qué es la tecnología Java Card?

Java Card es un estándar abierto creado por Sun Microsystems que permite ejecutar a las Tarjetas Inteligentes y tokens criptográficos pequeñas aplicaciones, conocidas como Java Card Applets, que utilizan tecnología Java [SUN-JAVACARD-TECH]. El objetivo de la tecnología Java Card es llevar los beneficios del desarrollo de software en Java al mundo de las tarjetas inteligentes.

El uso de esta tecnología ha permitido acelerar la adopción de las tarjetas inteligentes multiaplicación. Sus ventajas son la flexibilidad inherente a un desarrollo en Java, la rapidez de desarrollo de applets y los mecanismos de seguridad que provee. Esta conformada por un grupo de aplicaciones para ejecutar un subconjunto del lenguaje Java sobre una tarjeta inteligente (Java Card).

Las tarjetas inteligentes que hayan sido creadas utilizando la plataforma Java Card contienen applets almacenados en ellas. Varios applets pueden convivir dentro de la misma tarjeta, y pueden ser cambiados o incluidos después de que la tarjeta ha sido personalizada.

De manera similar a la plataforma Java, la especificación de la plataforma Java Card incluye tres documentos que sirven de base para las implementaciones de tres de los cuatro componentes que forman parte de la tecnología Java Card:

- *Virtual Machine Specification for the Java Card Platform* La especificación de la Máquina Virtual para la plataforma Java Card (Java Card Virtual Machine, JCVM)

define las características, servicios y el comportamiento que debe seguir una implementación de la tecnología Java Card.

Incluye el conjunto de instrucciones de la JCVM, la descripción del subconjunto del lenguaje Java que implementa y los formatos de los archivos utilizados para la instalación de applets y librerías en los dispositivos que implementan la tecnología Java Card.

- *Runtime Environment Specification for the Java Card Platform* Esta especificación define el comportamiento necesario del ambiente de ejecución de cualquier implementación de la tecnología Java Card. El JCRE además proporciona servicios en tiempo de ejecución como la selección de applets.
- *API for the Java Card Platform* La API (Application Program Interface) para Java Card contiene la definición de las clases requeridas para utilizar la Java Card VM y el Java Card RE.

El último componente es el *Java Card Development Kit*, que es un conjunto de herramientas para diseñar nuevas implementaciones basadas en la tecnología Java Card y para desarrollar applets basados en la API de Java Card . El JCDK incluye lo siguiente:

- **C-JCRE (C - Java Card Runtime Environment)** Es una implementación de referencia del JCRE hecha en lenguaje C. C-JCRE incluye un interprete para la JCVM.
- **Off-card platform components** El JCDK proporciona herramientas para utilizarse fuera de la tarjeta, como el *Java Card Converter* y el *Java Card Verifier.*
- **Otros** Junto con el JCDK se incluyen herramientas adicionales que permiten a los desarrolladores probar sus prototipos.

Para poder comprender como funciona una Java Card, hay que tener en cuenta que al realizar la especificación de la plataforma, Sun se apegó al estándar ISO 7816, el cual establece, entre otras cosas, la forma de comunicación entre una tarjeta inteligente y un lector. De acuerdo al ISO 7816, el intercambio de información y comandos entre la tarjeta y el lector se realiza a través de APDUs (Application Protocol Data Units), los cuales son paquetes de información con un formato específico.

De acuerdo al estándar, las tarjetas inteligentes nunca inician la comunicación con el lector, sino que sólo responden a los comandos que éste le envía. Se define dos tipos de APDU, los llamados *COMMAND APDU*, que son los que envía el lector a la tarjeta, y los *RESPONSE APDU*, que son los que envía la tarjeta al lector como respuesta a un *COMMAND APDU*.

La siguiente figura se describe el formato de ambos tipos de APDU.

Command APDU						
Encabezado Obligatorio				Cuerpo Opcional		
CLA	INS	P1	P2	LC	Data field	LE

Response APDU		
Cuerpo Opcional	Cola Obligatoria	
Data field	SW1	SW2

Campo	Tamaño	Descripción
CLA	1 byte	Clase de instrucción. Indica la estructura y el formato para una categoría de COMMAND y RESPONSE APDU
INS	1 byte	Código de instrucción. Especifica la instrucción del comando
P1	1 byte	Parámetros de la instrucción. Proveen más información sobre la instrucción
P2	1 byte	
LC	1 byte	Número de bytes en el Data Field del APDU
Data Field	LC bytes	Secuencia de bytes con información
LE	1 byte	Cantidad máxima de bytes esperados como respuesta

Data Field	LE bytes	Secuencia de bytes con información
SW1	1 byte	Status Word (palabra de estado). Denotan el estado del procesamiento del comando en la tarjeta
SW2	1 byte	

Figura 3.4. Composición del APDU.

El propósito de la tecnología Java Card es ofrecer un interfaz de alto nivel inter operable gracias al cual las aplicaciones desarrolladas con él puedan ejecutarse en cualquier tarjeta compatible Java Card. Estas aplicaciones siguen la filosofía Java, *write once, run anywhere* [SUN-JAVACARD-INFO].

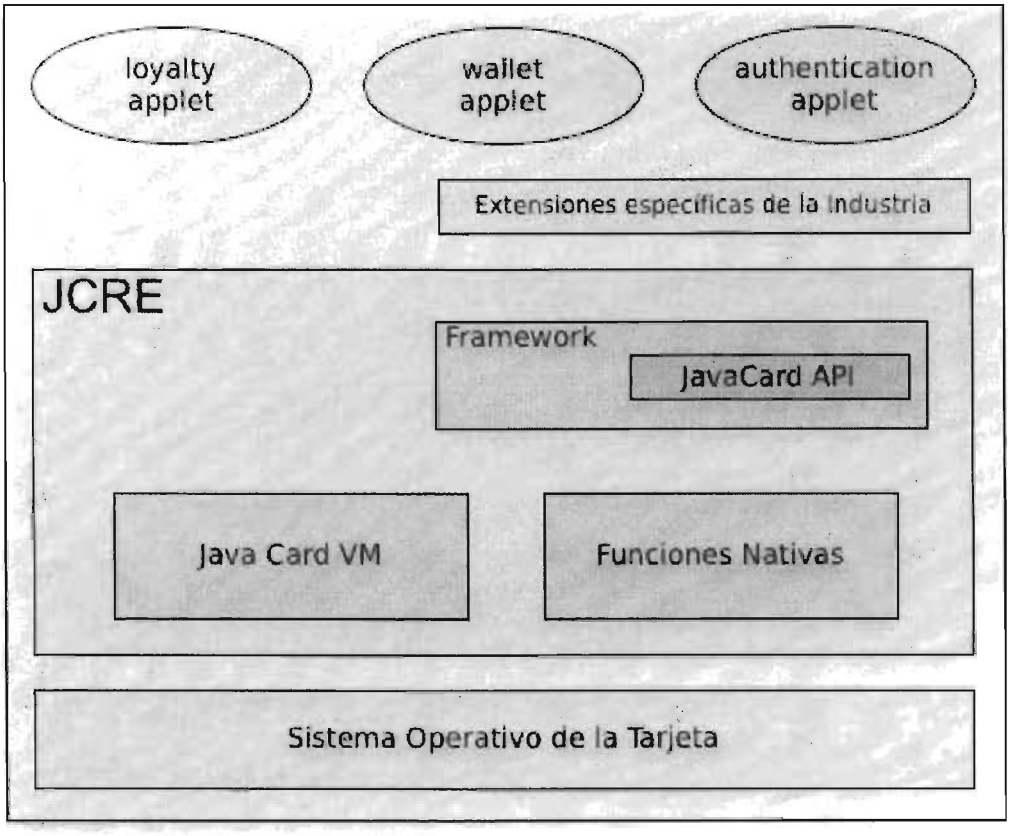

Figura 3.5. Arquitectura Java Card.

3.2.2. Aplicaciones Java Card

El número de aplicaciones para Java Card, y de tarjetas inteligentes en general, va en un aumento constante, y abarca áreas muy diversas como las que se vieron anteriormente. Algunos ejemplos específicos pueden ser los siguientes:

- Electronic Purse o Electronic Wallet (ePurse y eWallet): esta aplicación se utiliza como dinero electrónico.
- Transacciones Seguras: Ya sea a través de cajeros automáticos o de Internet, las tarjetas inteligentes proveen un nivel de seguridad muy superior al de las tarjetas magnéticas comunes o los sistemas basados en contraseñas o cookies, ya que es normal que incluyan un API de criptografía fuerte.

- Identificación Digital / Firma Digital: este tipo de aplicaciones se utiliza para validar la identidad del portador de la tarjeta, o para poder certificar el origen de ciertos datos. Normalmente se basan fuertemente en las primitivas criptográficas del API y/o las que están implementadas en hardware.
- Programas de Lealtad: Esta aplicación sirve a las empresas que ofrecen servicio preferencial a clientes frecuentes con el objeto de validar la identidad del cliente, y descentralizar la información.
- Sistemas de Prepago: Aplicados a servicios que pueden variar desde telefonía móvil hasta TV cable, pasando por acceso a sitios web o transporte público.
- Tarjetas Hospitalarias: Sistema de identificación de pacientes y almacenamiento de los principales datos de la historia clínica de los mismos en tarjetas inteligentes para agilizar la atención.
- Control de Acceso y de Asistencia. Sistema de identificación personal.

3.2.3. Beneficios de la tecnología Java Card

- Interoperabilidad. Los applets desarrollados haciendo uso de la tecnología Java Card funcionan en cualquier tarjeta inteligente basada en dicha tecnología, es decir, son independientes del fabricante de tarjetas y del hardware utilizado.
- Seguridad. La tecnología Java Card confía en la seguridad inherente del lenguaje Java para proporcionar un entorno de ejecución seguro.
- Capacidad para soportar múltiples aplicaciones. Existe la posibilidad de hacer coexistir de forma segura en una misma tarjeta inteligente múltiples aplicaciones.
- Dinamismo. Las tarjetas inteligentes basadas en la tecnología Java Card son actualizables. Se les puede añadir o eliminar funcionalidad, nuevas aplicaciones, de forma segura y dinámica, respondiendo así a las necesidades de cada momento.
- Compatibilidad con los estándares existentes. La API de las Java Card es compatible con los estándares de tarjetas inteligentes ISO-7816 o EMV.

3.3. Software Libre

3.3.1. Introducción

El término *Software Libre*, se refiere a software que una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente por cualquier persona. Frecuentemente este tipo de software se hace disponible en Internet de manera gratuita, aunque el software libre puede venderse también en búsqueda de ganancias económicas [FSF]. La definición de software libre se debe principalmente a la *Free Software Foundation* y al proyecto GNU.

3.3.2. Historia

En 1983, Richard M. Stallman, entonces investigador en el *Massachusetts Institute of Technology*, anunció el inicio del Proyecto GNU, un intento de crear un sistema operativo similar a UNIX, pero con algunas mejoras - y mucho más importante - con la característica de que GNU sería "libre" [5] . Después, en 1985, Stallman creó en los Estados Unidos la Free Software Foundation (FSF - Fundación de Software Libre) como una organización dedicada a trabajar por los derechos de los usuarios de computadoras a usar, estudiar, copiar, modificar y redistribuir programas de computadora. Si bien muchos de estos derechos fueron asumidos implícitamente por miles de desarrolladores de software antes de 1985 (particularmente en la década de 1970), la creación de la FSF constituyó el primer intento formal de establecer ciertas libertades de los usuarios sobre el software, ante la popularización en la década de 1980 de licencias de uso, que por el contrario, imponían cada vez más restricciones a los usuarios.

Stallman introdujo los conceptos de "software libre" y "copyleft" que tienen por objetivo dar libertad a los usuarios e impedir la posibilidad de que un tercero se apropie del software, respectivamente.

3.3.3. Filosofía

El movimiento del software libre basa sus acciones sobre la premisa de que la sociedad debe tener como uno de sus valores más altos la cooperación entre las personas (a cualquier nivel: individual, internacional, etc...). Esta cooperación debe incluir tam-

[5] Para ese momento, el concepto de software libre no estaba claramente definido aún. En éste documento la definición precisa será vista más adelante

bién al software en tanto que información. Los partidarios de la filosofía del software libre, consideran inmoral anteponer la búsqueda de beneficio monetario individual a la de cooperar con la comunidad existente. También consideran que restringir el uso de información es irracional. Esta filosofía puede entrar en conflicto con muchas aspectos del desarrollo de software, específicamente con cuestiones legales, éticas y económicas: ¿Quién remunerará el trabajo de los desarrolladores? Si la sociedad necesita software, *la propiedad* del software es necesaria para motivar a los desarrolladores a producir más. En la opinión de los partidarios del software libre, producir más software no significa necesariamente producir mejor software o ni siquiera el software que la sociedad necesita.

Se ha discutido mucho acerca de la validez de esta postura y de los argumentos que la sostienen, pero lo que es cierto es que al día de hoy el software libre se sitúa en la escena comercial como uno de los modelos de desarrollo más impacto han tenido en los últimos años.

3.3.4. El concepto

¿Qué es el software libre? ¿Cómo se puede hablar de *libertad* cuando se habla de software? La FSF define al "software libre" como aquél software que no tiene restricciones para que sus usuarios lo puedan ejecutar, copiar, distribuir, estudiar o cambiar. Más precisamente, hablar de "*libertad del software*" es en realidad hablar de cuatro libertades específicas **para sus usuarios**:

- La libertad de ejecutar o usar el software, con cualquier fin (**libertad 0**).
- La libertad de estudiar al software y a su funcionamiento, y de adaptarlo a necesidades específicas (**libertad 1**).
- La libertad de redistribuir copias de los programas con el propósito de ayudar al prójimo (**libertad 2**).
- La libertad de mejorar al software y de hacer disponibles las mejoras para todo el público, de manera que toda la comunidad se beneficie (**libertad 3**).

Es importante hacer notar que las libertades 1 y 3 tienen como requisito el acceso al código fuente de los programas que componen el software. Así pues, el software es llamado "libre" cuando sus usuarios tienen estas cuatro libertades [GNU].

La FSF considera además algunas otras condiciones para nombrar una pieza de software como "software libre". Así, la libertad 0 se refiere a la libertad de cualquier persona u organización para usar el software para cualquier propósito, en cualquier tipo de sistema computacional, sin requerir pedir permiso al desarrollador o a cualquier otra persona o entidad. Respecto a la libertad de redistribuir copias (libertad 2), la FSF considera que ésta debe incluir a los programas con modificaciones o sin ellas y tanto la forma ejecutable como la forma de código fuente. Además, todas estas libertades deben ser irrevocables. Si el desarrollador del software tiene el poder de revocar estos permisos sin que el usuario le haya dado razón, el software no es libre. Algunas restricciones sobre la forma en que se distribuye el software son aceptadas dentro de la definición, mientras éstas no entren en conflicto con las libertades centrales.

Una de las características del software libre sobre la cual se enfatiza mucho es su capacidad de poseer un valor comercial. Debido a que la palabra inglesa "*free*" es usada tanto para decir 'libre' como para decir 'gratis', el término "*free software*" se presta a cierta confusión. Software Libre no significa software no-comercial. Siempre será posible, bajo la definición de la FSF, cobrar por una copia del software, sin importar cómo la hayamos obtenido. Además, el uso o desarrollo con fines comerciales es perfectamente válido.

En la práctica, todas estas libertades son protegidas por las leyes de derecho de autor (*copyright*), y a través de licencias de uso de software. Existen varias de éstas licencias que la FSF considera como licencias que logran hacer al software libre. Una de ésas licencias es la Licencia Pública General (GPL) de la FSF. Una copia de esta licencia puede ser encontrada en el Anexo F. Esta licencia implementa además, el concepto de *copyleft*. Copyleft un método por el cual un programa de computadora (u otra forma de obra intelectual) se vuelve libre, y además exige a toda persona que modifique el software, a volverlo libre también. Esto más que interferir con las libertades centrales, las protege. Es importante hacer notar que no todo el software libre implementa el concepto de copyleft, por ejemplo, el software de dominio público de fuente abierta es libre pues cumple con las 4 libertades de la definición, pero versiones modificadas de él podrían redistribuirse sin las mismas libertades.

3.3.5. Relevancia

3.3.5.1. En la industria del software

¿Por qué es importante el software libre? Primeramente, aquellos individuos que con-

cordamos con su filosofía lo consideramos como un esfuerzo ético importante para lograr una sociedad más próspera.

Desde el punto de vista de la ingeniería de software, el software libre tiene una consecuencia muy importante que es el modelo de distribución de fuente abierta u *Open Source*. De manera general, este término se refiere a cualquier programa cuyo código fuente está disponible para su uso o modificación por otros desarrolladores o usuarios. Éste modelo de distribución ha probado promover de gran manera el desarrollo de software de muy buena calidad, e incluso se le ha nombrado como uno de las tendencias más importantes en cuanto a desarrollo de software y su comercialización [WALL-2000]. En la última década, compañías de gran importancia en el negocio del software, tales como IBM, Apple, HP, Sun, SGI, Sharp, Novell, Google o Red Hat han incursionado exitosamente en desarrollos con software libre, mostrando que muchas de las ideas de los partidarios del software libre son posibles.

Como ejemplo de ésta tendencia, podemos mencionar que el servidor web más utilizado y que más crecimiento ha tenido al momento de escribir este reporte, es *Apache*, un importante proyecto de software libre. Este servidor web libre hace funcionar a más del 50% de las páginas web en el mundo, según estadísticas de *Netcraft* [6].

Además "Open Source" es una certificación que otorga la organización Open Source Initiative (OSI) a programas que cumplan con ciertas características muy similares a las del software libre y el copyleft de la FSF. La diferencia más importante entre la OSI y la FSF es filosófica, ya que para la OSI, el software debe tener ciertas libertades por cuestiones de calidad y mientras que para la FSF las razones son éticas.

3.3.5.2. En el presente proyecto

En cuanto a la importancia del software libre en el presente proyecto, consideramos como una gran ventaja la capacidad de usar el software, estudiarlo y redistribuirlo libremente. Hasta ahora hemos usado ampliamente piezas de software que habrían sido imposibles de costear si no es por la licencia de uso que nos otorga amplias libertades como usuarios. Estas libertades nos permiten adaptar tecnología existente a nuestras necesidades. Debido a que nuestro caso de estudio se trata de una institu-

[6] Netcraft es una compañía de servicios de Internet, localizada en Bath, Inglaterra. Es una autoridad en cuanto a la obtención y el análisis de datos acerca de Internet

ción académica con una comunidad de alrededor de 10,000 estudiantes, sería altamente costoso hacer uso de *software propietario**, que otorgara una licencia de uso restringida. Creemos que al usar software libre a nivel de plataforma de desarrollo, como sistema operativo y como herramienta de aplicación, el prototipo creado y el modelo propuesto de identificación de alumnos serán considerablemente más baratos que una solución de software propietario.

3.4. Características del software disponible para el desarrollo de aplicaciones

3.4.1. Panorama general

Java es un lenguaje de programación desarrollado inicialmente para la comunicación entre dispositivos electrónicos de consumo (refrigeradores, microondas, lavadoras), permitiendo una programación en lenguaje de alto nivel, independientemente de la plataforma; solo es necesario contar con una máquina virtual que interprete los bytecodes generados por la compilación de Java.

La tecnología Java ha sido implementada exitosamente en las tarjetas inteligentes, desde dos perspectivas diferentes, que han proporcionado buenos resultados y han permitido el desarrollo de aplicaciones bajo los conceptos multitarjeta y multiaplicación.

El primer enfoque permite ver a las tarjetas inteligentes como inicialmente se concibió a los dispositivos electrónicos desde la perspectiva java, es decir que existen muchos fabricantes de tarjetas inteligentes y cada circuito es diferente. De tal forma que cada fabricante proporciona un conjunto de clases que no solo permita comunicarse con la tarjeta inteligente, sino además explotar las funcionalidades implementadas en esa tarjeta en particular.

Con este enfoque se dejan los detalles de bajo nivel al fabricante; el desarrollador no tiene que lidiar con detalles específicos de la tarjeta y puede enfocarse en las funcionalidades que se le proporcionan, de manera que si requiere utilizar otra tarjeta sólo tendría que utilizar un nuevo conjunto de clases proporcionadas por el fabricante.

El segundo enfoque es mucho más especializado, puesto que trata de ver a la

tarjeta inteligente como una pequeña computadora con un sistema operativo capaz de ejecutar un Máquina Virtual de Java (JVM). De esta visión surgen la denominada Java Card, que permiten al desarrollador olvidarse completamente del fabricante de tarjetas y de especificaciones de bajo nivel, ya que la JVM dentro de la tarjeta permite el desarrollo de programas Java llamados Java Card Applets, que son totalmente independientes de la plataforma en la que se ejecuten (en este caso la tarjeta inteligente) y pueden ser transportados de una a otra sin realizar modificación alguna.

Como se verá a continuación la primera propuesta queda cubierta con la especificación denominada OpenCard Framework y complementada por la especificación PC/SC, que permite el desarrollo de programas escritos en lenguaje Java y traducidos después a instrucciones especificas de la tarjeta (APDUs).

Para el uso de tarjetas inteligentes con tecnología java contenida (Java Card), se hace uso de la especificación Java Card 2.1 proporcionada por Sun Microsystems. Que brinda la especificación para la programación de Applets y el método para portarlos en la tarjeta.

En un nivel superior encontramos a las librerías Piccola que ayudan en la administración de los programas contenidos en la tarjeta.

3.4.2. Tecnología OpenCard Framework (OCF)

La iniciativa OpenCard fue iniciada a principios de 1997 y actualmente no sigue siendo mantenida. La razón principal del uso del OCF es porque la capa de comunicación esta diseñada para permitir a los diferentes proveedores incorporar fácilmente su lector de tarjeta y servicios de tarjeta. Combinado con el poder de la tecnología Java, permite una independencia de la plataforma, así como independencia del proveedor en diferentes niveles.

OCF es un software que permite comunicarse con tarjetas inteligentes, el cual provee un API a las capas superiores, para que las aplicaciones puedan utilizar las funcionalidades de las tarjetas inteligentes. Específicamente es una colección de clases e interfaces definidas en el lenguaje de programación java [OCF].

3.4.3. OpenCard Consortium y el OpenCard Framework

OpenCard Framework (OCF) es un framework estándar desarrollado con tecnología

java, anunciado por un consorcio de Industrias para la interoperabilidad entre tarjetas inteligentes, a través de diferentes plataformas de hardware y software. Es un estándar abierto que provee una arquitectura y un conjunto de APIs que permite a los desarrolladores de aplicaciones y prestadores de servicios construir y publicar soluciones con tarjetas inteligentes en cualquier ambiente OpenCard compatible.

OCF esta diseñado para soportar todas aquellas plataformas que dispongan de tecnología Java como las computadoras personales (PCs), NetComputers (NCs) o cualquier tipo de dispositivo capaz de soportar tarjetas inteligentes como las Automatic Teller Machines (ATMs), terminales de punto de venta, set-top boxes y hand-helds (palm, organizadores personales). Habilita la interacción *Computadora Personal / Tarjeta Inteligente* (PC/SC por sus siglas en inglés) [OCF][PCSC-WORKGROUP].

3.4.4. Objetivos del OpenCard Consortium

Cuando buscamos obtener aplicaciones para tarjetas inteligentes desde el punto de vista de desarrolladores de aplicaciones, al menos tres roles cruciales pueden ser identificados:

- **Proveedores de Lectores de Tarjetas.**

  El proveedor de lector de tarjeta proporciona el lector de tarjeta. Cada proveedor proporciona una gama de modelos de lectores de tarjetas, desde unidades muy sencillas hasta los más sofisticados (en conjunto con display, PIN pad, teclado o cualquier dispositivo de entrada biométrico). Los proveedores no tiene ninguna interfaz estándar en común además de que utilizan diferentes protocolos

- **Proveedores de Sistema Operativo de Tarjetas.**

  Existe un gran número de compañías que compiten ofreciendo diferentes sistemas operativos y APIs para las tarjetas inteligentes. El resultado es una gran variedad de comandos y códigos de respuesta.

- **Emisores de Tarjetas y Proveedores de Aplicaciones/Servicios.**

  Actualmente hay entidades que emiten tarjetas inteligentes a los clientes y estos son quienes deciden donde colocar las aplicaciones residentes en la tarjeta inteligente. Por lo que el alojamiento del código puede variar considerablemente.

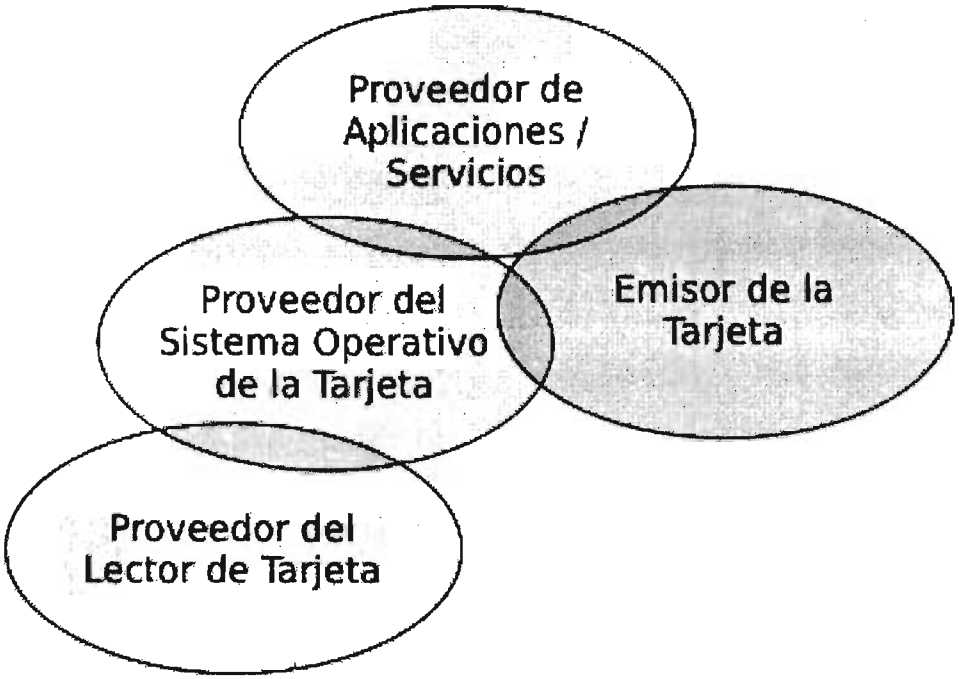

Figura 3.6. Partes involucradas en el desarrollo de software para tarjetas inteligentes

3.4.5. Objetivos del OpenCard Framework

- Independencia del proveedor de lector de tarjeta.
- Independencia del proveedor de sistema operativo de tarjeta.
- Independencia del emisor de tarjeta.

Para lograr estos objetivos, el núcleo de la arquitectura del OCF se divide en dos partes: La capa del lector (CardTerminal) y la capa de servicio (CardServices). Además la problemática de independencia del emisor es manejada por separado mediante el componente de administración de aplicación (ApplicationManagment) [SEIGERS].

3.4.5.1. CardTerminal

Esta capa proporciona acceso físico a la terminal de tarjeta y la tarjeta inteligente insertada para lo cual el driver compatible apropiado debe ser proporcionado por el fa-

bricante. También esta incluido el API de Java para poder acceder a los lectores soportados por la especificación PC/SC. Esta capa permite al OCF manejar el amplio rango de lectores utilizados.

3.4.5.2. CardService

Esta capa hace posible para el OCF trabajar con la gran variedad de sistemas operativos de tarjetas existentes y poder ofrecer diferentes funciones. A través del CardService se pueden utilizar los servicios FileAccessCardService y SignatureCardService.

El FileAccessCardService proporciona un conjunto completo de interfaces y clases (abstractas) haciendo las funciones del sistema de archivos ISO disponibles al programador. Como el resto del framework, estas clases e interfaces han sido diseñadas para ajustarse dentro del modelo de programación java existente.

El SignatureCardService ofrece los métodos para crear y verificar las firmas digitales basadas en algoritmos de llave pública como RSA y DSA. Adicionalmente a los servicios permite que la llave privada y pública puedan ser importadas a la tarjeta o generadas directamente en la tarjeta.

3.4.5.3. ApplicationManagment

Con la introducción de múltiples aplicaciones que pueden ser cargadas dentro de una sola tarjeta inteligente, nuevas dependencias -- como aquellas aplicaciones que están disponibles en la tarjeta o donde una aplicación y sus datos son físicamente colocados en la tarjeta -- son creadas. Es ahí donde los componentes del ApplicationManagment aparecen, este es capaz de:

- Localizar y seleccionar aplicaciones en cualquier tarjeta inteligente.
- Listar las aplicaciones con soporte a tarjetas inteligentes particulares.
- Instalar y desinstalar aplicaciones en las tarjetas inteligentes.
- Bloquear y desbloquear aplicaciones dentro de las tarjetas inteligentes.

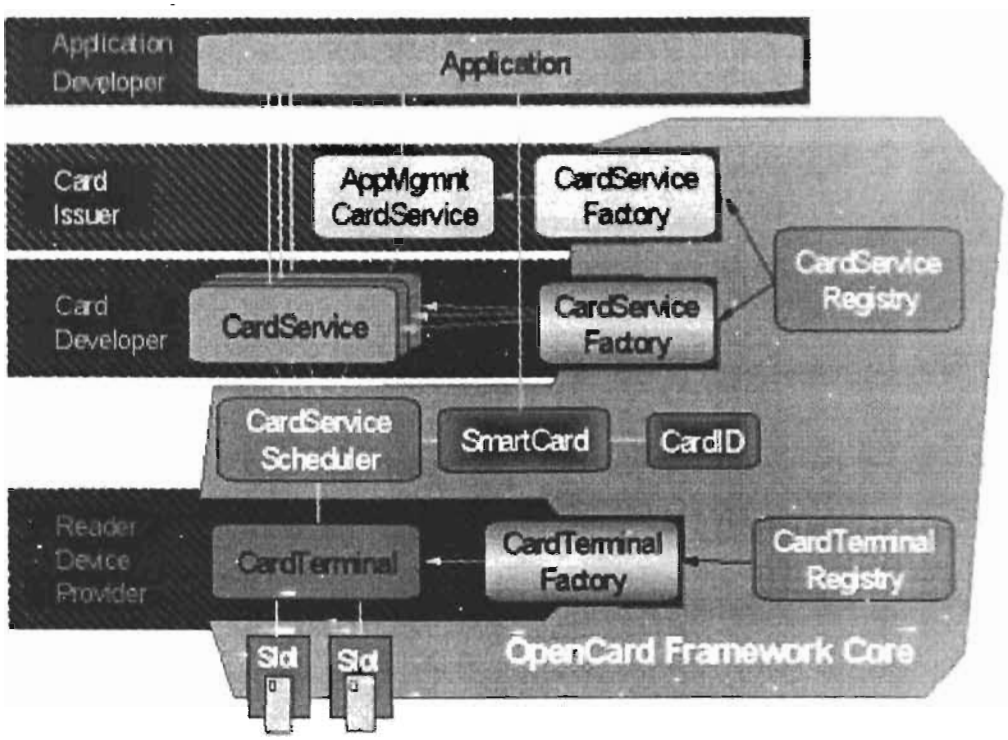

Figura 3.7. Componentes del OpenCard Framework [SEIGERS]

3.4.6. Ventajas del OCF

El OCF ofrece ventajas significativas para los desarrolladores de aplicaciones y proveedores de servicios así como también a los proveedores de tarjetas inteligentes y lectores de tarjetas.

Beneficios para los desarrolladores de aplicaciones y proveedores de servicios:

- *Independencia del vendedor*: Los desarrolladores pueden escoger tarjetas y lectores de diferentes proveedores.
- *Protección asegurada*: La extensibilidad de la arquitectura permite a los desarrolladores participar en futuros desarrollos de tecnología en tarjetas inteligentes a un costo muy bajo de migración del nivel de API.
- *Reducción del tiempo de desarrollo*: Los desarrolladores pueden disminuir los ciclos de desarrollo para la programación debido a la API de alto nivel.

- *Bajo costo de desarrollo*: Los desarrolladores pueden recuperar el costo extra de portar sus aplicaciones a diferentes plataformas y beneficiarse de los bajos niveles de requerimientos para finalizar las tareas solicitadas.

Beneficios para los proveedores de tarjetas inteligentes y lectores de tarjetas:

- *Incremento en la clientela*: Los proveedores pueden obtener acceso a nuevos mercados en busca de más consumidores.
- *Competencia mejorada*: Los proveedores pueden competir en términos de funcionalidad y son menos vulnerables a la predominación de un solo vendedor.
- *Menor esfuerzo de desarrollo*: Los proveedores heredan la funcionalidad proporcionada por el framework que reduce sus esfuerzos de desarrollo.

3.4.7. Uso de la tecnología OpenCard

La tecnología OpenCard aporta los elementos necesarios para desarrollar aplicaciones para tarjetas inteligentes utilizando la tecnología java, proporcionando una independencia del modelo de la tarjeta inteligente y lector utilizado [SEIGERS]. El OCF nos permite escribir aplicaciones en un lenguaje de alto nivel, delegándole la responsabilidad de la comunicación con el lector y la tarjeta a capas inferiores.

OCF nos facilita la comunicación de la tarjeta a través de un API de programación, este framework nos permite utilizar diferentes lectores de tarjetas (ver Anexo **OCF.B**), así como también tiene soporte para un amplio número de tarjetas criptográficas y tarjetas java (ver Anexo **OCF.C**).

Una característica adicional que proporciona el OCF es el uso del estándar PCSC, todos los lectores de tarjetas que soporten este estándar podrán ser conectados al OCF a través de APIs especificas ó por el bridge OCF-PCSC.

OCF es útil para programar tarjetas inteligentes con sistema operativo independiente, así como también poder las tarjetas java.

3.4.8. Especificación PC/SC

PC/SC es una especificación enfocada en la comunicación entre la computadora personal (PC) y el lector de tarjetas (SC) conectado a ella [PCSC-WORKGROUP]. El componente central de la especificación es el **Administrador de Recursos**, el cuál debe de integrase al sistema operativo de la computadora. Esta implementación esta disponible para computadoras de 32 bits utiiizando el sistema operativo Microsoft Windows y a través del proyecto M.U.S.C.L.E. [MUSCLE] en el sistema operativo Linux [KIRCH-2003]. Para la regulación de las tarjetas inteligentes, PC/SC esta basado y es compatible con el ISO 7816. Además PC/SC puede ser usado junto con cualquier tarjeta inteligente.

PC/SC con el nombre completo "***Interoperability Specification for ICCs and Personal Computer Systems 1.0***", abarca el uso de tarjetas inteligentes con las computadoras personales. En ocho partes la especificación recorre desde las características físicas requeridas por las tarjetas y lectores, hasta la capa de programación de aplicación. Se encuentra enfocado en la interoperabilidad de la tarjeta inteligente (llamada *Integrated Circuit Card*, o ICC) y el lector de la tarjeta (llamado *Interface Device*, o IFD) y la cooperación entre el lector y el sistema operativo de la computadora personal.

El PC/SC WorkGroup fue formado en mayo de 1996. Las compañías que condujeron la especificación PC/SC 1.0 son lideres de la industria de tarjetas inteligentes [PCSC-WORKGROUP]. La mayoría de estas compañías anunciaron productos compatibles con la especificación PC/SC, particularmente todos los tipos de lectores de tarjetas inteligentes.

Actualmente se ha liberado la especificación 2.0, que incluye una novena capa que brinda una expansión a las funcionalidades del IFD, además de dar soporte a las tarjetas inteligentes sin contacto (contactless). Para nuestro caso de estudio, nos enfocaremos en la versión 1.0. [SEIGERS]

Las ocho partes de la especificación PC/SC 1.0 se muestran en la siguiente figura y una breve descripción más adelante.

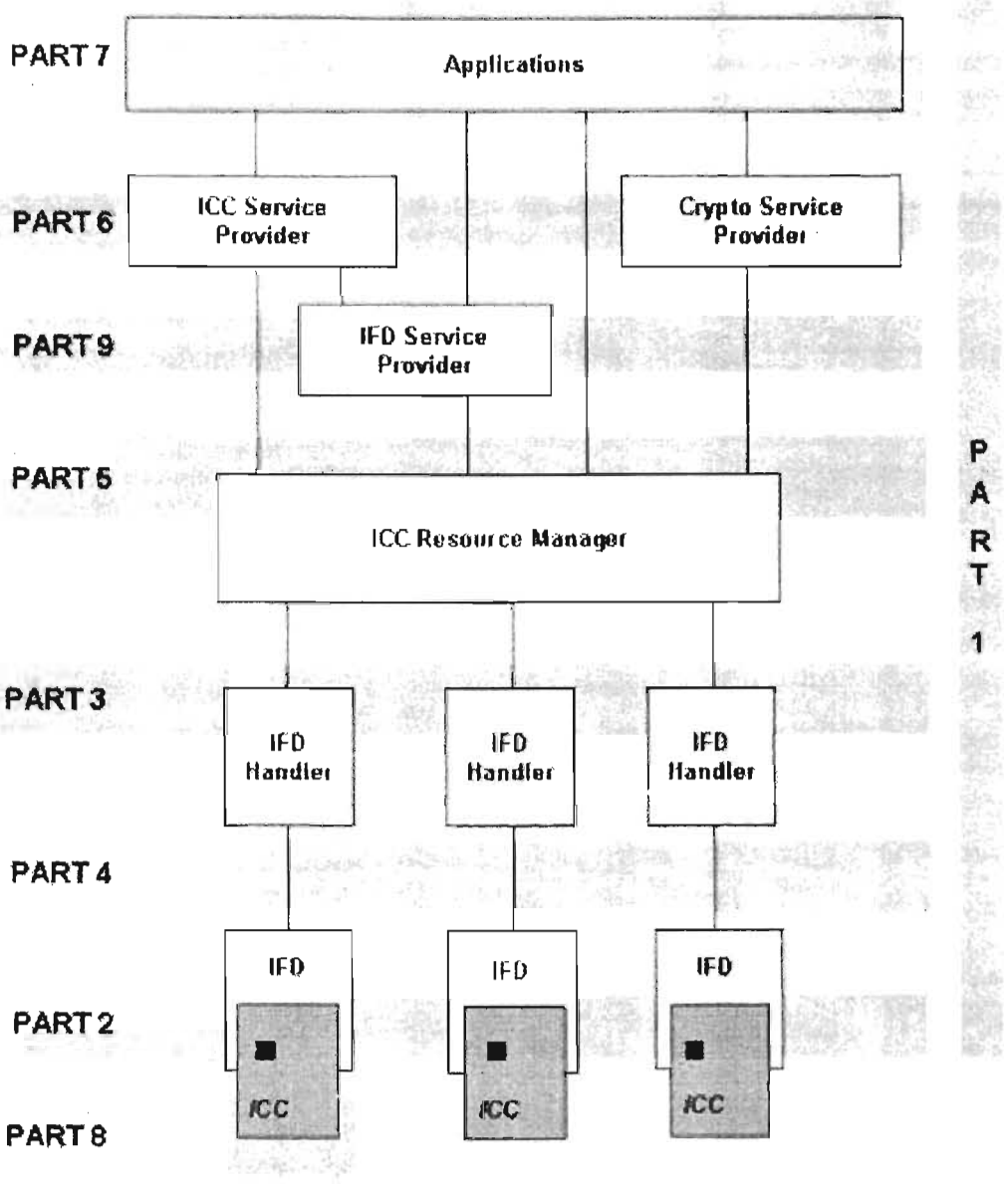

Figura 3.8. Capas de la Arquitectura y partes de la especificación PC/SC 1.0 [SEIGERS]

1. **Introducción y panorama general de la Arquitectura.**
2. **Requerimientos de Interfaces para tarjetas con IC compatibles e Interfaces de dispositivos.**

La segunda parte especifica las características físicas de la tarjeta y el hardware del lector, como por ejemplo los voltajes. Los protocolos de transporte de bajo nivel, incluyendo el protocolo de manejo de reglas de errores son especificado en este nivel. Además se encuentra la especificación para el mensaje esperado *Answer To Reset* (ATR).

3. **Requisitos para las interfaces de dispositivos conectado a la PC.**

La tercera parte especifica las características del subsistema de interfaz del dispositivo. Permitiendo a las capas más altas del PC/SC comunicarse con la Interfaz del Dispositivo (IFD) de manera independiente al dispositivo.

El IFD proporciona las funciones básicas como enviar comandos a la tarjeta o verificar la inserción de la tarjeta. Opcionalmente puede tener otras capacidades, como encender o apagar la tarjeta insertada o utilizar una autentificación con un PIN pad, teclado, lector de huella digital, lector de retina, etc.

4. **Consideraciones del diseño de IFD y Referencia de la información del diseño.**

Proporciona la ayuda y las guías de consulta para el diseño de los protocolos internos de la interfaz del subsistema del dispositivo.

5. **Definición del Administrador de Recursos ICC.**

El administrador de recursos del ICC es el componente central de un sistema PC/SC. Este controla todos los IFDs, así como también los recursos fuera de la tarjeta proporcionados por la interfaz de programación de aplicación de las tarjetas (llamada Proveedor de Servicio ICC) y las funciones criptográficas (el Proveedor de Servicio Criptográfico). El Administrador de Recursos ICC es un componente privilegiado, el cual debe de ofrecer el mismo grado de protección y seguridad como el sistema operativo base. En las versiones superiores del sistema operativo Microsoft Windows 95, el Administrador de Recursos ICC es parte integral del sistema operativo.

6. **Definición de la interfaz del Proveedor de Servicio ICC.**

Especifica el Proveedor de Servicio ICC y las interfaces del Proveedor de Servicio Criptográfico. Estas dos son las interfaces primarias que utilizará el de-

sarrollador de aplicación.

El **Proveedor de Servicio ICC** (ICCSP) debe ofrecer una clase obligatoria SCARD que mantenga el contexto de la comunicación con la tarjeta inteligente. El ICCSP puede ofrecer opcionalmente las clases para el acceso a los archivos en la tarjeta inteligente y para las funciones de autorización necesarias para acceder a la tarjeta.

El Proveedor de Servicio Criptográfico es opcional. Si la función criptográfica es necesaria para el acceso a la tarjeta, esa función se localiza por el Proveedor de Servicio Criptográfico. Tal función a menudo esta bajo limitaciones de exportación para los componentes criptográficos. Mantenerla localizada permite un mejor control. Para el Proveedor de Servicios Criptográficos las interfaces se especifican para la generación de llaves, administración de llaves, importación/exportación de llaves, firma digital, hashing (digestión), y los servicios de cifrado.

Los Proveedores de Servicio ICC necesitan ser registrados junto con la tarjeta y las interfaces de que soportan. Esto permite al sistema identificar y traer los Proveedores de Servicio ICC apropiados para una tarjeta determinada. Para la identificación de la tarjeta, se utiliza el ATR o partes de ella.

7. **Consideraciones en el Diseño/Desarrollo de Aplicaciones.**

Contiene consejos para el desarrollo de aplicaciones sobre cómo utilizar las interfaces proporcionadas por el Administrador de Recursos ICC para obtener información sobre los lectores de tarjetas instalados, esperar la inserción de una determinada tarjeta y otras tareas comunes.

8. **Recomendaciones para la implementación de seguridad y privacidad de los dispositivos ICC.**

Contiene consejos de cómo manejar la identificación, autentificación y almacenamiento seguro y como obtener integridad, seguimiento y confidencialidad de la información en una solución con tarjetas inteligentes. En la sección acerca de Servicios Criptográficos, también ofrece una descripción excelente de los métodos criptográficos actuales.

Microsoft ha proporcionado implementaciones de la especificación PC/SC

1.0 desde las plataformas Windows 95, Windows NT 4.0 y ha lanzado su implementación a la Red pública. Para Windows NT 5.0 el administrador de recursos ya forma parte del sistema operativo, desde Windows 98 se incluye como parte del paquete de instalación.

3.4.9. Similitudes en Arquitectura del OCF y el PC/SC

Las iniciativas OCF y PC/SC tienen el objetivo de soportar la interoperabilidad de diferentes elementos de una solución con tarjetas inteligentes, de tarjetas, lectores y software de aplicación. Para comprender como realizan estos objetivos la siguiente figura muestra la correspondencia de los componentes de la arquitectura del OCF y PC/SC.

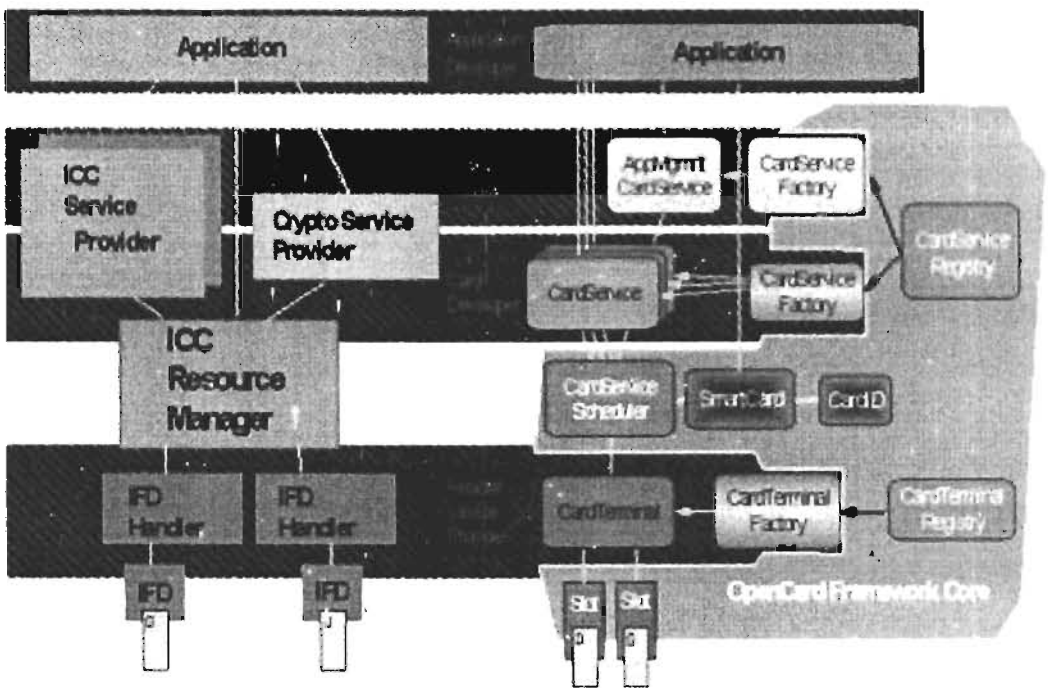

Figura 3.9. Comparación de los componentes de la Arquitectura OCF y PC/SC [SEIGERS]

Las aplicaciones desarrolladas para tarjetas inteligentes utilizan principalmente los servicios específicos de la tarjeta y los de administración de aplicaciones. En la especificación PC/SC la capa del Proveedor de Servicios del ICC ofrece los servicios,

en el OCF quien realiza esta tarea son los CardServices. Los servicios del lector de tarjetas en el OCF se concentran en la capa del CardTerminal que corresponde al de Interface Device Subsystem en el PC/SC. En conclusión, la mayoría de los componentes en el OCF y el PC/SC tienen mucho en común.

3.4.10. Relación del OpenCard Framework y el PC/SC

Tanto el OCF como el PC/SC proporcionan acceso a las tarjetas inteligentes desde la computadora de diferente forma. Podemos pensar que tienen conceptos y mecanismos en común. Comparando los componentes individuales de cada uno, nos damos cuenta de que en realidad es cierto.

3.4.11. Colaboración entre el OCF y el PC/SC

A pesar de sus similitudes y debido a que surgieron de diferentes especificaciones cada una de estas se enfoca a resolver diferentes problemas. El PC/SC se concentra más sobre el dispositivo lector. El OCF hace énfasis en proporcionar una separación de los servicios específicos de aplicación y los servicios de administración de aplicación.

El OCF fue construido y diseñado por programadores Java para programadores Java. Esta centrado en el mundo de Java y la red. Los programadores Java se sienten cómodos con el estilo de programación del OCF, por ejemplo el manejo de archivos de entrada y salida de las tarjetas inteligentes es muy similar a la entrada y salida estándar nativa de Java.

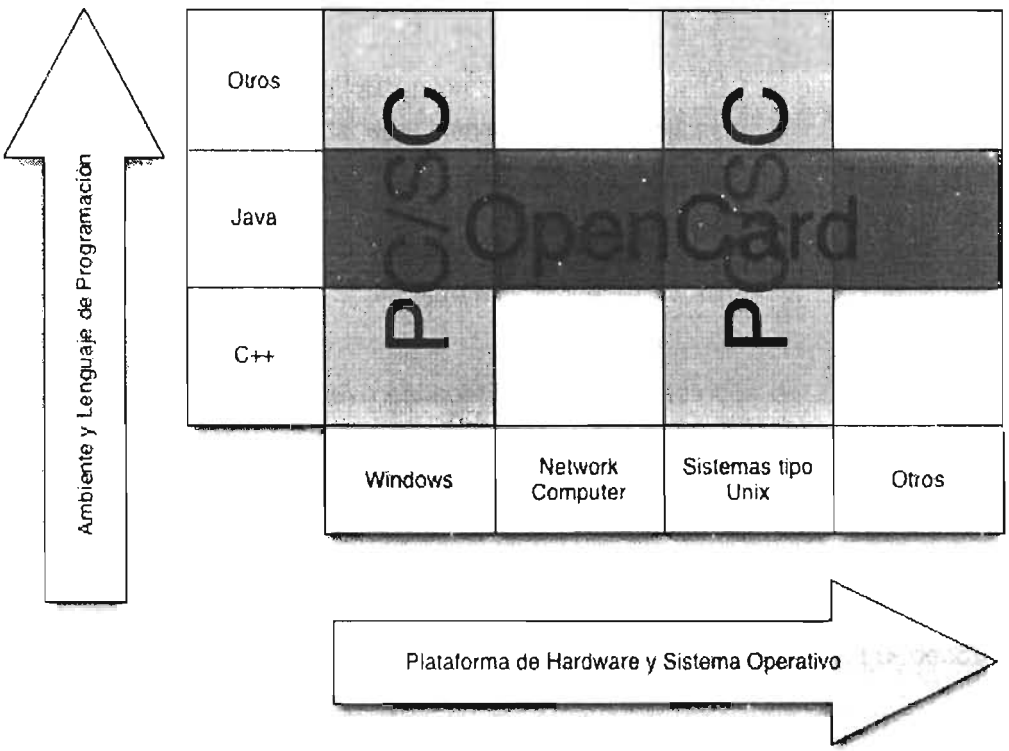

Figura 3.10. Alcance y traslape entre el PC/SC y OCF.

3.4.12. TITANIUM Piccola

Piccola es una librería de administración de tarjeta inteligente que permite a los usuarios manejar aplicaciones en tarjetas inteligentes multiaplicación [Piccola]. Esto lo realiza proporcionando un framework de administración que permite manipular de manera segura tanto la tarjeta como las aplicaciones que puedan residir en ellas independiente del fabricante. El objetivo primario apunta hacia las tarjetas inteligentes compatibles con la especificación *Global Platform*.

Ayuda a explorar un nuevo escenario e innovar métodos de administración de aplicaciones en tarjetas con un sistema portátil de dispositivo embebido en un nuevo modelo de negocios.

Piccola hace uso de OCF, para la capa de comunicación, puesto que OCF esta diseñado para permitir agregar servicios de tarjetas y lectores de diferentes fabricantes fácilmente. Combinado con el poder de Java también proporciona independencia

del fabricante a diferentes niveles.

Cualquier tarjeta inteligentes certificada por la Global Platform puede ser utilizada con Piccola. Siempre y cuando las llaves estáticas de la tarjeta o la llave madre/ maestra sean conocidas.

Aunque Piccola esta diseñado para trabajar con Java Cards, pero no utiliza el Java Card development kit puesto que se basa en su propia implementación.

3.4.13. M.U.S.C.L.E.

De la siglas en inglés *Movement for the Use of Smart Cards in a Linux Environment.* MUSCLE es un proyecto para coordinar el uso de tarjetas inteligentes y el desarrollo de aplicaciones bajo el entorno del sistema operativo Linux. El propósito es desarrollar un conjunto de *drivers* que cumplan con las APIs (Application Programming Interface) y el administrador de recursos para diferentes tarjetas inteligentes y lectores para el ambiente GNU. El código fuente es distribuido libremente en su sitio web [MUSCLE] el cual soporta todas las tarjetas inteligentes que cumplen con la especificación ISO-7816-4.

El proyecto MUSCLE ofrece una gama de software de desarrollo que nos permite interactuar con las tarjetas inteligentes, pero la pieza central de todo el desarrollo es la aplicación **pcsc-lite** que permite establecer una comunicación con el lector a bajo nivel, con lo cual se pueden enviar los APDUs necesarios para controlar la tarjeta. Esta parte de software controla únicamente el lector y es el que soporta un mayor número de dispositivos.

3.4.14. OpenSC

OpenSC actualmente soporta un gran número de tarjetas inteligentes que cumplen con la especificación ISO-7816, incluyendo las tarjetas Cryptoflex, GPK, Setec y CardOS/M4 (el cual es el sistema operativo utilizado p.e. por el eToken de la compañía Aladdin).

OpenSC corre bajo los ambientes Linux, Windows y en Mac OS X (aunque en menor proporción). Cuando hablamos del lector, OpenSC puede ser utilizado con los drivers PCSC, CT-API y OpenCT para terminales de tarjetas. La librería OpenSC esconde la diversidad de estas tarjetas y drivers por debajo de una API que permite uti-

lizar la tarjeta con funciones de bajo nivel como son la lectura/escritura de un archivo o la generación de una firma digital. El código fuente es distribuido libremente en su sitio web [OpenSC].

OpenSC provee la librería **libopensc** que permite utilizar las tarjetas inteligentes compatibles con la especificación ISO-7816-4, además de proveer otras utilidades para inicializar la tarjeta (dar formato) y escribir datos sobre la misma (almacenar llaves privadas) a través del estándar RSA denominado PKSC#15. También incluye en modulo **pam_opensc** que permite a PAM realizar la autentificación basada en tarjetas inteligentes.

Capítulo 4.

Planteamiento del modelo

4.1. Introducción.

Una vez que realizamos el estudio de las diferentes tecnologías asociadas al desarrollo de aplicaciones con tarjetas inteligentes, hemos tratado de reducir las opciones que contamos para el desarrollo de aplicaciones, debido a que existen diferentes alternativas y enfoques se contraponen.

Con nuestro estudio determinamos dos diferentes modelos de identificación, las diferencias entre cada uno de estos, son a nivel de integración de datos así como la viabilidad para utilizar cada una de las tecnologías involucradas.

Cada modelo tiene la finalidad de satisfacer el proceso de identificación del portador de la tarjeta inteligente, así como poder determinar la información mínima necesaria para poder ofrecer un servicio.

4.2. Modelo 1: Identificación de alumnos basado en tarjetas inteligentes multiaplicación

4.2.1. Enfoque del modelo propuesto

Este primer modelo habla sobre la integración de servicios escribiendo información relevante en una tarjeta inteligente. Debido a esta característica, la tarjeta debe ser capaz de almacenar objetos con la información requerida, así como también debe ser recuperable, para este propósito las tarjetas inteligentes que mejor se adaptan son las conocidas Java Cards.

Entendemos "servicio" como un sistema computacional (no necesariamente bajo nuestro control) que sirve a una dependencia de la facultad de ingeniería para procesar información acerca de los alumnos. Ejemplos:

- Sistema de inscripciones.
- Sistema de préstamo de libros en la biblioteca.
- Sistema de autentificación en laboratorios de cómputo.

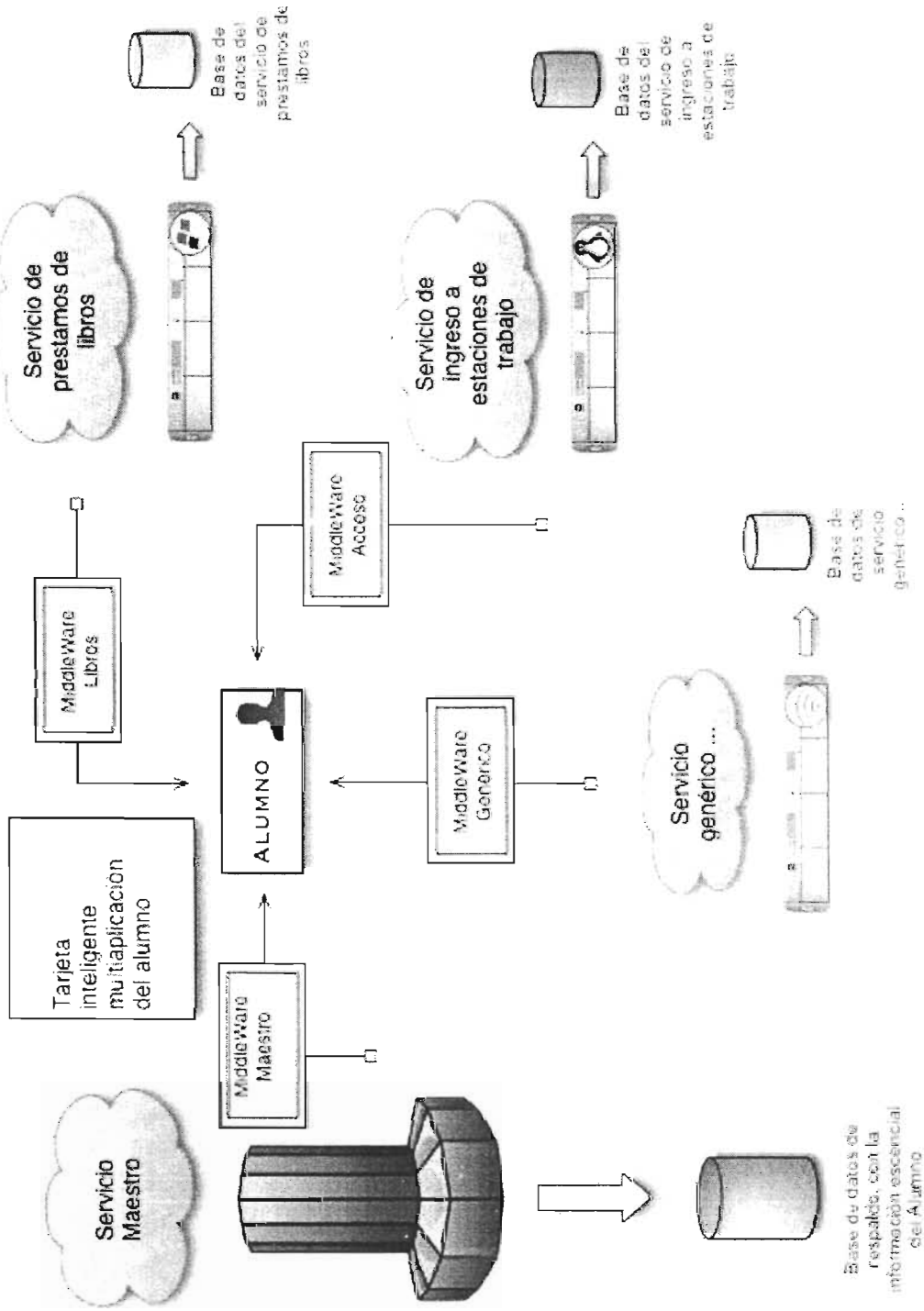

Figura 4.1. Modelo basado en tarjetas inteligentes multiaplicación.

4.2.2. Descripción del modelo de identificación

Existe un "servicio maestro" que es aquél que tiene el control sobre toda la información esencial del alumnos, es decir, la información que es común a todos los servicios. El servicio maestro usará tarjetas inteligentes para comunicar ésta información a los otros servicios. Únicamente el servicio maestro puede cambiar la información esencial de los alumnos. La información existente en la base de datos del servicio maestro es en todo momento consistente con la almacenada en las tarjetas inteligentes.

Cada alumno tiene una tarjeta de identificación y sólo una tarjeta de identificación, expedida por el servicio maestro.

Para comunicarse con los diferentes servicios, existen middlewares (uno por servicio) que obtienen cualquier información esencial almacenada en la tarjeta por el servicio maestro y la adecuan según las necesidades del servicio.

Cada servicio puede obtener la información que requiera (incluso información esencial) sobre el alumno para utilizarla como mejor le convenga. Además podrá agregar información adicional en la tarjeta del alumno que sirva para sus propósitos específicos, está información no podrá ser modificable por otros servicios, únicamente el servicio que genero esa información será capaz de realizar modificaciones sobre los datos o incluso eliminarlos.

Ningún servicio diferente al servicio maestro será capaz de realizar modificaciones sobre la información esencial, tampoco eliminarla o ampliarla. Solo el servicio maestro es capaz de realizar estas funciones.

La información que nosotros hemos consideramos esencial que se encuentre almacenada en la tarjeta es la siguiente:

Tabla 4.1. Información esencial del Alumno

nombre(s)
apellido paterno
apellido materno
clave única de registro de población (CURP)

número de cuenta
clave de carrera
estado académico (inscrito, regular, baja, etc.)
fecha y hora de expedición
fecha y hora de expiración
número de reposición
hash (clave de expedición única)

4.2.3. Características

La característica principal de este modelo es que la información esencial se lleva dentro de la tarjeta, es decir si algún servicio requiere conocer la información del alumno al cual se le brinda el servicio lo pueda localizar dentro del dispositivo, con lo cuál se logran evitar los errores de captura que usualmente se comenten al dar de alta nuevos usuarios en algún sistema, con lo cual se tienen registros homogéneos. Otra ventaja que proporciona este modelo de datos es que de esta manera es posible obtener información especifica del alumno en mucho menor tiempo puesto que en lugar de buscar un registro en una base de datos de N registros, el universo se reduce una búsqueda especifica en solo un registro.

En este modelo planteamos que los servicios pueden generar información especifica para un alumno y tener la posibilidad de almacenarla en la tarjeta, esta característica es aprovechada para poder almacenar las llaves primarias requeridas para la búsqueda de información en la base de datos del servicio especifico, por ejemplo si un alumno requiere el préstamo de un libro, el servicio preguntara por los datos principales del alumno con el propósito de desplegarlos en pantalla, en la misma búsqueda obtiene el identificador de alumno que tiene en la base de datos del servicio (almacenado previamente por el servicio), con esto agilizará la búsqueda debido a que se realizará con la llave primaria y con la ayuda de índices, una vez obtenido el registro se sabrá si el alumno tiene derecho a realizar el préstamo.

4.2.4. Ventajas

La ventaja de poder utilizar una tarjeta multiaplicación es el poder permitir que una sola tarjeta pueda compartir información universal con diferentes aplicaciones, con lo cual los datos esenciales que se encuentren contenidos en la tarjeta deben ser accesibles en todo momento por las aplicaciones que los soliciten, se debe tener especial atención de que esta información no pueda ser modificable por cualquier aplicación y

que solo el servicio maestro pueda realizar las modificaciones en esta información. Además esto permite que no existan errores de captura cada vez que el usuario active un nuevo servicio, debido a que los datos esenciales serán almacenados una sola vez, los demás servicios podrán almacenar una copia local de esos datos libres de error.

También es indispensable que la información registrada por un servicio no pueda ser modificable por otros servicios, debido a que si esto ocurriera podría haber aplicaciones que modificaran intencionalmente los datos del alumno para algún servicio especifico, solo basta con imaginar con algún servicio de prepago donde se utilizara la tarjeta inteligente como una especie de monedero electrónico, donde el servicio que escribe el saldo es el de la caja de pagos universitaria y que un usuario mal intencionado desarrolle un aplicación especifica para modificar el saldo que escribe el servicio de pagos, esto incurriría en una gran vulnerabilidad.

Por lo tanto es muy importante que el área especificada de almacenamiento de datos para cada servicio tenga una dirección de datos única, que sea solo modificable por el servicio apropiado, mediante una verificación exhaustiva previa, antes de poder modificar cualquier dato.

En servicios en los que se requiriera trabajar en conjunto, en donde una parte adquiere los datos y escribe la información con la cual se va a operar, será necesario otorgar los permisos para poder acceder a esas zonas de datos y poder recuperarlos desde el servicio que los necesita.

La mayor virtud que representa este modelo es que en el caso de que la tarjeta fuese extraviada, esta podría ser repuesta rápidamente, debido a que el servicio maestro contiene la copia de los datos esenciales con los que se expidió la primera vez y en caso de hacer alguna actualización sobre los datos estos se harían solamente en el servicio maestro, los cuales se pueden reflejar rápidamente en los otros servicios. Para poder tener una integridad completa es necesario que los servicios sean responsables de los datos que se escriben dentro de la tarjeta inteligente, puesto que al realizar la etapa de recuperación de datos, estos tendrán que escribir los datos que se encontraban en la tarjeta pertenecientes a su servicio la ultima vez que fue utilizada.

4.2.5. Desventajas

Una de las principales complejidades que nos encontramos en el uso de tarjetas inteligentes multiaplicación la escasa información por parte de los proveedores, debido a que ellos ofrecen directamente el desarrollo de las aplicaciones con la mínima participación del usuario en el proceso. Ejemplo de esto son las tarjetas inteligentes Gemplus GemXpresso las cuales contienen un applet precargado por el fabricante con el cual se administran las aplicaciones que se requieran cargar en la tarjeta inteligente.

Otra limitante que se obtiene al tratar de implementar este tipo de soluciones es la dependencia que se tiene con el proveedor, ya que al estar sujeto a la especificación de programación de la tarjeta también se deben de obtener programas específicos para el manejo de las tarjetas inteligentes, retomando el ejemplo este applet solamente se comunicara con el software adecuado para completar la carga de las aplicaciones en la tarjeta, para el mismo caso el software necesario GemXplorer RAD II, es licenciado por el fabricante. El cual tiene un costo bastante elevado para los propósitos de esta investigación, por tal motivo no se pudo realizar una evaluación especifica y compararlo directamente con otras opciones como en el caso del "Muscle Card Applet".

Como se menciono en el capitulo anterior, el uso de las tarjetas inteligentes multiaplicación aun es incipiente pues por más que se desee aun no se pueden integrar todos los servicios en uno solo, debido a características faltantes como protección de memoria y asignación de espacios en la tarjeta inteligente de forma estándar entre los diferentes proveedores de servicios.

4.3. Modelo 2: Identificación de alumnos basado en tarjetas inteligentes criptográficas

4.3.1. Enfoque del modelo propuesto

Partimos del hecho de que las tarjetas inteligentes no permiten almacenar información de una aplicación especifica dentro de la tarjeta inteligente (tarjetas criptográficas), la información contenida dentro de esta es especifica y solo proporciona los elementos necesarios para lograr la autentificación.

Entendemos como Autoridad Certificadora aquella entidad facultada para la expedición de certificados digitales dentro del ámbito de una PKI (Public Key Infrastruc-

ture - Infraestructura de Llave Pública). Esta autoridad certificadora debe tener un control de su llave secreta, ya que con esta se firman los certificados que expedirá, además de dar valides a toda la estructura de la PKI. Aunque la autoridad certificadora puede realizar todo el trabajo (registro de los datos del usuario, verificación de la identidad de la persona a la que se le expedirá el certificado, elaboración del precertificado sujeto a firma), es recomendable contar con una, o varias, Autoridad Registradora a la cuál se le puedan delegar funciones especificas.

No es nuestro cometido establecer las reglas para obtener una PKI robusta, sino simplemente alinear nuestro Modelo a esta forma de trabajo, que es ampliamente aceptada, la cual permitirá crear un ambiente de trabajo ampliamente seguro.

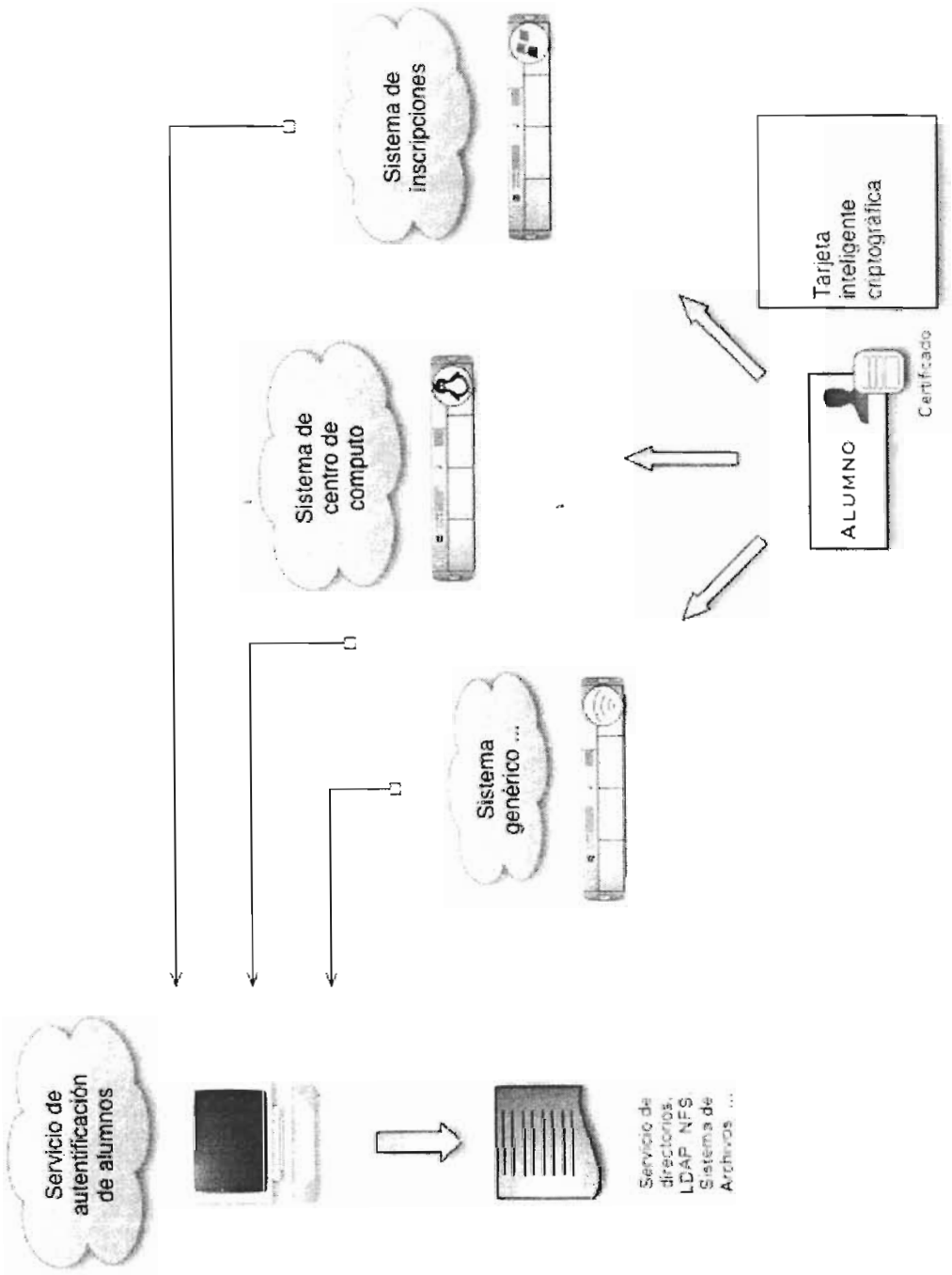

Figura 4.2. Modelo basado en tarjetas inteligentes criptográficas.

4.3.2. Descripción del modelo de identificación

Existe una Autoridad Certificadora que se encarga de expedir las tarjetas inteligentes con los mecanismos necesarios de seguridad para validar la autenticidad de la persona a quien se le emite.

Esta Autoridad Certificadora mantendrá una copia del certificado expedido, así como también información sobre el alumno, hemos de mencionar que el certificado contara con una vigencia este contiene la fecha de expedición así como la fecha de expiración.

El alumno contará con una tarjeta de identificación con características criptográficas, esto quiere decir que la tarjeta podrá almacenar llaves criptográficas, para poder acceder a estas llaves es necesario conocer el PIN (*Personal Identification Number*) además de contar con la tarjeta inteligente.

Los sistemas existentes que requieran conocer la autenticidad del usuario deberán de implementar el uso de un servicio de autentificación con el cual lograran asegurar que la persona quien dice ser, es a la que se requiere otorgar el servicio.

Este servicio de autentificación utilizara un sistema de directorios (LDAP, NFS, sistema de archivos) para almacenar los certificados y poderlos ofrecer para localizar al usuario que desea utilizar el servicio.

Las tarjetas no podrán almacenar información del usuario o de algún servicio especifico, solo podrán contener las llaves privadas que hayan sido expedidas por la autoridad certificadora. La única información que será común a todos los sistemas sera la que provenga del certificado expedido por la autoridad certificadora, en el cual puede incluir tanto el nombre del alumno, como el plantel al que pertenece

Debido a esta limitante, las aplicaciones que sean construidas con este enfoque deben de mantener su propia copia de los datos del usuario, así como los estados convenientes de su ultima acción en el sistema.

4.3.3. Características

Sabemos que en los mecanismos de identificación donde se usan algoritmos de llave

asimétrica mencionan que para una llave publica (PK) le corresponde una y solo una llave secreta (SK) por lo tanto al utilizar un certificado emitido por una autoridad certificadora de confianza (en este caso la autoridad certificadora UNAM que llamaremos CertiUNAM),lo que estamos realizando es publicar un certificado que será válido dentro de nuestra PKI, y los mensajes o secretos que se cifren con la llave pública contenida en el certificado solo podrán ser descifrados mediante la correspondiente llave secreta. En nuestro caso los servicios que requieran un mecanismo de autentificación podrán utilizar el servicio de autentificación, el cual conociendo el nombre del usuario buscara el certificado asociado y realizando un desafío con la llave pública contenida en el certificado tratara de que el usuario demuestre que es la persona que dice ser respondiendo con el secreto utilizando la llave secreta correspondiente.

El modelo de identificación será válido siempre y cuando los alumnos tengan responsabilidad sobre su tarjeta de identificación, podemos suponer que en el momento de que algún alumno haga mal uso de su tarjeta puede ser totalmente implicado debido a las características de unicidad que brinda la tarjeta inteligente en conjunto con la PKI, Estas implicaciones a las cual se haga merecedor el alumno a causa del mal uso de la tarjeta no solamente tendrán quedar claras en el manual de procedimiento del sistema sino que deben ser parte del reglamento general.

La tarjeta por si misma no brinda ningún derecho sobre la información (*el que puede hacer*) que pudiese contener algún sistema, solamente es el medio con el cual se puede identificar una persona (*el quien es*), por lo cual las acciones realizadas dentro del sistema son independientes del sistema de autentificación, una vez que se identifica al alumno el sistema que brinda el servicio deberá establecer los permisos en las acciones sobre la información (consultar, eliminar, modificar, etc.).

El acceso se puede realizar de forma distribuida, debido a que la tarjeta contiene la llave privada para la llave pública contenida en el certificado expedido por CertiUNAM, esto nos da la garantía de acceso de que solamente una tarjeta inteligente contendrá esa llave privada, por lo cual en el momento que se desee autentificar a un alumno se realizará a través del certificado. No necesariamente deberá de existir un sistema de autentificación para cada uno de los servicios sino que este se puede delegar a un solo sistema de autentificación, con lo cual se centralizará la información, este caso solo existiría un sistema de autentificación para la facultad de ingeniería.

4.3.4. Ventajas

El uso de las tarjetas inteligentes criptográficas permitirá que los servicios que se proporcionen se hagan de manera segura, solo se proporcionara servicio a los alumnos que cuenten con su tarjeta de identificación y que estén autorizados para ello.

No existirán errores de búsqueda en el momento de autentificar al alumno, debido a que no se podrá confundir con otro gracias al uso de llaves asimétricas. Es decir en la búsqueda del alumno que solicita el servicio no se puede obtener otro alumno que no sea ese en el momento que la autentificación sea exitosa, solo podrá dar como resultado un alumno autentificado.

La mayor ventaja que encontramos es que con el uso de estas tarjetas inteligentes criptográficas es el tener seguridad en la identidad de de una persona (siempre y cuando se use de forma correcta), la baja probabilidad de poder romper la llave pública y de comprometer la seguridad, dado que no se tendrán múltiples archivos de la llave privada (como suele suceder en casos como el uso de PGP), en diferentes computadoras, sino que se encontrará dentro de la tarjeta inteligente.

4.3.5. Desventajas

El mayor inconveniente es que no se pueden almacenar otros datos en la tarjeta que no sean llaves criptográficas, es decir las aplicaciones no pueden almacenar datos específicos dentro la tarjeta. Los datos requeridos como llaves primarias en las aplicaciones no podrán ser trasportados dentro de la tarjeta por lo cual en la búsqueda de información de un alumno especifico, se tendrá que realizar la consulta sobre los N registros que contenga la base de datos del servicio, en lugar de hacer una consulta especifica con la ayuda de índices primarios.

El uso de la tarjeta esta limitado solo a aquellos servicios que requieran el uso de un mecanismo de autentificación que les brinde seguridad de que la información sensible que se proporcionará sera únicamente para la persona que tiene derecho de usarla.

Las aplicaciones existentes que deseen utilizar este mecanismo de autentificación, tendrán que reescribir parte de su código para hacer uso de esta herramienta, inclusive en algunos sistemas tendrán que generarse completamente desde cero.

La seguridad del modelo recae sobre la correcta administración y mantenimiento que se realice en el sistema de autentificación, en lugar de la tarjeta inteligente, aquí vemos la tarjeta como un dispositivo y no como parte esencial del modelo, si bien la tarjeta almacena la llave privada mediante un PIN. La perdida de la tarjeta y la exposición del PIN podría dar beneficios a un atacante de adquirir permisos sobre algún sistema.

4.4. Evaluación de Tecnologías y Herramientas

Dada la gran diversidad de herramientas para el desarrollo de aplicaciones y soluciones basadas en tarjetas inteligentes, es necesario hacer un análisis de evaluación para escoger entre las diversas tecnologías existentes. Ésto es aún más necesario cuando algunas de estas tecnologías se contraponen, resultan incompatibles entre sí, o implican filosofías de desarrollo muy distintas.

Por supuesto, la evaluación de dichas tecnologías y herramientas fue mucho más allá de la simple discusión escrita que presentamos aquí. Durante varios meses probamos infructuosamente muchos programas, lectores y tarjetas inteligentes distintas. La inversión en tiempo y dinero representó una de las más grandes dificultades de ésta investigación. La integración entre dichas herramientas también representó un gran problema, debido a que a veces las herramientas parecían funcionar bien por sí solas, pero no en conjunto.

Así, la siguiente discusión tiene los siguientes objetivos, que deben ser entendidos como **complementarios y paralelos**:

- Determinar, en función de la disponibilidad de las tecnologías existentes, el modelo de identificación que se ha de realizar el prototipo.
- Determinar, en función del modelo elegido, las herramientas específicas que usaremos para los prototipos.

4.4.1. Sistema Operativo

Como se dijo antes, en la sección de discusión de requerimientos, sería deseable contar con aplicaciones y soluciones multiplataforma, que se pudieran ejecutar en cualquier computadora, independientemente de su arquitectura o sistema operativo. Desgraciadamente hasta el día de hoy esta idea sólo se ha implementado en un nú-

mero limitado de aplicaciones entre las que se cuentan mucho más aplicaciones de alto nivel que aquellas donde es necesario acercarse al hardware.

Hasta hace unos meses, las tarjetas inteligentes resultaban un dominio desconocido para nosotros. Pretender incursionar en su uso sobre múltiples plataformas hacía aún más difícil la tarea de entender su funcionamiento. Fue necesario pues, elegir una plataforma *principal* de desarrollo, para reducir el dominio. En éste sentido, el sistema operativo **GNU/Linux** (simplemente Linux de ahora en adelante) nos pareció una buen punto de partida: desde su aparición en el año 1991, GNU/Linux continúa ganado terreno en muchas áreas de la computación. Los derechos legales que nos otorga al ser software libre lo hacen ideal para una investigación como la nuestra. Además de ésto, se trata de un sistema operativo que al día de hoy se utiliza ampliamente en nuestra facultad.

4.4.2. Acceder a lectores y terminales

Una vez escogido el S.O., existen básicamente dos tecnologías y dentro de ellas 3 herramientas que nos permiten acceder a lectores y terminales. Por un lado, tenemos a *Open Card Framework* que promete *interoperabilidad* en cualquier plataforma que soporte a la tecnología Java, y la facilidad de acceso y la consistencia que conocemos de éste lenguaje de programación. Por otro tenemos a PC/SC, tecnología apoyada por muchos miembros de la industria de la computación, específicamente Microsoft.

Si bien primero nos inclinamos por OpenCard, **PC/SC** resultó ser una mejor solución ya que al día de hoy se encuentra más actualizado en cuanto a controladores de hardware nuevo (USB, protocolo T1, etc...), hardware que además es el que se encuentra con mayor facilidad en el mercado. Además, OpenCard parece estar en decadencia franca en cuanto a sus actualizaciones, ya que no permite el uso mas que de un número muy limitado de modelos de tarjetas.

Entre las herramientas específicas que implementan el estándar PC/SC para Linux, hubo que escoger entre OpenCT y PCSC-lite. Ambas herramientas proporcionan básicamente la misma funcionalidad, pero para modelos distintos de lectores. Específicamente, OpenCT se enfoca a lectores con múltiples ranuras o teclados numéricos integrados. PCSC-Lite por otro, cuenta con controladores libres para dispositivos CCID (como los lectores que se pueden conseguir en México). Así, para el manejo de nuestros lectores, escogimos **PCSC-lite**.

4.4.3. ¿Tarjetas multiaplicación o criptográficas?

El modelo de autentificación en el que más estamos interesados es el modelo de tarjetas multiaplicación. Las capacidades que éstas tarjetas tienen son muy amplias y su uso comienza a generalizarse en todo el mundo y México no es la excepción (Cinemex, Cinépolis, tarjetas SIM en telefonía GSM...). Sin embargo, su desarrollo es relativamente nuevo y aún no existen estándares muy completos al respecto: cada fabricante de tarjetas propone el suyo. Diversas alianzas que surgieron en la industria no sobrevivieron más que a una existencia efímera y nuevos grupos se crean. En éste sentido las tarjetas criptográficas proporcionan una mayor seguridad en cuanto a la inversión de tiempo y dinero. Existe mucho más software listo para utilizarse que en el caso de las tarjetas criptográficas. Los estándares de la industria han hecho también más baratas a éstas tarjetas como al software que se necesita para programarlas y utilizarlas. Como ejemplo, el precio del software de desarrollo de Gemplus para tarjetas GemXpresso PRO (Java Cards multiaplicación) es de aproximadamente 399 USD. Para usar éste software es necesario tomar un curso de 5 días en las instalaciones de Gemplus en Houston, E.E.U.U., cuyo costo simplemente sale de nuestras capacidades financieras.

Es por estas razones que decidimos usar **tarjetas Criptográficas** en lugar de tarjetas multiaplicación. La herramienta que resultó ser ideal para acceder a éstas tarjetas fue **OpenSC**, software que al día de hoy a cobrado mucha fuerza al ser usado en las nuevas (diciembre, 2004) identificaciones nacionales de Finlandia y Bélgica. Además se trata también de software libre, descargable en su totalidad desde Internet, apoyado por una comunidad creciente de desarrolladores y usuarios.

4.4.4. El modelo elegido para realizar el prototipo

Escogidas las tarjetas criptográficas, el modelo de identificación que ha de ser implementado está también definido. Las herramientas que utilizaremos además de las que ya mencionamos, son piezas de software relativamente conocido y utilizado en ambientes Unix: NFS, PAM y OpenSSL.

Capítulo

5.

Generación de Prototipos

« The only way to understand the wheel is to reinvent it. »

-Michael Bond, University of Cambridge

5.1. Introducción

Con el propósito de evaluar la factibilidad del modelo propuesto, hemos usado algunas de las tecnologías que describimos en el capítulo 3 para implementar sistemas prototipo. Para ésto, escogimos dos sistemas que consideramos muy representativos en cuanto a la utilidad de las tarjetas inteligentes para la identificación de alumnos.

El primero es un sistema de autentificación centralizado para estaciones de trabajo Unix. En este sistema el objetivo fue lograr autentificar usuarios distintos en diversas estaciones de trabajo teniendo la información relevante para la autentificación en un solo *host*. Por otra lado los secretos de autentificación debían ser guardados exclusivamente en tarjetas inteligentes (y en alguna otra base de datos, para propósitos de respaldo). Así, cada usuario debería poder usar la misma tarjeta inteligente para acceder a diversas estaciones de trabajo (con instancias diferentes del sistema operativo). Por supuesto, hubo que comunicar las estaciones de trabajo con el *host* que albergaba la información relevante para la autentificación. Mantener los mismos archivos de usuario en todos los *hosts* sería además un atributo deseable. Por último, evaluamos la seguridad de este sistema.

La segundo sistema se trata de una aplicación de consulta de historiales acadé-

micos. En este sistema los alumnos se autentifican mediante una tarjeta inteligente, de manera que ningún alumno puede revisar información que no le pertenece. Como se verá más adelante, en este sistema se desarrollan las bases para la implementación de un sistema general de trámites escolares, en el cual los alumnos pueden entrar al sistema desde cualquier estación de trabajo.

5.2. Recursos

A continuación, presentamos brevemente los recursos de hardware y software con los que contamos para el desarrollo e implementación de éstos prototipos. Consideramos importante mencionarlos puesto que la elección de los prototipos a realizar así como la manera de implementarlos dependieron de manera muy importante tanto de las plataformas físicas como lógicas que tenemos a nuestra disposición. Así por ejemplo, adquirir un SDK (*Software Development Kit*) especializado, producido por la misma compañía [7] que fabrica las tarjetas inteligentes que usamos, tiene un costo de aproximadamente $399 USD. Este precio resultó simplemente costoso para los propósitos de ésta investigación.

5.2.1. Hardware

Tabla 5.1. Nuestros recursos de Hardware

Nombre	Descripción	Papel
allzee	i686 AMD Athlon(TM) XP 3200+ AuthenticAMD	Servidor NFS, Terminal de desarrollo y pruebas.
minibrit	i686 Pentium III (Coppermine) GenuineIntel	Estación de trabajo hipotética.
cad.cele.unam.mx	i686 Intel(R) Pentium(R) 4 CPU 2.80GHz GenuineIntel	Repositorio CVS, Servidor de Bases de Datos.
plumbago	iBook PowerPC 750 (2.2) 700MHz Power Macintosh	Terminal de desarrollo y pruebas.
Gemplus GemPC Twin	Lector que cumple ISO/IEC 7816-1,2,3,4 y el estándar CCID (Chip Card Interface Device 1.0.)	Lector de tarjetas inteligentes con interfaz USB y RS232.
Gemplus GemPC433-SL	Lector que cumple ISO/IEC 7816-1,2,3,4 y el estándar CCID	Lector de tarjetas inteligentes con interfaz USB.

[7] Axalto, un compañía líder en la producción de soluciones con tarjetas inteligentes. http://www.axalto.com

Nombre	Descripción	Papel
	(Chip Card Interface Device 1.0.)	
Axalto Cryptoflex 32K	EEPROM 32Kbytes, Coprocesador criptográfico, ISO 7816, protocolo ISO T=0. Cumple con las especificaciones PC/SC y PKCS#11.	*Token criptográfico* para guardar llaves privadas y certificados RSA.

5.2.2. Software

El software que utilizamos es principalmente software libre. Las razones para utilizar este tipo de software han sido ya expuestas en el capítulo 3. La siguiente tabla es una descripción del software que hemos utilizado en el desarrollo de ambos prototipos. El software específico de cada prototipo está descrito en la sección correspondiente. A menos que se indique lo contrario, el software que se muestra aquí es libre y se puede descargar gratuitamente de Internet.

Tabla 5.2. Software utilizado en ambos prototipos

Nombre y versión	Función
GNU/Linux 2.6.5-gentoo	Sistema operativo Linux y entorno de desarrollo utilizado para la programación del prototipo.
Libusb 0.1.8	Esta biblioteca permite el acceso de usuario a dispositivos USB. (en nuestro caso el lector de tarjetas inteligentes). [LibUSB]
pcsc-lite 1.2.9-beta6.	Capa de lector del estándar PC/SC. Este es el software que hace uso de un controlador específico para poder usar el lector a bajo nivel.
USB CCID IFD Handler 0.9.2	Chip/Smart Card Interface Devices Driver. Controlador específico para los lectores que usamos.
opensc 0.9.2	Libopensc es una biblioteca para acceder a tarjetas inteligentes compatibles con ISO 7816-4. Es posible hacer uso de servicios criptográficos en tarjetas que implementen PKCS#11 y 15.
Linux-PAM	Linux-PAM es una implementación libre del estándar DCE-RFC 86.0 de SunSoft. PAM es un sistema de bibliotecas que maneja las tareas de autentificación en muchos sistemas UNIX. PAM provee una manera de desarrollar programas que sean independientes de los mecanismos de autentificación.

Nombre y versión	Función
OpenSSL 0.9.7c	OpenSSL es un conjunto de herramientas criptográficas que implementan los protocolos de red SSL (*Secure Socket Layer*) y TLS (*Transport Layer Security*), así como los estándares criptográficos requeridos por éstos.

5.3. Instalación del software común a ambos prototipos

5.3.1. Capa del lector: pcsc - lite

Como ya se explicó con más detalle en secciones anteriores, pcsc-lite representa la capa de acceso al lector o terminal donde pretendemos usar las tarjetas inteligentes. Es necesario pues, instalar éste software tanto en el servidor (donde se personalizarán las tarjetas y se emitirán los certificados), como en cada una de las terminales donde se implementarán los prototipos.

Al momento de escribir este documento, la última versión disponible de pcsc-lite en el *Portage* [8] de Gentoo Linux es la 1.2.0. Es necesario pues, descargar la última versión disponible del software (1.2.9-beta6). Una vez obtenido el archivo de la distribución de pcsc-lite, en nuestro caso específico, `pcsc-lite-1.2.9-beta6.tar.gz` es necesario desempaquetarlo, compilarlo e instalarlo:

```
$ tar -zxvf pcsc-lite-1.2.9-beta6.tar.gz
$ cd pcsc-lite-1.2.9-beta6/
$ ./configure
$ make
$ su -c "make install"
```

Una vez hecho ésto, será necesario descargar el código fuente del driver que utilizan nuestros lectores. El controlador USB CCID IFD Handler 0.9.2 es parte del proyecto M.U.S.C.L.E. y puede ser descargado de la misma página que pcsc-lite.

Un prerequisito para usar dispositivos CCID-USB (como nuestros lectores) sobre Linux, es la librería libusb. Su instalación resulta trivial:

```
# emerge libusb
```

[8] Más información acerca de Gentoo Linux y su sistema de administración Portage puede encontrarse en Gentoo Linux Wiki http://gentoo-wiki.com/.

Hecho esto, desempaquetamos el driver, lo compilamos y lo instalamos en su directorio de destino predeterminado.

```
$ tar -zxvf ccid-0.9.2.tar.gz
$ cd ccid-0.9.2
$ ./configure
$ make
$ su -c "make install"
```

5.3.2. Capa de acceso a tokens: OpenSC

Falta un paso más antes de comenzar con los prototipos. Es necesario instalar OpenSC para tener acceso a los tokens criptográficos que estaremos usando. De nuevo, al momento de escribir este documento, la versión de OpenSC que se encuentra en el *Portage* de Gentoo Linux no es la última. Es necesario pues, descargar la última versión del software y llevar a cabo un proceso similar al anterior. Aquí, es necesario indicarle al script de configuración la ruta donde pcsc-lite se encuentra instalado:

```
$ tar -zxvf opensc-0.9.2.tar.gz
$ cd opensc-0.9.2
$ ./configure --with-pcsclite=/usr/local/lib/
$ make
$ su -c "make install"
```

Con esto, tenemos ya listas todas las bibliotecas dinámicas y ejecutables que usaremos en ambos prototipos del proyecto IDI.

5.4. Prototipo 1: Autentificación centralizada en estaciones de trabajo Unix

5.4.1. Antecedentes y Requerimientos

En un centro de cómputo donde hay más usuarios que estaciones de trabajo se hace necesaria el uso de sistemas operativos multiusuario. Uno de ellos, que además está típicamente vinculado a la ingeniería es el sistema operativo UNIX. Uno de los problemas más importantes al administrar este tipo de sistemas operativos es la autentificación de múltiples usuarios en múltiples máquinas. Algunas de las preguntas típicas que surgen en los administradores de sistemas son:

- ¿Cómo autentificar **al mismo usuario** en diferentes estaciones de trabajo?
- ¿Cuantas bases de datos con información del usuario se deben mantener?
- ¿Dónde se deben guardar los archivos de los usuarios?

La experiencia nos muestra que una sola base de datos donde se guarde la información de la autentificación es mucho mejor mantener muchas bases de datos (típicamente, cada sistema Unix mantiene su propia base de datos en `/etc/passwd`). Las razones de esto son obvias: primero, para mantener la consistencia, los administradores tendrían que llevar a cabo la misma operación en todas las estaciones de trabajo en cada cambio de información de usuarios, tal como dar de alta un nuevo usuario, dar de baja uno viejo, o cambiar un password. Además, el riesgo de cometer errores en cuanto a la administración de los sistemas es mucho más grande cuando se tienen muchas bases de datos, y por lo tanto los riesgos de seguridad aumentan.

Desde el punto de vista de los usuarios, también deseable que en un laboratorio de cómputo se maneje siempre la misma información. Es especialmente importante que los usuarios tengan acceso a sus archivos en cualquier estación de trabajo, ya que no se puede esperar que los usuarios puedan utilizar siempre el mismo equipo.

Es necesario pues, un sistema de "login unificado". Esta necesidad es tan común que muchas empresas de computación han abordado el problema proporcionando diferentes soluciones que, aunque pretenden ser genéricas, siempre se tienen que adaptar a las necesidades específicas del centro de cómputo del que se trata, de manera que nunca se proporciona un software que soluciona el problema, sino más bien un conjunto de herramientas que permiten a los administradores llevar a cabo ésta tarea. Así, este prototipo tuvo las siguientes metas:

1. Establecer un sistema de autentificación de factor dual para cada usuario.
2. Almacenar la información relevante para la autentificación **en un host único**
3. Hacer uso de tokens criptográficos para el almacenamiento de los secretos de autentificación
4. Permitir a los usuarios acceder a sus archivos desde cualquier estación de trabajo

5.4.2. Descripción general.

Usamos OpenSSL para crear una *autoridad certificadora*. Esta autoridad certificadora firma una llave privada y un certificado que sirven para identificar a un usuario. Una vez hecho ésto usamos OpenSC y pcsc-lite para acceder a una tarjeta inteligente, donde se almacenarán permanentemente esta llave privada y este certificado. Después de esto, ambos archivos se guardan en un lugar seguro para propósitos de respaldo. OpenSC nos provee de un módulo de PAM que cada estación de trabajo Unix usa junto con un lector de tarjetas inteligentes para realizar la autentificación a nivel local, sin embargo, los directorios *$HOME* de los usuarios están montados en el sistema de archivos local usando NFS, de manera que el almacenamiento físico de los certificados así como de los demás archivos de usuario se lleva a cabo en un servidor remoto.

La seguridad del sistema se analiza desde el punto de vista del tráfico de información en la red, de la dificultad de violar el sistema de autentificación, y de los riesgos inherentes a la forma de compartir los archivos.

5.4.3. Instalación y configuración

A continuación se describen a detalle los pasos necesarios para implementar el sistema.

5.4.3.1. La red

El primer paso para la creación de un sistema de login unificado es por su puesto la instalación y puesta en marcha de la red en donde nos interesa proporcionar dicho servicio. La red que usamos fue una de protocolo de Internet (TCP/IP) sobre Ethernet. Se trata de una red tipo C. Esta es la lista de hosts:

Tabla 5.3. Direcciones IP de los hosts utilizados

Nombre	Dirección IP
Alizee	192.168.0.1
Plumbago	192.168.0.2
Minibrit	192.168.0.3

La máscara de red usada es de 24 bits. Las tres computadoras se conectan a un *hub*

por medio de cables Ethernet. Esta configuración, si bien simple, también es típica de los laboratorios de cómputo en la Facultad de Ingeniería.

El siguiente es un diagrama general de la red que construimos para nuestro prototipo:

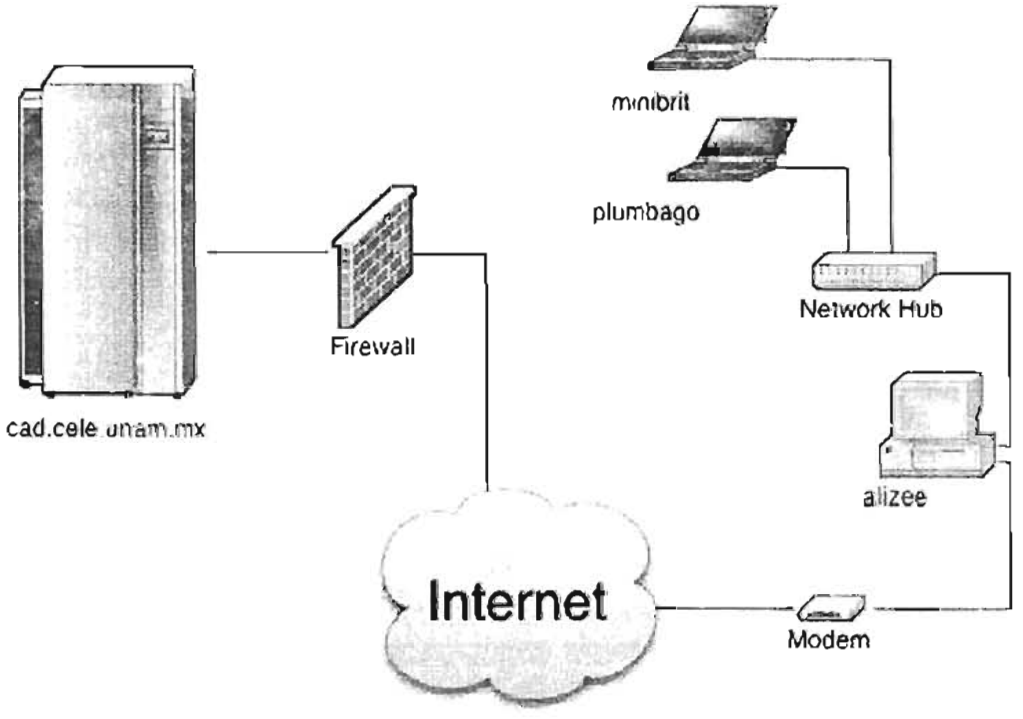

Figura 5.1. Diagrama general de nuestra red.

5.4.3.2. El servidor

El servidor se refiere al equipo de la red que albergará los archivos de los usuarios y los certificados RSA que servirán para autentificar. También se trata de la puerta de enlace de nuestra red. En esta máquina llevamos a cabo la creación de la autoridad certificadora y la instalación de un servidor NFS. También utilizamos ésta computadora para llevar a cabo la personalización de las tarjetas inteligentes.

5.4.3.2.1. Llaves y certificados con OpenSSL

Para crear la autoridad certificadora y todos los archivos criptográficos de los usua-

rios, usamos OpenSSL. La instalación de dicha herramienta en el sistema operativo que usamos (Gentoo Linux) es trivial:

```
alizee root # emerge openssl
```

Una vez terminado el proceso de instalación, procedemos a crear las llaves RSA y los certificados X.509 [STRUCK], [OpenSC]. Estos certificados serán guardados en la tarjeta junto con la llave privada. Si la instalación fue exitosa, el script `/etc/ssl/misc/CA.sh` proporciona la manera más fácil de crear nuestra autoridad certificadora. Durante el proceso, es necesario elegir un *passphrase* que protegerá a la autoridad certificadora.

```
alizee root # /etc/ssl/misc/CA.sh -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to './demoCA/private/./cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:mx
State or Province Name (full name) [Some-State]:Distrito Federal
Locality Name (eg, city) []:Mexico
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UNAM
Organizational Unit Name (eg, section) []:nibble.um
Common Name (eg, YOUR name) []:nibble security team
Email Address []:samuelflores@gmail.com
```

Después de ésto, el directorio `.demoCA/` contendrá todos los archivos de la autoridad certificadora:

```
alizee root # ls .demoCA/
cacert.pem  crl        index.txt.old  private  serial.old
```

```
certs       index.txt  newcerts       serial
```

Estos archivos tienen una validez predeterminada de un año. Ese tiempo probablemente no sea suficiente para nosotros, por lo que aumentamos esa validez a diez años:

```
alizee root # openssl x509 -in demoCA/cacert.pem -days 3650 \
-out demoCA/cacert.pem -signkey demoCA/private/cakey.pem
Getting Private key
Enter pass phrase for demoCA/private/cakey.pem:
```

una vez hecho esto podemos ver los valores del certificado de nuestra Autoridad:

```
alizee root # openssl x509 -in demoCA/cacert.pem -text
Certificate:
Data:
...
Validity
Not Before: Oct 23 20:18:54 2004 GMT
Not After : Oct 21 20:18:54 2014 GMT
...
```

El siguiente paso es generar una solicitud de certificado **para cada uno de nuestros usuarios.** Por cada una de estas solicitudes se nos pedirá un passphrase que el administrador guardará en un lugar seguro.

```
alizee root # /etc/ssl/misc/CA.sh -newreq
Generating a 1024 bit RSA private key
...++++++
.................++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) [AU]:mx
State or Province Name (full name) [Some-State]:Distrito Federal
Locality Name (eg, city) []:Mexico
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UNAM
Organizational Unit Name (eg, section) []:nibble.um
Common Name (eg, YOUR name) []:nibble security team
Email Address []:samuelfloress@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
```

Esto generó una llave privada cifrada con el passphrase y una solicitud de certificado en el archivo `newreq.pem`. Esta llave privada es la que se almacenará en nuestra tarjeta, pero antes tendremos que descifrarla:

```
alizee root # openssl rsa -in newreq.pem -out newkey.pem
Enter pass phrase for newreq.pem:
writing RSA key
```

Ahora, firmamos la solicitud de certificado usando nuestra autoridad certificadora:

```
alizee root # /etc/ssl/misc/CA.sh -sign
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
... clip ...
Certificate is to be certified until Oct 23 19:04:27 2005 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
... clip ...
-----END CERTIFICATE-----
Signed certificate is in newcert.pem
```

Después de este proceso tendremos la llave descifrada en `newkey.pem` y el certificado firmado por nuestra autoridad certificadora en `newcert.pem`. Un proceso similar es necesario para cada uno de los usuarios que pretendamos tener.

5.4.3.2.2. Personalización de las tarjetas Cryptoflex 32K

La personalización de los tokens criptográficos que usamos en nuestros prototipos se hizo con la herramienta pkcs15-init. Este programa usa la biblioteca `libopensc.so` para acceder a tarjetas inteligentes que cumplan con el estándar pkcs15. Los pasos a seguir se describen a continuación.

Inicializar la tarjeta según la estructura marcada por el estándar pkcs15, borrando toda la tarjeta antes de hacerlo (-E) y pidiendo la llave de transporte (-T):

```
alizee root # pkcs15-init -E -T --create-pkcs15 --profile pkcs15
Please enter transport key:
New Security Officer PIN (Optional - press return for no PIN).
Please enter Security Officer PIN:
Please type again to verify:
Unblock Code for New User PIN (Optional - press return for no PIN).
Please enter User unblocking PIN (PUK):
Please type again to verify:
```

Se introduce las llave de transporte que permite desbloquear la tarjeta. Después se introduce un PIN y un PUK para que identificarán al administrador (security officer). Después de hecho ésto, será necesario el PIN del administrador para cualquier cambio importante en la tarjeta.

Este comando guardará el PIN de un usuario en la primera posición libre:

```
alizee root # pkcs15-init --auth-id 1 --store-pin --label "Samuel Flores"
New User PIN.
Please enter User PIN:
Please type again to verify:
Unblock Code for New User PIN (Optional - press return for no PIN).
Please enter User unblocking PIN (PUK):
Please type again to verify:
Security officer PIN required.
Please enter Security officer PIN:
```

Como se ve, es necesario que el administrador presente su PIN para poder almacenar el del usuario. Es necesario que el usuario recuerde su PIN, ya que no hay ninguna manera en que el administrador pueda cambiarlo.

Después almacenamos la llave privada asociada a ese PIN y usuario:

```
alizee root # pkcs15-init --auth-id 1 --key-usage sign,decrypt \
--store-private-key newkey.pem
Security officer PIN required.
Please enter Security officer PIN:
```

Una vez que la llave está guardada en la tarjeta, es imposible que el administrador la pueda extraer, sin conocer el PIN del usuario.

Por último, se guarda el certificado del usuario, firmado por nuestra autoridad certificadora:

```
alizee root # pkcs15-init --auth-id 1 --store-certificate newcert.pem
Security officer PIN required.
Please enter Security officer PIN:
```

Esto finaliza la personalización de las tarjetas criptográficas que usamos. Es labor del administrador guardar con sumo cuidado todos los archivos de los usuarios con propósitos de respaldo en un lugar seguro (con excepción de la llave privada `newkey.pem` la cual sería mejor borrar por completo). Los secretos para acceder a la tarjeta (PIN y PUK) son indispensables para recuperar dicha llave privada. Sin el pin la tarjeta quedará inservible.

5.4.3.2.3. Compartiendo archivos con NFS

Como se explicó en el capítulo 3, NFS nos permite montar sistemas de archivos remotos a través de una red IP. Mantener los directorios `$HOME` de los usuarios accesibles por red desde un solo host tiene dos efectos importantes:

- Permite mantener los certificados relevantes para la autentificación en un host único
- Permite acceso a los archivos de usuario en cualquier estación de trabajo

La instalación de NFS en Gentoo Linux es trivial:

alizee root # emerge -p nfs-utils

esto instalará los demonios de cliente y servidor de NFS. Hecho esto, será necesario llevar a cabo la configuración del servidor. Para ésto será necesario editar como mínimo 3 archivos: `hosts.deny`, `hosts.allow` y `exports`. Los pasos que segui-

mos fueron los siguientes:

5.4.3.2.3.1. Editar `/etc/exports`

Este archivo contiene la lista de directorios que serán compartidos por el servidor. Una entrada típica de `/etc/exports` se ve así:

Ejemplo 5.1. Entrada típica en `/etc/exports`

```
directorio máquina1(opción11,opción12) máquina2(opción21,opción22) ...
```

En donde:

Directorio.

Es el directorio por compartir. Éste puede ser toda una partición, aunque no es necesario. Todos los subdirectorios también serán compartidos.

MáquinaX.

Son los nombres o direcciones IP de los hosts cliente que tendrán acceso al directorio. Usar direcciones IP es más seguro pues disminuye el riesgo de *phishing* en el DNS.

OpciónYZ.

Se trata de opciones que describen el tipo de acceso que cada máquina tendrá al directorio. Algunas opciones importantes: **ro** para sólo lectura (predeterminado), o **rw** (para lectura y escritura)

En nuestro prototipo, el archivo `/etc/exports` se encuentra así:

Ejemplo 5.2. Archivo `/etc/exports` en alizee

```
# /etc/exports: NFS file systems being exported.  See exports(5).
/home        192.168.0.3(rw)
```

5.4.3.2.3.2. Editar `/etc/hosts.deny` y `/etc/hosts.allow`

Estos archivos no pertenecen directamente a NFS, sino más bien al programa *tcpd* (TCP Wrappers). Este programa "envuelve" a todas las conexiones TCP entrantes de manera que podemos tener cierto control sobre ellas antes de que xinetd pueda pasar el control a algún *demonio* o servicio. Este cierto control se basa en información de la conexión, específicamente en su fuente y su destino (quién se conecta y qué servicio quiere). Además de ésto, nos proporciona registros detallados de la conexión. En general, el archivo `hosts.deny` establece las políticas generales para aceptar conexiones, mientras `hosts.allow` permite hacer excepciones a éstas políticas.

Explicar el funcionamiento detallado de TCP Wrappers está fuera del alcance de éste documento, así que nos enfocaremos a la configuración que usamos para garantizar la seguridad mínima de éste prototipo. Un análisis de la seguridad será hecho más adelante. Por lo pronto, en nuestro archivo `/etc/hosts.deny` negamos las conexiones a todo mundo:

Ejemplo 5.3. Archivo `/etc/hosts.deny` en alizee

```
ALL:      PARANOID
ALL:      ALL
portmap:  ALL
lockd:    ALL
mountd:   ALL
rquotad:  ALL
statd:    ALL
```

Hecho ésto, es necesario permitir el acceso de nuestras dos estaciones de trabajo hipotéticas. Ésto se hace a través del archivo `/etc/hosts.allow`:

Ejemplo 5.4. Archivo `/etc/hosts.allow` en alizee

```
sshd:     ALL
portmap: 192.168.0.2, 192.168.0.3
lockd:    192.168.0.2, 192.168.0.3
rquotad: 192.168.0.2, 192.168.0.3
mountd:  192.168.0.2, 192.168.0.3
statd:    192.168.0.2, 192.168.0.3
```

Hecho esto tenemos lista la configuración básica del servidor NFS. No nos queda más que iniciar el servidor y agregarlo al *runlevel* predeterminado, para que inicie con el servidor:

```
alizee root # /etc/init.d/nfs start
alizee root # rc-update add nfs default
```

5.4.3.3. Los clientes

Ahora veamos la configuración necesaria del lado de cada estación de trabajo.

5.4.3.3.1. Montando directorios del servidor

Un prerequisito para poder acceder a nuestro servidor NFS es que el núcleo del sistema operativo tenga provea dicha funcionalidad, y que el programa *mount* sea relativamente reciente [BARR-LANGFELDT-2002]. El primer paso es instalar el cliente de NFS que nos permitirá hacer uso de los directorios compartidos por el servidor. La instalación del cliente NFS en Gentoo Linux es trivial:

```
minibrit root # emerge -p nfs-utils
```

Esto instalará tanto los programas del cliente como los del servidor de nfs en nuestra estación de trabajo. Como a nosotros simplemente nos interesa la funcionalidad de cliente, únicamente necesitamos agregar los scripts de inicio de portmap, lockd y statd al *runlevel* adecuado. En nuestro caso, los agregamos al runlevel *default* para que inicie normalmente junto con todo el sistema:

minibrit root # rc-update add nfs-utils default

Podemos probar la instalación de nuestro servidor montando el sistema de archivos compartido:

```
minibrit root# mount alizee:/home /home
minibrit root# cd /home
minibrit root# touch prueba.txt
minibrit root# umount /home
```

Después tendremos que editar el archivo `/etc/fstab` para que los sistemas de archivos se monten en el momento de inicio del sistema.

Ejemplo 5.5. Archivo `/etc/fstab` en la estación de trabajo minibrit

```
/dev/hda2           /           reiserfs   noauto,noatime,notail   0 1
alizee:/home        /home       nfs        ro,hard,intr            0 0
/dev/hda4           none        swap       sw                      0 0
/dev/cdroms/cdrom0  /mnt/cdrom  iso9660    noauto,ro,user          0 0
/dev/fd0            /mnt/floppy auto       noauto,user             0 0
none                /proc       proc       defaults                0 0
none                /dev/shm    tmpfs      defaults                0 0
```

5.4.3.3.2. PCSCD

Como vimos en la sección 3.4.13 y 5.3.1, pcsc-lite nos proporciona la capa de acceso al lector. Es necesario pues, iniciar el *demonio* `pcscd`. Con el objetivo de iniciar el demonio automáticamente junto con todo el sistema, editamos el archivo `/etc/conf.d/local.start`, el cual se ejecutará después de todos los scripts de un determinado *runlevel* en Gentoo Linux. En nuestro caso:

Ejemplo 5.6. Archivo `/etc/conf.d/local.start` en minibrit:

```
# This is a good place to load any misc.
# programs on startup ( 1>&2 )
/usr/local/sbin/pcscd
```

Esto deja al sistema listo para acceder a nuestro lector por medio de cualquier aplicación.

5.4.3.3.3. Configurando PAM

Linux-PAM es una implementación libre del estándar DCE-RFC 86.0 de SunSoft. En términos generales, PAM (*Pluggable Authentication Modules*), es un sistema de bibliotecas que maneja las tareas de autentificación en la mayoría de los sistemas UNIX. Como su nombre lo indica, PAM provee *módulos* de autentificación que son independientes de las aplicaciones que requieren la dicha autentificación, de manera que los programadores puede hacer uso de éstas para desarrollar programas que sean independientes de los mecanismos de autentificación. Por último, éstas mismas interfaces permiten a los administradores de sistemas cambiar las políticas de autentificación y acceso a diferentes recursos del sistema, sin tener que modificar en nada las aplicaciones.

En la sección 5.3.2, vimos cómo instalar OpenSC. Dicho software también nos proporciona el módulo de PAM que utilizaremos para hacer la autentificación en cada estación de trabajo. Ahora es necesario modificar los archivos de configuración de PAM para que usen el módulo de OpenSC [MORGAN-2002]. Desde la versión 0.56, PAM requiere un archivo de configuración para cada programa que requiera de su servicio de autentificación (en vez de uno solo donde se configuran todos los servicios). Para esta investigación, los archivos relevantes son dos: `/etc/pam.d/login` y `/etc/pam.d/check_user`. El primero se trata de un archivo de estándar del sistema, que servirá para el prototipo de login centralizado, el segundo es creado específicamente para nuestro prototipo de sistema de consulta de historiales académicos. La sintaxis de un archivo de configuración de PAM está dada por:

Ejemplo 5.7. Sintaxis de un archivo `nombre-del-servicio` de configuración de PAM

```
tipo-de-módulo   bandera-de-control   ruta-al-módulo   argumentos
```

Un poco de explicación: `nombre-del-servicio` es, de manera poco sorprendente, el nombre que identifica a la aplicación que requiere de servicios de autentificación. Típicamente el servicio lleva el mismo nombre que la aplicación. Como veremos más adelante, el manejo de éste nombre en nuestras aplicaciones es de suma

importancia para evitar una potencial *degradación de restricciones* [MORGAN-2001].

En cuanto al *tipo-de-módulo* éste puede ser de alguno de éstos cuatro tipos:

- *auth*; este tipo de módulo provee dos aspectos de la autentificación del usuario. Primero, establece métodos para asegurarse de que el usuario es quien dice ser. Segundo, el módulo establece privilegios (como la pertenencia a un grupo) a través de sus *credenciales*.
- *account*; este tipo de módulo realiza administración de la cuenta sin basarse en la autentificación (por ejemplo, para restringir la entrada de usuarios a ciertas horas del día, etc).
- *session*; este tipo de módulo esta asociado a las acciones que se deben realizar antes o después de que al usuario se le otorgue el servicio (por ejemplo, registrar la entrada o montar directorios).
- *password*; este último tipo sirve para establecer la política del cambio de credenciales por el usuario.

La *bandera-de-control* se refiere a la manera en que PAM reaccionará al éxito o fracaso del módulo asociado dicha bandera. Dado que los módulos pueden estar apilados, las banderas de control indican la importancia relativa entre ellos. La bandera puede ser alguna de las siguientes:

- *required* si el servicio es requerido pero no suficiente, regresando el control al programa hasta que todas las pruebas hayan terminado.
- *requsite* es similar a required, excepto en que regresa inmediatamente el control a la aplicación en caso de fallo.
- *sufficient*, si el éxito en dicho módulo satisface a PAM para que otros módulos ni siquiera sean examinados.
- *optional* si se trata de un módulo con alguna función que no sea critica para proporcionar o no el servicio.

Hecha ésta explicación, mostramos a continuación el contenido del archivo `/etc/pam.d/login` que controla la política de autentificación del programa **Login** en

nuestras estaciones de trabajo. Login es un programa en modo texto muy usado en sistemas Unix para proporcionar acceso a usuarios locales. Existen muchos otros programas para que requerirán autentificación, por ejemplo GDM, XDM, KDM, su, o xscreensaver.

Ejemplo 5.8. Archivo `/etc/pam.d/login`

```
#%PAM-1.0
auth       required     /lib/security/pam_securetty.so
auth       required     /lib/security/pam_opensc.so
auth       required     /lib/security/pam_nologin.so

account    required     /lib/security/pam_opensc.so

password   required     /lib/security/pam_deny.so

session    required     /lib/security/pam_opensc.so
session    optional     /lib/security/pam_console.so
```

Hecho esto, tendremos listo un sistema de login centralizado. No nos queda más que mostrar una pequeña prueba del sistema y hacer un breve análisis de su seguridad.

5.4.4. Probando el funcionamiento.

El funcionamiento del sistema una vez configurado es muy sencillo. Primero, el servidor NFS debe estar accesible desde nuestra red todo el tiempo. Es necesario que sea éste el *host* que inicie primero. Las estaciones de trabajo (nuestros clientes nfs) iniciarán `pcscd` de inmediato, habilitando los lectores de tarjetas. Ésto se puede verificar viendo las bitácoras del sistema. En nuestro caso, usamos *syslog-ng* que guarda los registros en `/var/log/messages`. Así, para ver las bitácoras ejecutamos:

minibrit root # tail -f -n 30 /var/log/messages

Ejemplo 5.9. Mensajes del sistema en minibrit durante el *boot time*

```
...
Feb 26 16:13:13 minibrit usb 1-1:
 new full speed USB device using address 2
```

```
Feb 26 16:08:57 minibrit pcscd: pcscdaemon.c:440:
  main pcsc-lite 1.2.9-beta6 daemon ready.
Feb 26 16:13:13 minibrit pcscd:
  readerfactory.c:1055:RFInitializeReader \
  Attempting startup of GemPC Twin 00 00.
Feb 26 16:13:14 minibrit pcscd:
  readerfactory.c:929:RFBindFunctions Loading IFD Handler 3.0
Feb 26 16:13:14 minibrit pcscd: ifdhandler.c:67:
  IFDHCreateChannelByName lun: 0, device: usb:08e6/3437:libusb:001:002
Feb 26 16:13:14 minibrit pcscd:
  ccid_usb.c:199:OpenUSBByName \
  Manufacturer: Ludovic Rousseau (ludovic.rousseau@free.fr)
Feb 26 16:13:14 minibrit pcscd:
  ccid_usb.c:209:OpenUSBByName ProductString: Generic CCID reader v0.9.2
Feb 26 16:13:14 minibrit pcscd:
  ccid_usb.c:215:OpenUSBByName Copyright: This driver is protected by terms \
  of the GNU General Public License version 2, \
  or (at your option) any later version.
```

Estos mensajes corresponden al inicio normal de pcscd al iniciar el lector GemPC Twin. Todas las terminales deberían mostrar mensajes parecidos. Por otra parte, del lado del servidor es posible revisar los mensajes de nfs trabajando. En este caso:

Ejemplo 5.10. Mensajes del kernel en alizee al compartir un directorio

```
Feb 26 16:08:44 alizee portmap[9815]:
  user rpc not found, reverting to user bin
Feb 26 16:08:45 alizee rpc.statd[9832]]:
  Version 1.0.6 Starting
Feb 26 16:08:45 alizee exportfs[9835]]:
  /etc/exports [2]: No 'sync' or 'async' option specified \
  for export "192.168.0.3:/home". Assuming default behaviour ('sync').
Feb 26 16:13:49 alizee rpc.mountd:
  authenticated mount request from minibrit:981 for /home (/home)]
```

Un mensaje similar a este último debe aparecer por cada estación de trabajo conectándose al servidor. Por último, veamos las bitácoras del sistema al autentificar a un usuario:

Ejemplo 5.11. Mensajes del kernel en minibrit al autentificar un usuario

```
Feb 26 16:27:41 minibrit pcscd:
  ifdhandler.c:675:IFDHPowerICC lun: 0
Feb 26 16:27:41 minibrit pcscd:
  eventhandler.c:407:EHStatusHandlerThread Card inserted into GemPC Twin 00 00
Feb 26 16:27:41 minibrit pcscd:
  Card ATR: 3B 95 18 40 FF 62 04 01 01 05
...
Feb 26 16:27:54 minibrit pcscd:
  winscard.c:121:SCardConnect Attempting Connect to GemPC Twin 00 00
Feb 26 16:28:04 minibrit pcscd:
  winscard.c:1361:SCardTransmit Send Protocol: T=0
...
Feb 26 16:28:05 minibrit login(pam_opensc)[7762]:
  Authentication successful for cerealito at /dev/vc/1.
Feb 26 16:28:05 minibrit login(pam_opensc)[7762]:
  session opened for user cerealito by root(uid=0)
...
Feb 26 16:28:11 minibrit pcscd:
  eventhandler.c:337:EHStatusHandlerThread Card Removed From GemPC Twin 00 00
```

En éstos mensajes se puede ver la información de la tarjeta (ATR), el protocolo de comunicación y el módulo de PAM que se está usando (pam_opensc). Finalmente, login entrega una sesión al usuario.

Desde el punto de vista del usuario, el proceso es bastante transparente: el usuario deberá insertar su tarjeta emitida por la autoridad certificadora en el lector de cualquier estación de trabajo que haya sido configurada, e ingresar su *nombre de usuario* y el PIN que protege a su tarjeta. Si la persona posee la tarjeta correcta y el PIN correcto, el sistema le entregará una sesión. Los archivos del usuario serán los mismos en cualquier estación de trabajo.

5.4.5. Análisis y Conclusión

Por supuesto, las pruebas al sistema que mostramos en éste prototipo son significativas sólo en cuanto a la funcionalidad mínima. Pruebas de implementación **real** requerirían de muchos más recursos, los cuales no tenemos. Consideramos que un sistema de éste tipo, requiere indispensablemente un periodo de pruebas en ambiente de producción, antes de que se le confíen sistemas de misión crítica. Esto evidentemente es imposible para los propósitos y alcances de ésta investigación. Aún así consideramos muy importante el siguiente análisis:

5.4.5.1. Factibilidad

El primer problema que viene a la mente de cualquier administrador de sistemas al ver éste modelo es probablemente el tráfico de red y la carga en los discos duros. Ninguno de éstos dos problemas puede ser resuelto usando NFS con un sólo servidor. El número óptimo de sistemas de archivos compartidos, discos duros y servidores, así como la mejor distribución de la red, sólo podrán ser determinados con pruebas en un ambiente de producción. Con éste prototipo no pretendemos solucionar un problema específico de un laboratorio de cómputo en particular, sino sentar las bases para desarrollos posteriores.

5.4.5.2. Seguridad

La seguridad del prototipo mostrado se basa en dos pilares. El primero es por su puesto uno mismo de sus objetivos: **la inclusión de criptografía de llave pública (RSA) en la autentificación de usuarios y la implementación de un esquema de autentificación de factor dual**. Ésto hace de nuestro prototipo un sistema de login centralizado bastante más seguro que cualquier otro sistema basado en contraseñas (LDAP, NIS, SAMBA, etc) las ventajas de un sistema con autentificación de factor dual ya han sido discutidas en los capítulos 2 y 3.

Desgraciadamente, esta ventaja se encuentra únicamente a nivel de la autentificación, y la seguridad de un sistema de cómputo debe ser analizada desde muchos otros puntos de vista: ¿Existen formas diferentes de acceder al sistema, omitiendo o esquivando la autentificación de factor dual? (*backdoors*) ¿Qué otros ataques son posibles aparte del acceso no autorizado a un shell? (Denegación del servicio [*Denial of Service*], impostura [*phising, spoofing*], captura de datos en la red [*sniffing*], etc...)

El segundo pilar de seguridad es la inclusión de TCP wrappers para restringir el acceso a los directorios NFS. NFS no ha sido ni será jamás un ejemplo de una aplicación diseñada con la seguridad como objetivo primordial. A lo largo de su historia, NFS ha sido objetivo de múltiples ataques debido a las diversas vulnerabilidades encontradas en su código [BARR-LANGFELDT-2002]. Aún así, NFS sigue siendo una herramienta muy usada en nuestra universidad y en el mundo para compartir sistemas de archivos en Unix. En este sentido, TCP wrappers provee una capa de seguridad que resulta absolutamente indispensable y mínima. Incluso en un laboratorio de cómputo donde se establecieran mecanismos de seguridad adicionales, TCP wrappers debería estar siempre como una primera línea de protección.

En cuanto a la privacidad de los archivos del usuario, NFS transmite los archivos por la red sin cifrado alguno. Ésto introduce una desventaja con respecto a las terminales tontas donde todo el procesamiento se hace en un solo *host*, pero el usuario tiene control total sobre el nivel de acceso a sus archivos. Afortunadamente, nuestro prototipo evita la transmisión por red de contraseñas o llaves privadas (el intercambio de secretos de autentificación se lleva a cabo en cada host) lo único que viaja por red es el certificado RSA X.509, de donde es imposible extraer una llave privada.

Como conclusión, es necesario decir que la seguridad y la factibilidad de éste sistema es suficientemente buena para un prototipo. Una mejora que habría que considerar para una implementación en ambientes de producción es la inclusión de Secure Shell (SSH) para cifrar los datos de NFS haciendo un *túnel* (SNFS).

5.5. Prototipo 2: Sistema de Consulta de historiales académicos (SiCo)

5.5.1. Características

5.5.1.1. Alcances

El objetivo principal de la aplicación es mostrar de manera práctica y sencilla el funcionamiento del modelo de identificación basado en tarjetas criptográficas planteado en el capítulo 4, además de servir como base para facilitar el desarrollo de aplicaciones más robustas.

5.5.1.2. Funcionamiento

Descripción del funcionamiento de la aplicación.

El usuario interesado en realizar la consulta de su historial ejecutará la aplicación en una terminal configurada para utilizar el modelo de autentificación implementado en el prototipo anterior, introducirá su identificación inteligente previamente personalizada y tecleará su PIN. El sistema entonces realizará primero la autentificación del usuario, y si es un usuario válido mostrará el historial correspondiente.

Utilización de la aplicación.

Al iniciar la aplicación, el usuario de encontrará con una pantalla como al que se

muestra en la Figura 5.2, "Pantalla de inicio".

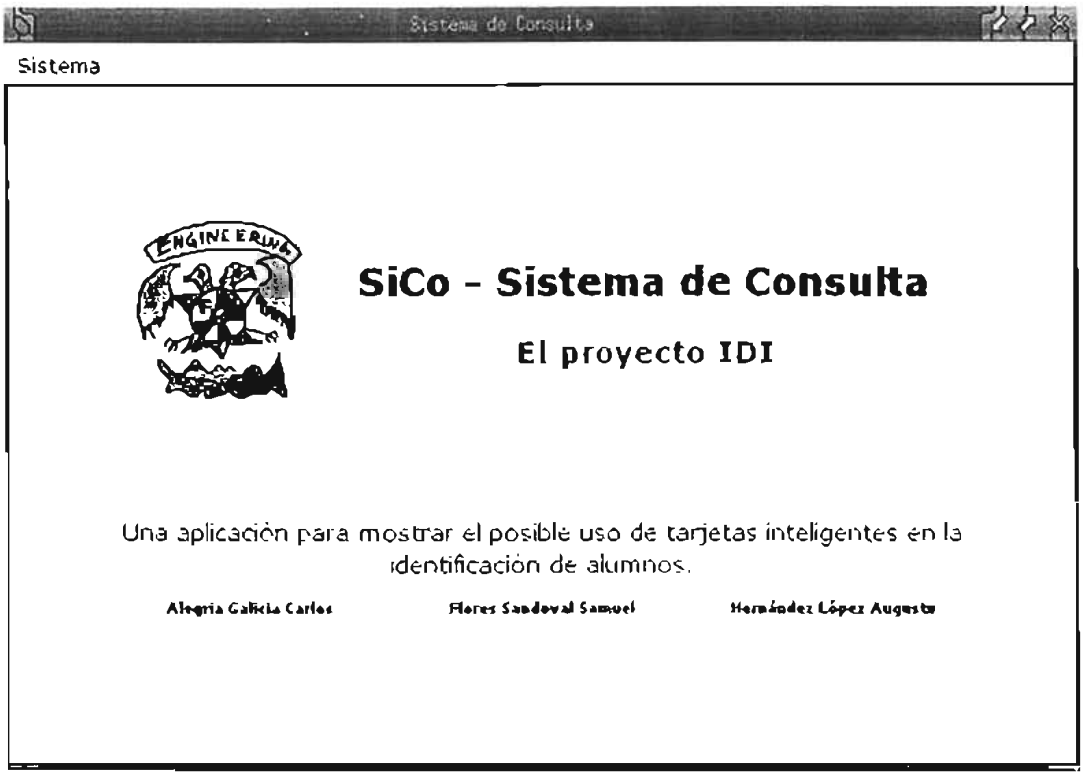

Figura 5.2. Pantalla de inicio

En la esquina superior izquierda de la ventana principal se encuentra el menú *Sistema*, que contiene opciones para todas las acciones que puede realizar el usuario (Figura 5.3, "Opciones del menú Sistema").

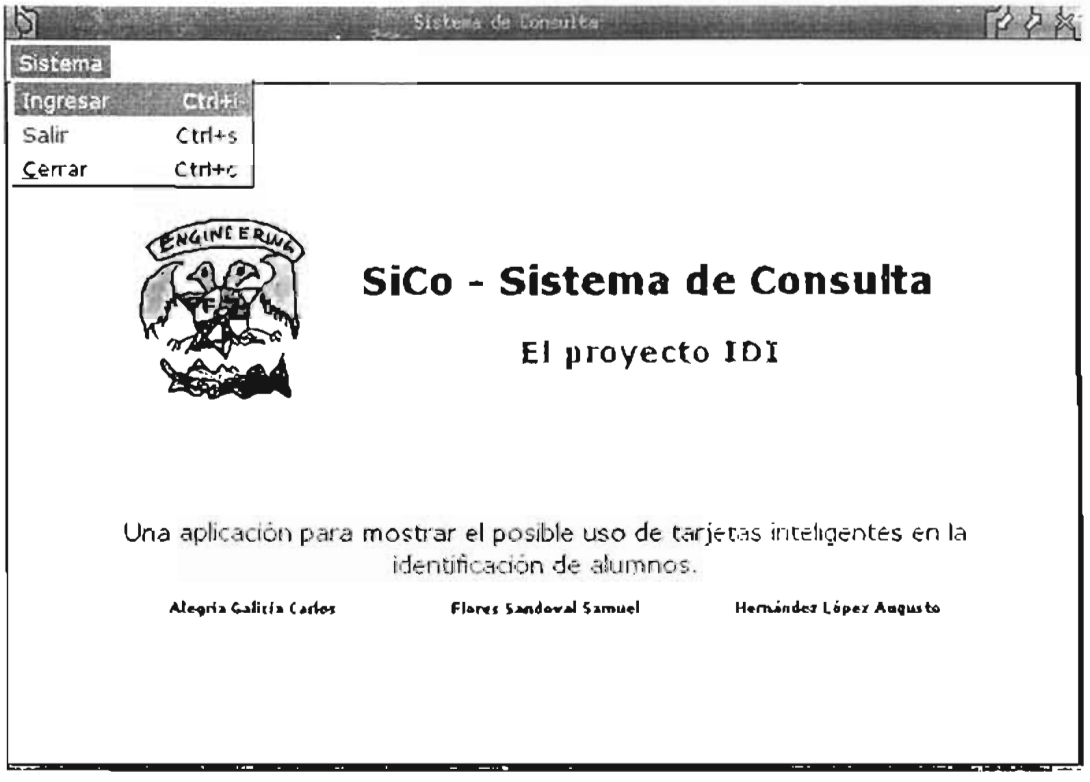

Figura 5.3. Opciones del menú *Sistema*

Para poder realizar la consulta de su historial académico, el usuario primero tiene que hacer login en el sistema, lo que se hace mediante la opción *Entrar* del menú *Sistema*. Luego de proporcionar la información que se requerida (Figura 5.4, "Opción Ingresar"), la aplicación tratará de realizar la autentificación del usuario.

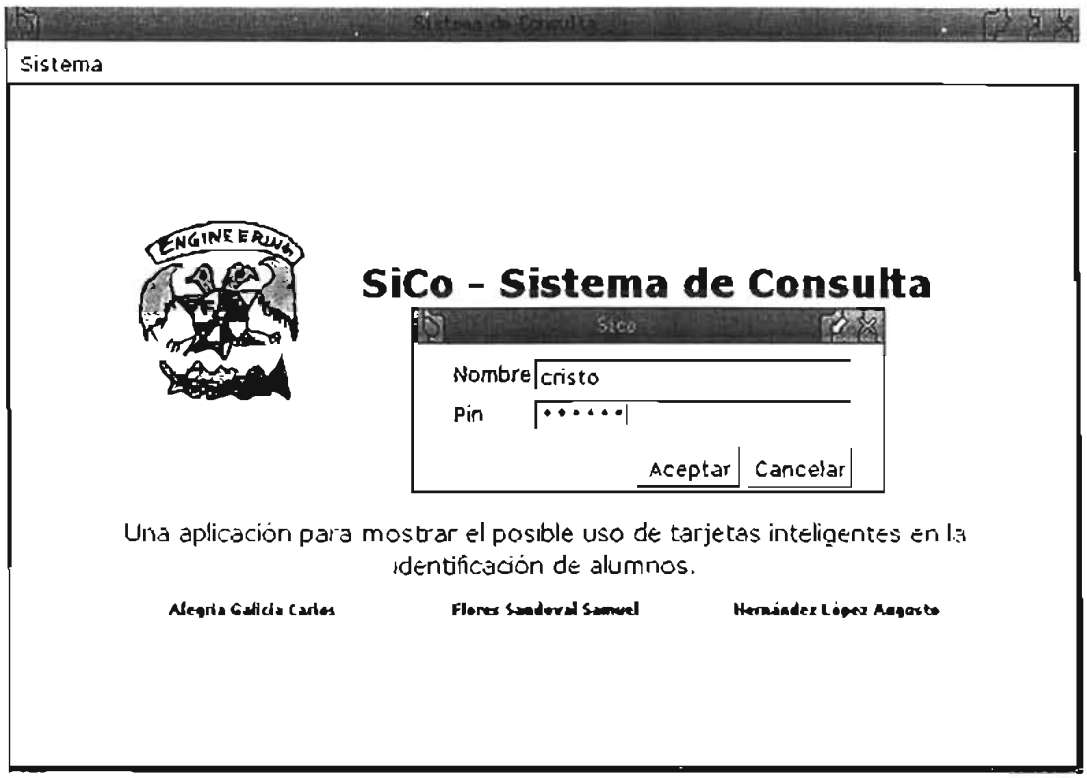

Figura 5.4. Opción *Ingresar*

Si el usuario realizó su autentificación exitosamente, la aplicación mostrará entonces la página que se muestra en la Figura 5.5, "Elección del historial académico", donde el usuario podrá elegir el historial que quiere consultar.

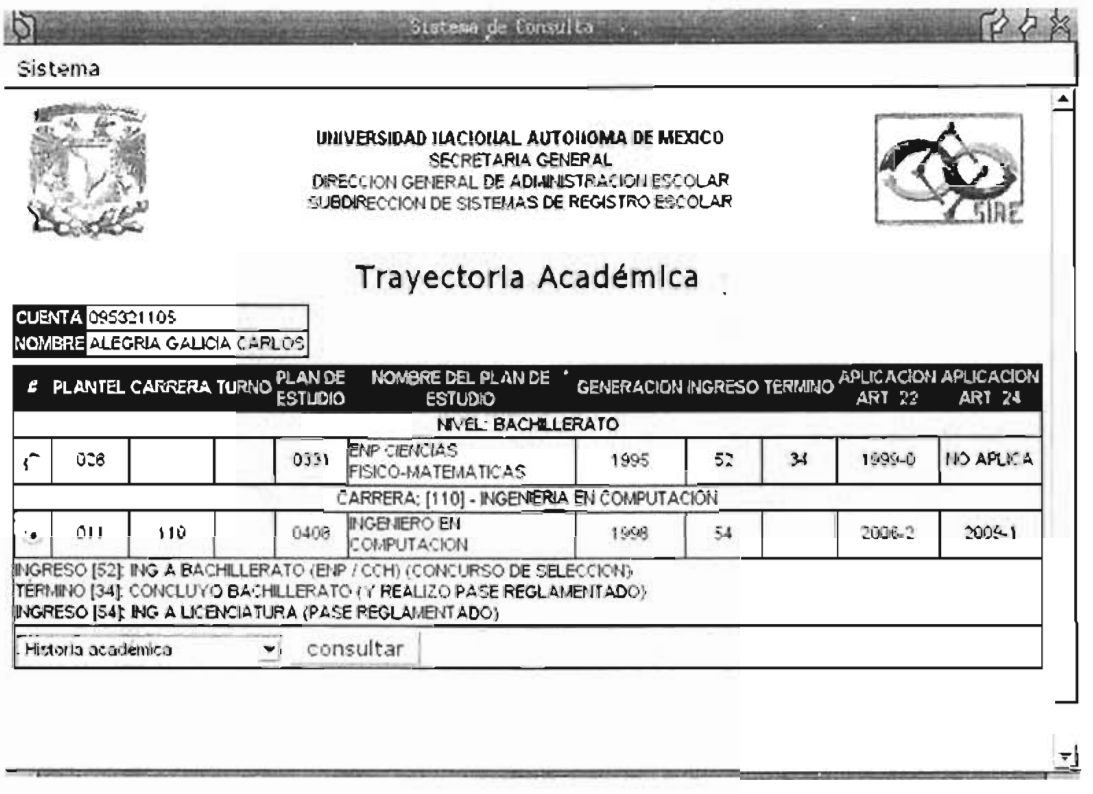

Figura 5.5. Elección del historial académico

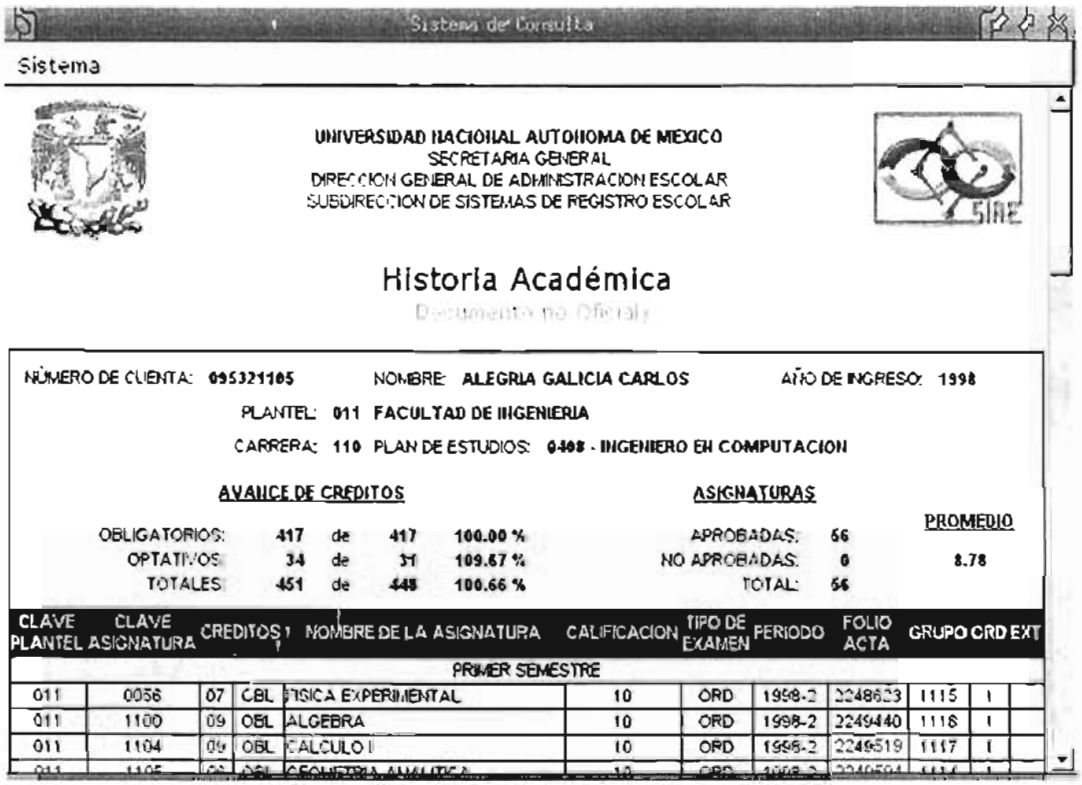

Figura 5.6. Consulta del historial académico

Luego de haber realizado la consulta, el usuario debe hacer logout del sistema, con lo que el historial se cierra y la aplicación puede ser utilizada por un usuario diferente. Esto se hace mediante la opción *Salir* del menú *Sistema* (Figura 5.7, "Opción Salir").

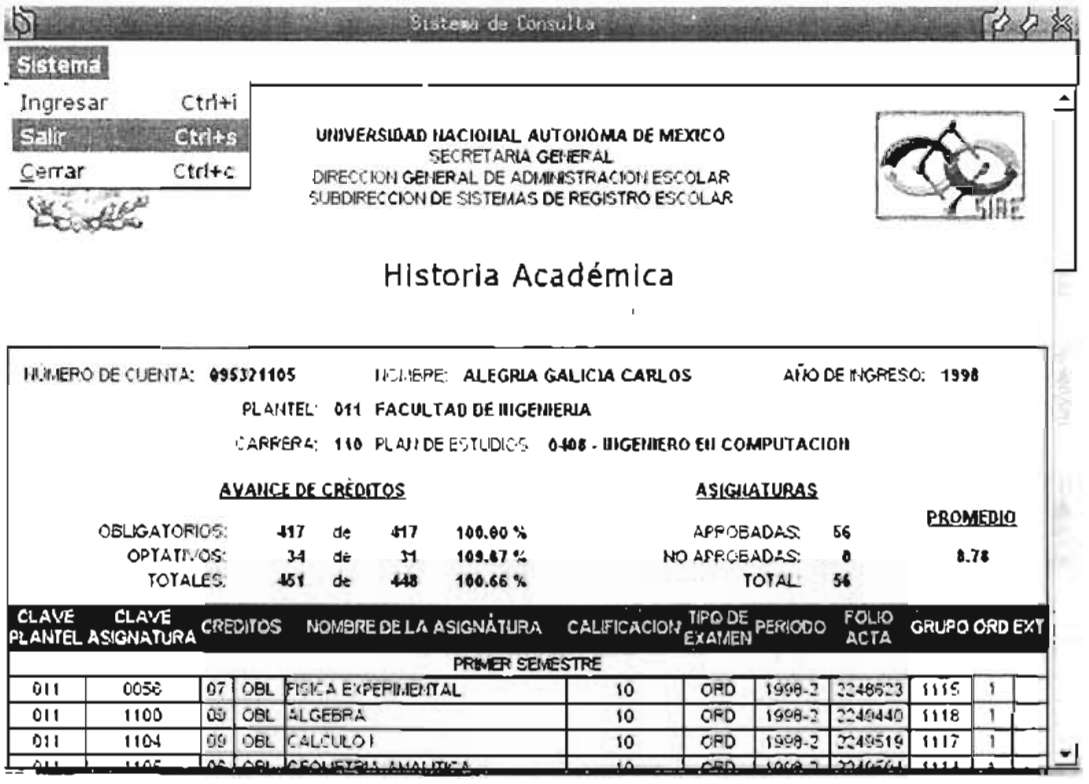

Figura 5.7. Opción *Salir*

Finalmente, para cerrar la aplicación, el usuario puede utilizar la opción *Cerrar* del menú *Sistema* (Figura 5.8, "Opción Cerrar") o el botón correspondiente de la ventana principal.

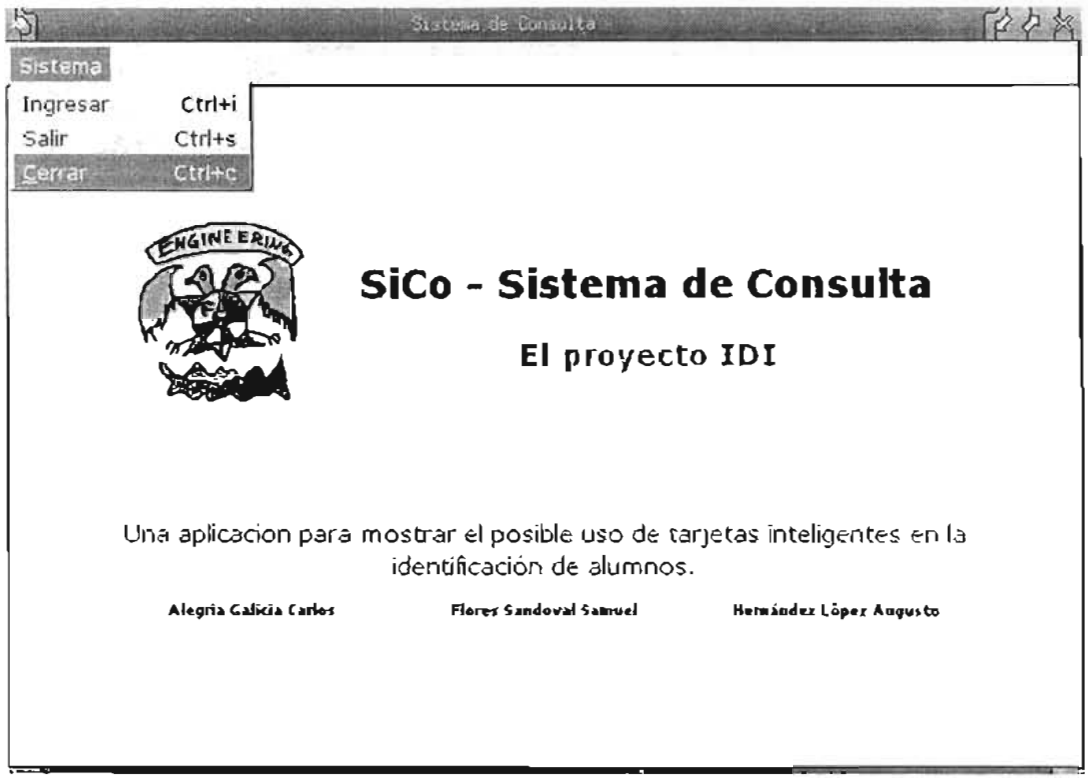

Figura 5.8. Opción *Cerrar*

5.5.2. Diseño

5.5.2.1. Descripción general

El sistema se compone principalmente de tres partes: la autentificación del en el sistema centralizado, la obtención de los datos del usuario, y el acceso al sistema de consulta de historiales académicos.

Primero, cuando el usuario inserta su tarjeta en el lector e introduce su nombre de usuario y su PIN, el sistema utiliza su módulo de autentificación para verificar que el usuario exista en el sistema (identificación), y que en realidad sea quien dice ser (autentificación).

Después, si la autentificación se llevo a cabo con éxito, el sistema obtiene la información que necesita del usuario consultando una base de datos. Esta información es ajena al sistema de autentificación, de manera que el modelo se mantiene independiente y puede ser utilizado por varias aplicaciones.

Ya que se ha obtenido la información del usuario de la base de datos, el sistema realiza una petición al servidor que da el servicio de consulta de historiales académicos, y muestra el resultado al usuario en pantalla.

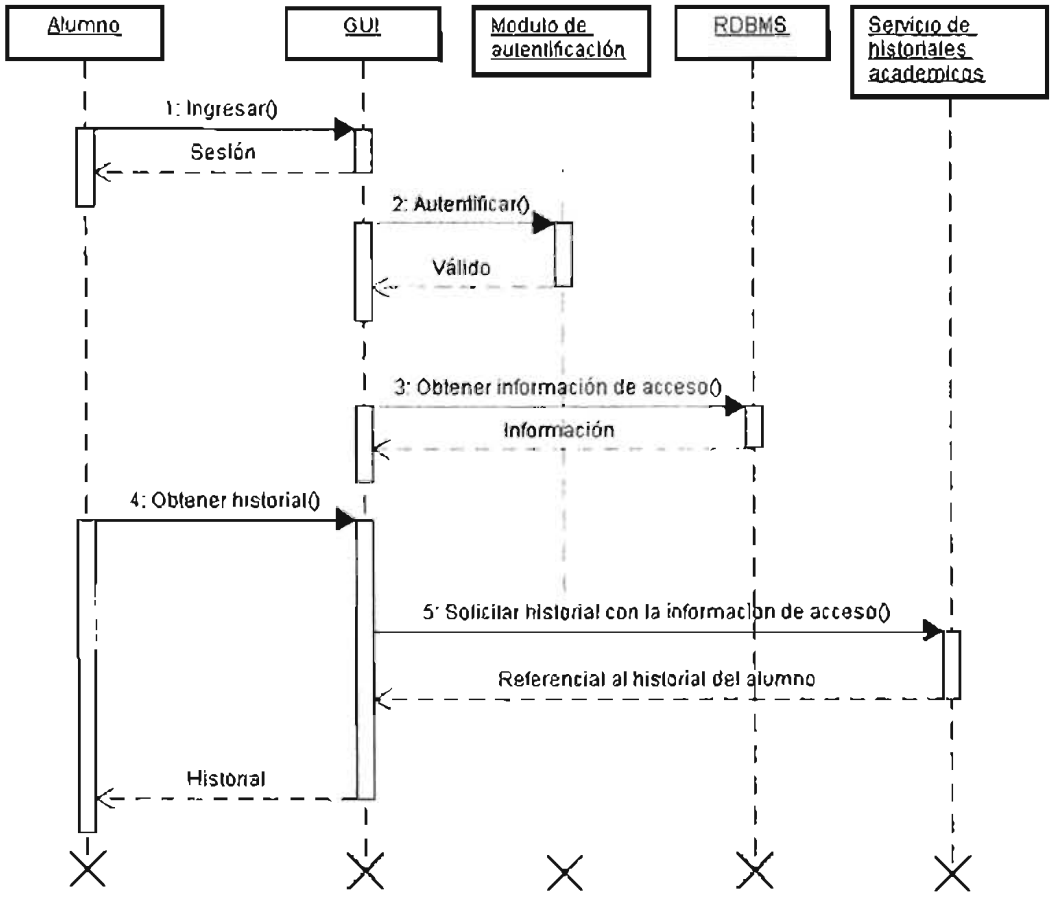

Figura 5.9. Secuencia del "happy path" para el sistema de consulta

5.5.2.2. Consideraciones

Sobre la autentificación.

La autentificación de los usuarios se realiza utilizando el esquema desarrollado en el prototipo anterior. Sin embargo, y debido a que no contamos con acceso tanto a la base de datos como al sistema de información de la universidad, simulamos la autentificación al sistema de información mediante una base de datos propia.

Seguridad.

Aunque la autentificación mediante el uso de tokens criptográficos almacenados en tarjetas inteligentes es uno de los métodos de autentificación más seguros y confiables que existen actualmente, la seguridad del sistema puede verse comprometida por factores como la manera en que los certificados son almacenados, y la manera en que esos certificados son leídos por el sistema. Todos los detalles acerca de la seguridad del modelo de autentificación utilizado se encuentran en el capitulo anterior.

Por otra parte, y ya que no se cuenta con un acceso directo a los servidores de la DGAE [9], la autentificación se realiza de la manera descrita anteriormente, pero para obtener el historial académico del alumno se realiza una petición directa al servidor de la manera en que se hace actualmente, es decir, proporcionando su número de cuenta y su PIN (fecha de nacimiento en formato DDMMAAAA). Esto es evidentemente un problema grave de seguridad, ya que la autentificación puede ser evitada fácilmente mandando una petición del mismo tipo al servidor desde cualquier navegador web, sin embargo, sirve para los propósitos de la aplicación no representa ningún problema. En una integración ideal con el SIAE [10], el sistema se comunicaría con el servidor mediante un protocolo seguro (como HTTPS [11], por ejemplo) para obtener la información.

5.5.2.3. Módulo de autentificación

El módulo de autentificación se encarga de comunicarse con el sistema centralizado de autentificación implementado en el prototipo anterior (ver Sección 5.4, " Prototipo 1: Autentificación centralizada en estaciones de trabajo Unix "), el cual se comunica con la tarjeta inteligente a través de PAM (ver Sección 5.4.3.3.3, "Configurando PAM"). El módulo OpenSC que se encarga de comunicarse con la tarjeta directamente proporciona información sobre el estado del lector y de las acciones que el usuario se encuentra realizando, como la inserción de la tarjeta en el lector, por lo que el modulo de autentificación debe interpretar esa información para realizar la autentificación apropiadamente.

Para efectos de este prototipo, la interacción entre el sistema y el módulo de autentificación es muy sencilla: el sistema sólo captura el nombre y el PIN del usuario

[9] Dirección General de Administración Escolar (http://www.dgae.unam.mx)
[10] Sistema Integral de Administración Escolar (http://www.dgae-siae.unam.mx)
[11] Secure Hyper Text Transfer Protocol

que quiere autentificar, le envía esta información al módulo de autentificación, éste se comunica con PAM e informa al sistema el resultado de la operación, el cual será exitoso si la autentificación se llevo a cabo, o erróneo para cualquier otro mensaje de PAM.

Para una aplicación más robusta, el módulo de autentificación debería informar además la razón por la cual la autentificación no pudo llevarse a cabo, de forma que el sistema pueda informar a su vez al usuario. El módulo OpenSC Implementa más de 20 mensajes de error; sin embargo, las razones más comunes por las cuales una autentificación falla son la siguientes:

- No se encuentra ninguna tarjeta insertada en el lector
- El nombre proporcionado no pertenece a ningún usuario
- El usuario se encuentra registrado en el sistema pero el PIN no es correcto
- El certificado de la tarjeta ha caducado

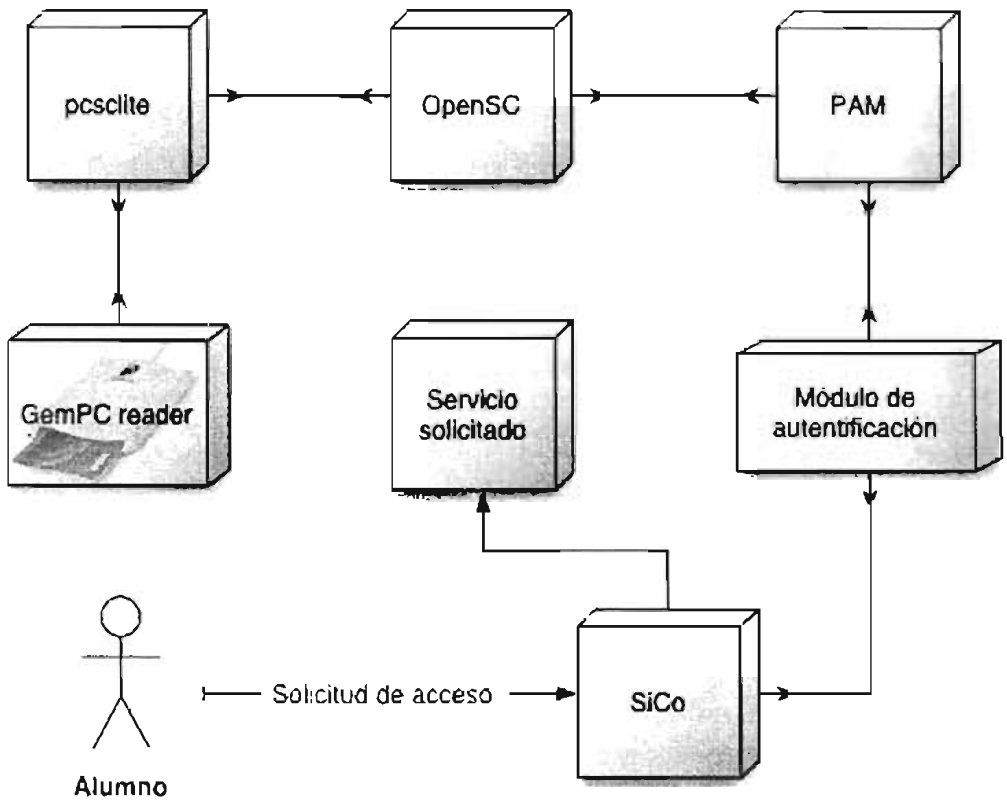

Figura 5.10. Proceso de autentificación

5.5.2.4. Datos del usuario específicos de la aplicación

Ya que en el modelo utilizado la tarjeta inteligente sólamente guarda un certificado digital (para realizar firmas digitales) y una llave privada para realizar la autentificación, cada una de las aplicaciones que requieran autentificar de ésta manera necesitan obtener la información que requieren de otra fuente (generalmente una base de datos). En el caso de éste prototipo, la base de datos almacena información necesaria para entrar al sistema de consulta de historiales académicos.

La información para el acceso a las bases de datos particulares a cada aplicación puede almacenarse en el sistema de archivos del servicio de autentificación centralizado (ver Sección 4.3, "Modelo 2: Identificación de alumnos basado en tarjetas inteligentes criptográficas").

La base de datos que maneja y almacena los datos que SiCo consistirá de una

tabla única llamada **ALUMNO**. Como su nombre lo sugiere, **ALUMNO** contendrá la información relevante del alumno para integrar al servicio de historiales Académicos de la *Dirección General de Administración Escolar* con los servicios de autentificación de la autoridad certificadora que vimos en el primer prototipo. La tabla cuenta con los siguientes campos:

- **username.**

  Se trata del *username* o *login_name* que el servicio maestro ha otorgado al alumno. Éste es el mismo username que se utiliza para identificar a los alumnos en el primer prototipo, es decir ante estaciones de trabajo Unix. Este dato se origina en el servicio maestro y pretende ser la información más importante para identificar a un alumno en todos los otros sistemas de información. Ésta es nuestra **llave primaria**.

- **acc_number.**

  Este campo es el número de cuenta del alumno. Se trata de de una cadena de 9 números decimales enteros consecutivos que tradicionalmente se ha empleado para identificar a los alumnos de la UNAM. Este dato se origina en el servicio externo.

- **birthday.**

  Se trata de la fecha de nacimiento del alumno en el formato DDMMAAAA, es decir, una cadena de 8 números decimales enteros consecutivos. Este dato también se origina en el servicio externo.

Esta pequeña base de datos de una tabla es suficiente para los propósitos que perseguimos.

5.5.2.5. Acceso al sistema de consulta de historiales académicos

En un escenario ideal, el sistema de consulta formaría parte del mismo sistema, es decir, que sería un módulo de consulta de historiales académicos el cual tendría la tarea de consultar la base de datos del sistema (que sería la misma de la cual se obtuvieron los datos extras del usuario) y mostrar los resultados en pantalla; sin embargo, debido a las limitaciones antes mencionadas, la presentación del historial se lleva a

cabo por un sistema diferente, en el cual también hay que realizar una autentificación.

Con los datos obtenidos de la base de datos se realiza una autentificación en el sistema de consulta, del cual se obtiene una respuesta que es presentada al usuario de manera transparente. Ya que el sistema externo tiene una interfaz de usuario a través de Internet, mostrar la misma interfaz en la aplicación luego de realizar la autentificación es una tarea que no tiene muchas complicaciones., ya que sólo se necesita un navegador de web embebido en la aplicación, que se encargará de hacer las peticiones al servidor y de mostrar al usuario la respuesta que se obtenga de éste.

5.5.3. Implementación

5.5.3.1. El lenguaje de implementación

El sistema fue desarrollado principalmente en Java y en C. El lenguaje C fue utilizado por que las librerías para la programación de aplicaciones que se comuniquen con PAM se desarrollaron en este lenguaje, de manera que cualquier aplicación que requiera realizar autentificación de usuarios vía PAM necesariamente debe tener programado algún modulo de comunicación con PAM desarrollado en C.

El lenguaje Java se utilizó para desarrollar el resto de la aplicación. La principal razón por la cual utilizamos éste lenguaje fue por la portabilidad. Como se verá más adelante, el desarrollar el resto de la aplicación en Java nos permite ejecutar el prototipo en varias plataformas sin realizar ningún cambio en el código.

5.5.3.2. Modulo de autentificación

Funcionamiento.

Como se mencionó anteriormente, el módulo de autentificación se encarga de comunicarse con el lector de tarjetas y comunicar el resultado de la autentificación al resto del sistema. La comunicación entre la implementación en C de la autentificación vía PAM y el resto de la aplicación en Java la realizamos mediante una herramienta que proporciona Java para la implementación de métodos en lenguaje nativo del sistema llamada *JNI* (Java Native Interface).

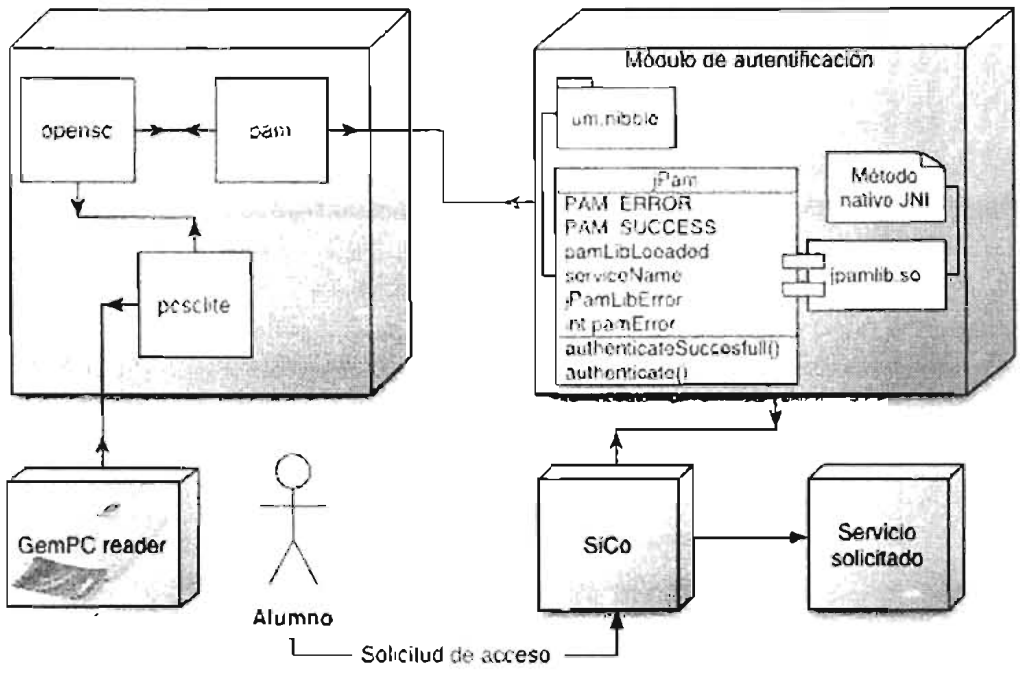

Figura 5.11. Proceso de autentificación a detalle

La librería *libjpam* es el centro de todo el módulo de autentificación, ya que es la encargada de implementar el método nativo que comunica las funciones de autentificación hechas en C, con la interfaz Java utilizada por el sistema.

El método nativo de autentificación.

La autentificación vía PAM puede verse como una comunicación bidireccional entre una aplicación y todo lo que involucra el sistema PAM. Cuando una aplicación realiza la autentificación utilizando las funciones definidas en las librerías de PAM, la comunicación se lleva a cabo del PAM hacia la aplicación, es decir, la aplicación requiere un servicio que PAM le proporciona, y ésta sólo tiene que verificar si el resultado es exitoso o no.

Por otra parte, cuando PAM se encuentra realizando la autentificación, necesita que la aplicación le proporcione los medios para realizar su trabajo; es decir, un login y un password; o para decirlo de forma más general, un identificador y las credenciales apropiadas.

La primera parte de la comunicación se hace utilizando funciones que utilizan las librerías especificadas para cada uno de los tipos de módulo de PAM en el archivo de configuración del servicio (ver Sección 5.4.3.3.3, "Configurando PAM"), y verificando que la operación se haya realizado satisfactoriamente. Estas funciones están definidas en el conjunto de librerías de PAM.

Para realizar la segunda parte de la comunicación, PAM necesita que el desarrollador le proporcione una *función de conversación* por medio de la cual la aplicación se comunicará directamente con las librerías de cada módulo, con el objetivo de proporcionarle la información del usuario que necesite para hacer su trabajo. Lo que debe hacer la función de conversación es leer una estructura en la que se encuentran todos los mensajes que el módulo necesita, y proporcionar una respuesta para cada uno de ellos.

Para obtener la información del usuario y poder responder a los mensajes, algunas aplicaciones interactúan con una consola, otras mediante un display gráfico, etc. En este caso, como la comunicación se realiza con la clase JPam, la información del usuario la proporciona la misma clase, y no hay necesidad de preguntar al usuario sus datos en éste nivel, ya que los datos se obtuvieron antes de llamar al método nativo.

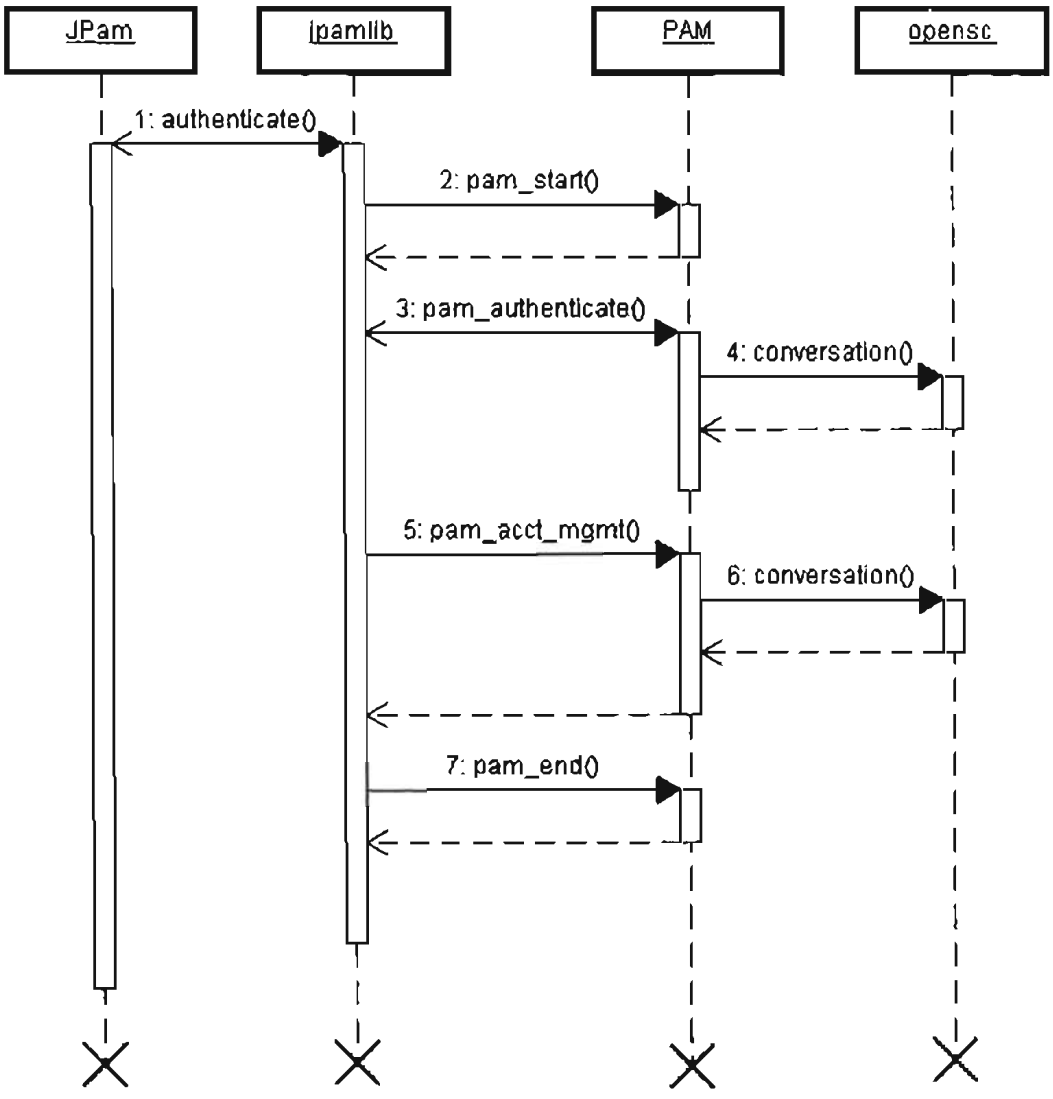

Figura 5.12. Comunicación JPam-jpamlib-PAM-opensc

5.5.3.3. La base de datos

La base de datos fue implementada en nuestro servidor **cad.cele.unam.mx**. Usando el servidor de bases de datos **MySQL**, un DBMS libre muy usado en el mercado. El *diccionario de datos* de la base de datos para el SiCo es el siguiente:

Tabla 5.4. Diccionario de Datos para la Base de Datos de SICo

Campo	Tipo	Nulo	Default
username (pk)	varchar(255)	No	
acc_number	int(9)	No	000000000
birthday	int(8)	No	00000000

Para probar el funcionamiento del sistema, poblamos la base de datos añadiendo algunos registros. En este caso, los datos de los autores de este documento son la mejor opción. A continuación, mostramos todo el *Dump File* para generar nuestra base de datos.

Ejemplo 5.12. *Dump File* para la base de datos utilizada en SICo

```
# phpMyAdmin SQL Dump
# version 2.5.7-pl1
# http://www.phpmyadmin.net
#
# Host: cad.cele.unam.mx
# Generation Time: Mar 02, 2005 at 12:22 AM
# Server version: 4.0.16
# PHP Version: 4.3.8
#
# Database : `sico`
#

# --------------------------------------------------------

#
# Table structure for table `ALUMNO`
#

CREATE TABLE `ALUMNO` (
  `username` varchar(255) NOT NULL default '',
  `acc_number` int(9) unsigned zerofill NOT NULL default '000000000',
  `birthday` int(8) unsigned zerofill NOT NULL default '00000000',
  PRIMARY KEY  (`username`)
) TYPE=MyISAM;

#
# Dumping data for table `ALUMNO`
#

INSERT INTO `ALUMNO` VALUES ('cerealito', 099546030, 18071980);
INSERT INTO `ALUMNO` VALUES ('dobeslao', 096335020, 09121980);
INSERT INTO `ALUMNO` VALUES ('cristo', 095321105, 13091979);
```

El tamaño de ésta base de datos es reducido básicamente por dos razones: primero, SiCo no es más que un prototipo. No pretendemos brindar una solución inmediata a ningún problema inmediato. Segundo, el número de tarjetas inteligentes criptográficas con las que contamos para éste desarrollo es reducido.

5.5.3.4. Acceso al servicio de consulta de historiales académicos

Para accesar al servicio de consulta de la *DGAE*, sólo se hace una petición al servidor como normalmente se haría desde cualquier navegador. El formato del URL para accesar al sistema es el siguiente:

```
http://www.dgae-siae.unam.mx/www_gate.php?acc=aut&usr_logi=login&usr_pass=password
```

Donde *login* es el número de cuenta del alumno, y *password* es su fecha de nacimiento en formato *ddmmaaaa*.

Después de hacer la petición al servidor con la URL anterior, es necesario abrir una sesión en el sistema, lo que se logra al realizar una nueva petición con el URL siguiente:

```
http://www.dgae-siae.unam.mx/www_try.php
```

Cuando el usuario ha terminado de realizar la consulta, éste debe salir del sistema. Ésta operación se lleva a cabo al hacer una petición al servidor con el siguiente URL:

```
http://www.dgae-siae.unam.mx/www_gate.php?acc=out
```

5.6. Reutilizando los prototipos

5.6.1. Introducción

Un punto fuerte de la creación de los prototipos expuestos en este capítulo, es que sientan las bases para la creación de nuevas aplicaciones que utilicen la autentifica-

ción con tarjetas inteligentes, con **la misma infraestructura** de llave pública que ya ha sido creada; de la misma manera que nuestra aplicación de muestra *SiCo*.

Las nuevas aplicaciones podrían ser escritas en C, acoplándose al sistema PAM, previamente configurado (como se vió en el Sección 5.4, " Prototipo 1: Autentificación centralizada en estaciones de trabajo Unix ") o según se describe en manuales de PAM [MORGAN-2001]; o podrían escribirse en el lenguaje de programación Java utilizando la clase JPam, creada enteramente durante el desarrollo de esta investigación. Ésto último resulta de una gran relevancia, debido a que Java es un lenguaje de muy alto nivel, muy popular en diversas industrias y con muchas ventajas (ver Capítulo 3, *Conceptos básicos*).

5.6.2. Acoplando un nuevo sistema al modelo de autentificación

La creación de un nuevo sistema requeriría incluir la clase JPam:

Ejemplo 5.13. Incluyendo la clase JPam en algun servicio

```
...
import um.nibble.jpam.*;
...
JPam            jpam = new JPam(SERVICE_NAME);
...
```

El constructor recibe como parámetro un objeto `String`, una cadena con el nombre del servico, el cual debe también existir como archivo de configuración en `/etc/pam.d/`. Una vez hecho esto, podemos hacer uso de los métodos de la clase. El siguiente fragmento de código muestra la definición del método `authenticateSuccessfull`

Ejemplo 5.14. Métodos relevantes en `JPam`

```
/**
 * Autentifica a un usuario específico según sus credenciales
 *
 */
public boolean authenticateSuccessfull(String userName, String credentials) {
```

```
        return (authenticate(userName, credentials) == JPam.PAM_SUCCESS);
}
```

`authenticate` es un método nativo, implementado según la interfaz JNI [SUN-JNI]. Dicho método "envuelve" a las funciones de PAM que nos permiten usar al módulo pam_opensc, y así acceder desde JAVA a tarjetas inteligentes criptográficas. En el caso específico de pam_opensc, la cadena de credenciales es el PIN del usuario.

Para usar el método desde alguna aplicación, Basta llamar al método y usar su valor de regreso en la toma de alguna desición de acceso. El siguiente es un ejemplo de dicha desición:

Ejemplo 5.15. Ejemplo de uso de la clase JPam

```
boolean         authOk = false;
LoginDialogData loginData;
JPam            jpam = new JPam(SERVICE_NAME);
...
authOk = jpam.authenticateSuccessfull(
                                        loginData.getLogin(),
                                        loginData.getCredentials());
if (authOk) {
    String query =
        "SELECT acc_number, birthday "
        + "FROM ALUMNO "
        + "WHERE username = '" + loginData.getLogin() + "'";
}
...
```

En este ejemplo, `loginData` es una clase con los métodos adecuados para pedir el *username* (`getLogin()`) y el PIN (`getCredentials()`) del usuario y regresarlos como un objeto `String`.

De esta manera, cualquier programador puede echar mano de los resultados de esta investigación para acceder a tarjetas criptográficas ISO-7816 desde Java, envolviendo de forma nativa a lib_opensc.

Capítulo

6.

Conclusión

Los métodos actuales de identificación de alumnos en la Facultad de Ingeniería, y en general los métodos de identificación de personas en toda nuestra universidad, presentan muchas deficiencias que fueron las que motivaron la investigación que reportamos en este documento. Cuando empezamos esta investigación, sabíamos que las *smartcards* se estaban popularizando y ganando terreno en muchos sistemas de identificación, pero no sabíamos prácticamente nada sobre cómo y en qué usarlas.

Nuestras preguntas centrales fueron: ¿es posible usar tarjetas inteligentes para crear una sistema identificación de alumnos? ¿Es posible que ese sistema sea mejor que el actual?, es decir, ¿Qué tanto podemos acercarnos a un sistema ideal (Sección 2.2, "Recopilación de requerimientos")?

El presente trabajo nos permitió obtener un panorama global del estado actual de los sistemas de identificación de personas en sistemas de cómputo y el uso de tarjetas inteligentes para este fin. Además, logramos estudiar diversas tecnologías asociadas y conocer diferentes herramientas de desarrollo y un rango muy amplio de posibles aplicaciones. Ahora conocemos bien los aspectos generales de esta tecnología e incluso hemos desarrollado un par de sistemas prototipo que nos ayudaron a responder nuestras preguntas centrales.

Primero, **es posible** crear un sistema de identificación de alumnos usando tarjetas inteligentes. La capacidad de transportar información, de protegerla y de procesarla dentro de una computadora que está en el *chip* de una tarjeta permite *imaginar* muchas aplicaciones nuevas y mejoras a sistemas computacionales ya existentes. Específicamente, las **tarjetas inteligentes criptográficas** se encuentran en un esta-

do de desarrollo y estandarización que permite establecer las bases para la creación de un sistema de identificación como los que describimos en el capítulo 4. Las bibliotecas de software existentes para este tipo de tarjetas son muchas, e incluso existen desarrollos de software libre (Sección 3.3, "Software Libre") que finalmente fueron los que nos permitieron acceder más rápidamente a estas tarjetas. Usar tarjetas inteligentes en la identificación de alumnos es posible y creemos que nuestros prototipos son una muestra de esta posibilidad.

Por otra parte, durante nuestra investigación encontramos diversas implicaciones que tiene *la identificación* de personas. La más importante es definitivamente la *autentificación.*, pero también existen implicaciones económicas y éticas, las cuales nos llevan a muchas otras preguntas, por ejemplo: ¿Vale la pena hacer una gran inversión en infraestructura para identificar alumnos? ¿En cuánto tiempo se recuperará esa inversión?. ¿Cómo se maneja la información personal de un alumno cuando es posible tener un gran control sobre ella? (la privacidad de la información personal es uno de los requerimientos en un sistema ideal [Sección 2.2, "Recopilación de requerimientos"]).

Creemos que estas preguntas requieren de un análisis detallado para ser contestadas, y dicho análisis está fuera de los alcances de ésta investigación. En ese sentido, nuestra segunda pregunta central queda sin una respuesta clara. No es posible responderla *totalmente* en los términos de la presente investigación pues hacen falta muchos más elementos además del acercamiento técnico a las tarjetas inteligentes.

Aun así, nos atrevemos a arriesgar una conclusión más. Tomando en cuenta que la tendencia actual es hacia brindar servicios integrados a través de medios electrónicos, creemos que la búsqueda, evaluación e investigación de nuevos mecanismos de identificación de personas, resulta fundamental para una institución como nuestra universidad, que pretende mantenerse a la vanguardia tecnológica. Las tarjetas inteligentes se popularizan, sus capacidades aumentan y no son pocas las organizaciones que las adoptan como un mecanismo de identificación. Así, las *smartcards* son, al menos en el corto plazo, un mecanismo de identificación que merece ser estudiado como el mejor medio de implementar en la realidad un modelo de identificación en la Facultad de Ingeniería.

Referencias

Artículos

[ANDERSON-KHUN-1996] Ross Anderson y Markus G. Kuhn. Noviembre 1996. *Tamper Resistance - a Caution Note.*

[ANDERSON-KHUN-1997] Ross Anderson y Markus G. Kuhn. Abril 1997. *Low Cost Attacks on Tamper Resistant Devices.*

[BONEH-DEMILLO-2001] Dan Boneh, Richard A. DeMillo, y Richard J. Lipton. 2001. Journal of CRYPTOLOGY. *On the Importance of Eliminating Errors in Cryptographic Computations.*

[CHAUMETTE-HATCHONDO] Serge Chaumette y Iban Hatchondo. LaBRI, Laboratoire Bordelais de Recherche en Informatique. *JCAT: An enviroment for attack and test Java Card™.*

[CHAUMETTE-SAUVERON] Serge Chaumette, Pascal Grance, Pierre Vigneras, y Damien Sauveron. LaBRI, Laboratoire Bordelais de Recherche en Informatique. *Computing with Java Cards™.*

[CHEUNG-GOODCHILD-2002] Eddy Cheung, Andrew Goodchild, Hoylen Sue, y Ben Fowler. 2002. Distributed Systems Technology Centre. *Strongly authenticated URLs.* Integrating of Web browsers and applications with strong authentication.

[GAUTERON-GIRARD] Laurent Gauteron y Pierre Girard. Gemplus Product Security Group.. *A Smart card Login Module for Java Authentication and Authorization Service.*

[KIRCH-2003] Olaf Kirch. Octubre 2003. Linux-Kongress. *OpenSC - Smart Cards on Linux.*

[KÖMMERLING-KHUN-1999] Oliver Kömmerling y Markus G. Kuhn. Mayo 1999. *De-*

sign Principles for Tampering-Resistant Smartcard Processors.

[LINCON] Arturo Lincón. AMITI, Asociación Mexicana de la Industria de Tecnologias de Información. Sclumberger, Test & Transactions. *Mitos y Realidades sobre las Tarjetas Inteligentes.* "SMART CARDS".

[PETRI] Steve Petri. Secure Service Provider. *An Introduction to Smart Cards.*

[PIETING-2003] Achim Pieting. 2003. PPC Card Systems GmbH. *Functional Specification of the OpenPGP application on ISO Smart Card Operating Systems.* versión 1.0.

[SEIGERS] Frank Seiger's. The OpenCard Consortium. *OpenCard and PC/SC - Two New Industry Initiatives for Smart Cards.*

[STRUCK] Daniel Struck. The OpenSC Project. *OpenSSL user authentification with Apache using x.509 certificates on smart cards.*

[VANDEWALLE-VATILLARD-1998] Jean-Jacques Vandewalle y Eric Vatillard. Septiembre 1998. Gemplus Research Group.. *Developing Smart Card-Based Applications Using Java Card.*

[WU-1998] Thomas Wu. 1998. Standford University, Computer Science Department. *The Secure Remote Password Protocol.*

Referencias Web

[ACM] *Association for Computing Machinery.* http://www.acm.org .

[APPLE-JAVA] Apple Computers. AppleDeveloper Connection. http://devworld.apple.com/java/ .

[BARR-LANGFELDT-2002] *Linux NFS-HOWTO.* . Tavis Barr , Nicolai Langfeldt , Seth Vidal , y Tom McNeal .

[CHEN-1998] Zhiqun Chen. 1998. Java World. *Undersanding Java Card 2.0.* Learn

the inner workings of the Java Card architecture, API, and runtime environment. http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html .

[CHEN-1999] Zhiqun Chen. 1999. Java World. *Learn the programming concepts and major steps of creating Java Card applets.* Learn the inner workings of the Java Card architecture, API, and runtime environment. http://www.javaworld.com/javaworld/jw-07-1999/jw-07-javacard_p.html .

[CardTechnology] *Card Technology Magazine*. The Smart Card News Source. http://www.cardtechnology.com/ .

[DIGIORGIO-1997] Rinaldo Di Giorgio. 1997. Java World. *Smart cards: A primer*. Develop on the Java platform of the future. http://www.javaworld.com/javaworld/jw-12-1997/jw-12-javadev.html .

[DONALD-1996] Davis Donald. 1996. CardTechnology. *How Chips, And Suppliers, Are Changing*. http://www.cardtechnology.com/cgi-bin/readstory.pl?story=20020501CTMN552.xml .

[DPA] Cryptographic Research. *Tamper Resistance*. A differential power análisis overview. http://www.cryptography.com/resources/whitepapers/DPA.html .

[DataQuest] Gartner, Inc.. *Dataquest*. http://www.dataquest.com/ .

[ECLIPSE-SNIPPETS] eclipse.org. Eclipse Fundation. *SWT, Standard Widget Toolkit*. snippets. http://dev.eclipse.org/viewcvs/index.cgi/%7Echeckout%7E/platform-swt-home/dev.html#snippets .

[EuroSmart] *Euro Smart*. The Voice of the Smart Card Industry. http://www.eurosmart.com/ .

[FSF] *The Free Software Foundation*. http://www.fsf.org .

[GNU] *Free as in freedom: The GNU operating system*. http://www.gnu.org .

[Gemplus] *Gemplus International SA.*. beyond the smart. http://www.gemplus.com .

[GemplusSCBasics] Gemplus International SA.. *Smart Technology at Gemplus.* What's so smart about smart cards. http://www.gemplus.com/smart/basics/index.html .

[GlobalPlatform] *Global Platform.* The Standard for Smart Card Infraestructure. http://www.globalplatform.org/ .

[ISO7816 part[1-3]] International Organization for Standardization (ISO). *ISO 7816 (part 1-3), Asyncronous smart card information.* http://www.ttfn.net/techno/smartcards/iso7816123.html .

[ISO7816 part[4]] Internationai Organization for Standardization (ISO). *ISO 7816 part 4, Interindustry command for interchange.* http://www.ttfn.net/techno/smartcards/iso7816prt4.html .

[ISOOverview] CardWerk^{SM}, Smarter Card Solution. *The ISO 7816 Smart Card Standard: Overview.* http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx .

[JAVACARD-GROUP] *Java Card Special Interest Group.* http://www.javacard.org/ .

[KILIÇLI-2001] Tolga KiliÇli. 2001. Internet FAQ Archives. *Smart Card HOWTO.* http://www.faqs.org/docs/Linux-HOWTO/Smart-Card-HOWTO.html .

[LibUSB] *The libusb project home.*. http://libusb.sourceforge.net/ .

[MARINACCI-2004-1] Joshua Marinacci. Mayo 2004. The Java Sketchbook. *The HTML Renderer Shootout, Part 1.* http://today.java.net/pub/a/today/2004/05/24/html-pt1.html .

[MARINACCI-2004-2] Joshua Marinacci. Junio 2004. The Java Sketchbook. *The HTML Renderer Shootout, Part 2.* http://today.java.net/pub/a/today/2004/06/14/html-pt2.html .

[MEDINA] César Medina Salgado. DCSH, UAM, Unidad Azcapotzalco. *Los Sistemas Automáticos de Identificación*.
http://www.azc.uam.mx/publicaciones/enlinea2/num1/1-1.htm .

[MICROSOFT-TECHNET] Copyright © 2005 Microsoft Corporation. Microsoft Tech-Net. *Tarjetas Inteligentes*.
http://www.microsoft.com/latam/technet/articulos/windows2k/smtcard/ .

[MORGAN-2001] Andrew G. Morgan. The Linux Kernel Archives. *The Linux-PAM Application Developer's Guide*.
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_appl.html .

[MORGAN-2001] *The Linux-PAM Application Developers' Guide*.
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html . Andrew Morgan.

[MORGAN-2002] Andrew G. Morgan. The Linux Kernel Archives. *The Linux-PAM System Administrator's Guide*.
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html .

[MORGAN-2002] *The Linux-PAM System Administrators' Guide*.
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html . Andrew Morgan.

[MUSCLE] *Movement for the Use of Smart Cards in a Linux Enviroment.*.
http://www.linuxnet.com/ .

[MUSCLE-PCSCLite] *Movement for the Use of Smart Cards in a Linux Enviroment.*.
PCSC-Lite Home page on Alioth. http://pcsclite.alioth.debian.org/ .

[MUSCLECARD-OSX] *MUSCLECARD on Mac OS X.*.
http://homepage.mac.com/jlgiraud/MscMOSXTutorial/MscMOSXTutorial.htm .

[Neohapsis] *Neohapsis Archives, PAM-List.*.
http://archives.neohapsis.com/archives/pam-list/ .

[OCF] *The OpenCard Framework*. www.opencard.org .

[OpenSC] Juha Yrjölä, Timo Teräs, Olaf Kirch, y Andreas Jellinghaus. Copyright © 2005 The OpenSC Project. *OpenSC Project*. http://www.opensc.org .

[PCSC-WORKGROUP] *PC/SC Workgroup*. Specifications. http://www.pcscworkgroup.com .

[PGP] *PGP International*. http://www.pgpi.org .

[Piccola] *DSTC Titanium, Piccola Project*. http://titanium.dstc.edu.au/security/Piccola/ .

[RANKI-2003] Ranki Wolfgang y Effing Wolfgang. Copyright © 1998-2005 Wolfgang Ranki. *Smart Card*. Handbook. http://www.wranki.de/SCH/SCH.html .

[RSALabs] *RSA Laboratories*. http://www.rsasecurity.com/rsalabs/ .

[RSALabs-PKCS#11] Junio 2004. *RSA Laboratories*. PKCS #11 v2.20: Cryptographic Token Interface Standard.. http://www.rsasecurity.com/rsalabs/node.asp?id=2133 .

[RSALabs-PKCS#15] Junio 2000. *RSA Laboratories*. PKCS #15 v1.1: Cryptographic Token Information Syntax Standard .. http://www.rsasecurity.com/rsalabs/node.asp?id=2141 .

[RSASecurity] *RSA Security*. http://www.rsasecurity.com/ .

[RousseauWeb] Ludovic Rousseau. *GemCore based PC/SC reader drivers*.. http://ludovic.rousseau.free.fr/softwares/ifd-GemPC/ .

[SCAlliance] *Smart Card Alliance*. http://www.smartcardalliance.org/ .

[SCBasics] CardLogix. TOWITOKO. FARGO. *Smart Card Basics*. http://www.smartcardbasics.com/ .

[SMARTCARD-TECH] *Smart Card Technology*. http://sct.co.kr/eng_n/ .

[SUN-JAVA-API] java.sun.com. Sun Microsystems, Java. *Java™ 2 Platform SE v5.0*. http://java.sun.com/j2se/1.5.0/docs/api/index.html .

[SUN-JAVACARD-INFO] Sun Microsystems, Java Card. *Java Card General Information*. http://java.sun.com/products/javacard/reference/docs/index.html .

[SUN-JAVACARD-TECH] java.sun.com. Sun Microsystems, Java. *Java Card™ Technology*.. http://java.sun.com/products/javacard/ .

[SUN-JNI] Beth Stearns. Sun Microsystems, Java. *Trail: Java Native Interface*. http://java.sun.com/docs/books/tutorial/native1.1/index.html .

[SauveronWeb] Damien Sauveron. LaBRI, Laboratoire Bordelais de Recherche en Informatique. *Installation of Smart Card Software*. http://damien.sauveron.free.fr/SmartCard-Howto.html .

[Schlumberger] *Schlumberger*.. smart card serenity. http://www.axalto.com .

[Schlumberger-1996] Copyright © 1997 Schlumberger Limited. Schlumberger. *Schlumberger Annual Report 96 - In Brief*. The Expanding World of the Smart Card. http://www.slb.com/ir/ar/ar96/feature/smartcard1.html .

[iButton] DALLAS Semiconductor. *iButton*.. Touch the Future!. http://www.maxim-ic.com/products/ibutton/ .

Libros

[CHEN-2002] Zhiqun Chen. Copyright © 2002 Sun Microsystems. Addison-Wesley. *Java Card™ Technology for Smart Cards*. Architecture and Programmer's Guide. 0-201-70329-7. Chapter 3. Java Card Technology Overview.

[MCGRAW-1999] Gary McGraw y Ed Felten. Copyright © 1999 Gary McGraw and Ed Felten.. John Wiley & Sons, Inc.. *Securing JAVA*. Getting Down to Business

with Mobile Code, 2nd Edition. 0-471-31952-X. Chapter 8. Java Card Security How Smart Cards and Java Mix.

[PRESSMAN-1997] Roger S. Pressman. Copyright © 1997 McGraw-Hill, Inc. Mayo 2000. McGraw Hill. *Ingeniería del Software*. Un Enfoque Práctico, cuarta edición. 84-481-1186-9. Capítulo 1. El producto.

[WALL-2000] Kurt Wall. Copyright © 2000 Prentice Hall, Inc.. Prentice Hall. *Programación en Linux*. Con Ejemplos. 987-9460-09-X. Capítulo 14. Creación y utilización de bibliotecas de programación.

Listas de correo

[MUSCLE_LIST] *m.u.s.c.l.e.*. MuscleCard Mailing List, muscle@lists.musclecard.com.

[OPENSC_LIST] *opensc*. Open SmartCard Mailing List, opensc-devel@opensc.org.

[PICCOLA_LIST] *titanium piccola*. Piccola Development Mailing List, piccola-dev@dstc.edu.au.