Certificados Digitales y Tarjetas Inteligentes

Máster en Informática

Seguridad en Sistemas de Información

Carlos Verdes Blanco
Mayo 2007

INDICE

1. Qué es una Tarjeta Inteligente

La necesidad de una mayor seguridad y privacidad se ve incrementada a medida que los métodos de identificación electrónicos van sustituyendo a los métodos basados en papeles y en la presencia física. El auge de Internet y la expansión de las redes corporativas para permitir el acceso a los recursos desde fuera de un entorno seguro han aumentado la demanda de soluciones basadas en la tecnología de criptografía de clave pública.

Algunos ejemplos del tipo de servicios que nos ofrece la tecnología de clave pública son: el establecimiento de canales de comunicación seguros a través de las redes públicas, la firma digital que asegura la integridad y confidencialidad de los datos, la autenticación de un cliente frente a un servidor y viceversa y el uso de tarjetas inteligentes para una autenticación segura.

El término “tarjeta inteligente” fue acuñado en 1980 por el publicista francés Roy Bright, pero fueron inventadas entre 1967 y 1968 por dos ingenieros alemanes, Jürgen Dethloff y Helmut Gröttrupp. Las tarjetas inteligentes son una plataforma segura adaptada especialmente para proporcionar una mayor seguridad y privacidad a aplicaciones que se ejecutan en entornos de computación de propósito general como los PCs, ya que son capaces de proporcionar funcionalidades de almacenamiento seguro para información sensible como puede ser:

- Claves privadas
- Números de cuentas
- Contraseñas
- Información médica.

Al mismo tiempo la tarjeta inteligente proporciona la capacidad de realizar distintos procesos con la información que contiene, de forma aislada, sin exponerla al entorno operativo del PC donde tendría que enfrentarse a código potencialmente malicioso (virus, troyanos, etc.). Este aspecto tiene una importancia crítica para determinadas operaciones como:

- Generación de firmas digitales, utilizando claves privadas, para identificación personal
- Autenticación en red basada en secretos almacenados
- Que se mantengan las representaciones electrónicas de un determinado valor (como en las tarjetas de compra prepago).

Una tarjeta inteligente es un pequeño ordenador a prueba de manipulaciones. Una tarjeta inteligente contiene una CPU y almacenamiento no volátil. En la mayoría de las tarjetas parte del espacio de almacenamiento es a prueba de manipulaciones mientras que el resto es accesible para cualquier aplicación que se comunique con la tarjeta.

Las tarjetas inteligentes son dispositivos de almacenamiento seguros con un “cerebro”, dado que almacenan y procesan información. Son dispositivos de almacenamiento con una parte mecánica que facilita la comunicación con los dispositivos lectores. Pueden adoptar distintas configuraciones para su sistema de ficheros y están particionadas en espacios públicos o privados que pueden hacer accesibles o bloquear. También poseen áreas inaccesibles para almacenar y mantener a salvo la información protegida, como las claves privadas asociadas con los certificados que contiene, el monedero electrónico y el sistema operativo al completo.

Las tarjetas inteligentes son resistentes a los ataques ya que su sistema operativo convertirá la tarjeta en inservible si detecta intentos de manipulación. Sin un lector de tarjetas, la información que contiene la tarjeta no es accesible. Incluso con un lector de tarjetas, el usuario debe conocer el PIN asociado con la tarjeta para poder acceder a los contenidos protegidos de la misma. Esto asegura la privacidad de la información segura almacenada en la tarjeta.

Un dato clave de las tarjetas inteligentes es que proporcionan un método seguro de almacenar información. Las tarjetas inteligentes soportan autenticación y autorización: el poseedor de la tarjeta se autentica por medio del PIN, y puede ser autorizado a acceder sólo a un rango de datos particular de la tarjeta, o a realizar unas operaciones particulares con la tarjeta. Usando utilidades instaladas en el ordenador cliente, los usuarios pueden ver el contenido de sus tarjetas, establecer el PIN, renovar certificados y añadir nuevos certificados.

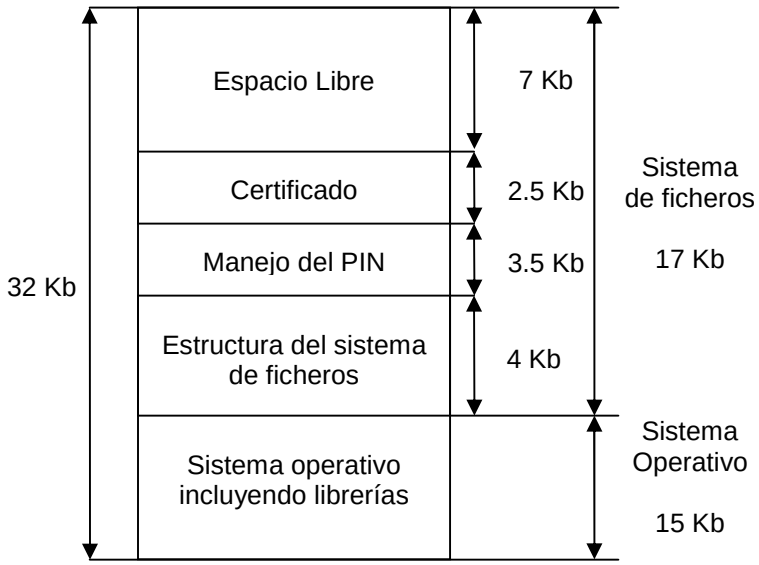
Dado que las tarjetas inteligentes son portables, los usuarios pueden llevar sus certificados de seguridad personales, así como sus correspondientes pares de claves, consigo allí donde vayan. Si un empleado pierde su tarjeta, tan solo es necesario un simple acto administrativo para revocar la validez de su certificado en nuestra red, volviendo la tarjeta inservible para accesos remotos. Incluso con una tarjeta válida, un intruso necesitaría el PIN para tener acceso a los certificados de la tarjeta. Todo esto minimiza el riesgo de un acceso no autorizado a las redes empresariales dado que la autenticación de un usuario se basa en dos cosas: algo que el usuario tiene (la tarjeta inteligente) y algo que el usuario sabe (el PIN).

1.1 Capacidades de proceso y tipos de Tarjetas Inteligentes

Actualmente las tarjetas inteligentes pueden usar microprocesadores de 8, 16 o 32 bits y tienen entre 4 y 256 Kb de RAM para almacenamiento seguro. Normalmente estos elementos están integrados en un único chip y aunque la mayor parte de las implementaciones utilizan un formato de tarjeta de crédito también podemos encontrarlas en otros formatos, como el utilizado en los teléfonos GSM, las tarjetas de suscripción a determinados servicios e incluso en dispositivos USB.

Como se ha comentado una tarjeta inteligente posee un sistema operativo embebido y alguna forma de sistema de ficheros en donde se almacena la información. Otro de los criterios para elegir una tarjeta y chip determinados puede ser su compatibilidad con un sistema operativo de tarjeta en particular. Como ocurre con cualquier otro microprocesador, los de las tarjetas no pueden ejecutar todos los posibles sistemas operativos.

El sistema de ficheros de una tarjeta inteligente es análogo a los sistemas de ficheros usados en los discos, es decir proporcionan la estructura sobre la cual se almacena la información de la tarjeta. Vamos a ver un ejemplo de cómo podría ser la estructura de datos de una tarjeta inteligente, en este caso con una memoria total de 32 Kb, de los que 15 Kb se han usado para almacenar el sistema operativo. Los restantes 17 Kb se han configurado como el sistema de ficheros de la tarjeta sobre el que se construye el resto de la solución. En este espacio se almacena una aplicación de gestión del PIN que facilita la comunicación con el servidor y que ocupa unos 3.5 Kb. El sistema de ficheros y carpetas consume unos 4 Kb y el certificado de usuario otros 2.5 Kb. Esto deja aproximadamente 7 Kb de memoria libre en la tarjeta, lo que nos permitiría alojar otros dos certificados de usuario.



Podemos clasificar las tarjetas inteligentes en dos tipos, de contacto y sin contacto:

- Las tarjetas de contacto necesitan de un lector que facilite la comunicación bidireccional. La tarjeta debe de insertarse en el lector que hace contacto con los terminales de contacto de la tarjeta y que son las vías de comunicación con el chip de

la tarjeta. Estas tarjetas pueden ser de 3 o de 5 voltios, al igual que ocurre con las CPUs de sobremesa. Los lectores de tarjeta de contacto los podemos encontrar insertados en edificios, en teléfonos móviles, en dispositivos portátiles y como dispositivos aislados conectados a un ordenador a través de un puerto serie o USB, como dispositivos PCMCIA para portátiles e incluso incorporados a un teclado.

§ Las tarjetas sin contacto utilizan detectores de proximidad para intercambiar información con la tarjeta. Se incluye una antena en el perímetro de la tarjeta que se activa cuando es radiada a una distancia específica del detector. La configuración de la antena de la tarjeta y del detector posibilita conexiones desde un par de centímetros a metros. La transmisión de datos está codificada y se puede cifrar usando una combinación de los algoritmos soportados por el chip de la tarjeta. Según lo sofisticada que sea la conexión se pueden mantener comunicaciones simultáneas con múltiples tarjetas que estén en el rango del detector. Dado que no se requiere una conexión física con el lector, el rango de usos de estas tarjetas se está expandiendo de forma notable.

1.2 Las tarjetas FNMT-RCM

En la actualidad la Fábrica Nacional de Moneda y Timbre – Real Casa de la Moneda dispone de dos chips distintos para sus tarjetas inteligentes, con su propio sistema operativo desarrollado también por la FNMT.

Las tarjetas FNMT-RCM están especialmente diseñadas para infraestructuras de clave pública en las que se requiere autenticación de una entidad, integridad, confidencialidad de datos y el no repudio en origen. Mantienen el material sensible criptográfico siempre interno a la tarjeta y protegen su uso mediante control de acceso. De esta forma se obtiene una considerable ventaja en términos de seguridad y portabilidad sobre las soluciones software.

Los dos modelos disponibles de tarjeta inteligente son el de Siemens SLE66CX320P y el de ST ST19XL34.

Características principales del chip SLE66CX320P:

- CPU de 16 bits de alto rendimiento
- 64 Kbytes de ROM para código
- 256 bytes RAM interna + 2 Kbytes de memoria RAM extendida
- 32 Kbytes de EEPROM
- Periféricos integrados
  - Cripto-procesador avanzado de 1100 bits
  - Generación real de números aleatorios
  - Módulo de cálculo de CRCs
  - Módulo para la transferencia asíncrona de datos
- Características de seguridad
  - Cifrado dinámico de memorias y buses
  - Sensores para control de tensión y frecuencia
  - Escudo activo sensible a manipulaciones del hardware
  - Generador aleatorio de estados de espera y picos de consumo
  - Identificación única para cada chip
  - Sensor óptico y funciones adicionales
  - Modo de ahorro de energía
  - Rango de frecuencia externa 1-5Mhz
  - Frecuencia interna de hasta 10Mhz

Características del Sistema Operativo FN19:

- 32 Kbytes de EEPROM disponibles para aplicaciones, de los cuales 16 Kbytes pueden ser usados para extender la funcionalidad del S.O.

- Adecuación a la norma ISO 7816-1/-2/-3. Protocolo de transmisión T=0
- Almacenamiento seguro de datos sensibles:
    - Claves RSA 1024 bits de firma y confidencialidad
    - Carga externa de claves RSA en claro o cifradas
    - Claves Triple DES
    - Códigos secretos de usuario (PIN)
    - Claves secretas de usuario para autenticación
    - Claves de aplicación
- Servicios criptográficos
    - Algoritmo de hash SHA-1. Posibilidad de realizarlo completo dentro de la tarjeta o sólo el último bucle, para ganar velocidad
    - Firmas digitales RSA de 1024 bits de tamaño de clave. Posibilidad de realizar el relleno PKCS#1 dentro o fuera de la tarjeta
    - Intercambio de claves de sesión o simétricas entre dos entidades basado en RSA de 1024 bits
    - Cifrado simétrico Triple DES
    - Generación de claves RSA de 1024 bits según el estándar PKCS#1
- Servicios de autenticación
    - Interna o de la tarjeta por la aplicación, RSA
    - De usuario ante la tarjeta, con claves de aplicación
    - De la aplicación ante la tarjeta, RSA o por claves de aplicación
    - Entre entidades remotas. Protocolos one-way y two-way
- Control de acceso definible para cada fichero
- Securización de comandos por criptograma. Destinados a garantizar la integridad del comando y su autenticidad
- Estructura interna de ficheros según el estándar PKCS#15

Características principales del chip ST19XL34V2:

- CPU de 8 bits avanzado con modos extendidos de direccionamiento
- 96 Kbytes de ROM para código organizable en particiones
- 4 Kbytes de RAM organizable en particiones
- 34 Kbytes de EEPROM
- Periféricos integrados
    - Cripto-procesador hardware avanzado de 1088 bits
    - Acelerador hardware con medidas de seguridad avanzadas para el cálculo de Triple Des
    - Firewalls de seguridad entre particiones de memoria; permite especificar restricciones de acceso entre las diferentes áreas
    - 2 generadores internos de números impredecibles
    - Módulo para la transferencia asíncrona de datos
    - Módulo para el cálculo de CRC ISO 3309
- Características de seguridad
    - Identificación única para cada chip
    - Características adicionales de seguridad de última generación
- Funciones adicionales
    - Modo Standby de ahorro de energía
    - Rango de frecuencia externa 1-10Mhz
    - Tensión de trabajo entre 2.7 y 5.5 voltios

Características del Sistema Operativo Ceres v2.0:

- 32 Kbytes de EEPROM disponibles para aplicaciones.
- Adecuación a la norma ISO 7816-1/-2/-3. Protocolo de transmisión T=0
- Interfaz de comandos según ISO 7816-4 y PC/SC
- Control de acceso a datos sensibles
    - Definible para cada fichero
    - Clave administrativa para las operaciones de escritura/lectura de datos, creación de ficheros y almacenamiento de claves
    - Control de acceso de tipo ‘Only-Once’; obliga a que la condición de acceso sea satisfecha justamente antes de realizar la operación

- RSA
    - Soporte para claves en formato CRT y en formato normal (exponente privado, exponente público y módulo)
    - Soporte para componentes p y q de longitud variable
    - Se puede almacenar, generar y usar claves de cualquier longitud de hasta 2176 bits en formato CRT y de hasta 1088 bits en formato normal
    - Operaciones de firma digital
    - Intercambio de claves de sesión o simétricas entre dos entidades
    - Optimización de las contramedidas ante ataques de tipo SPA y DPA
    - Claves almacenadas en ficheros independientes
    - Condiciones de acceso independientes para cada clave
    - Se pueden crear nuevos ficheros de clave durante la fase de vida de usuario. (P.ej. hasta 15 certificados con claves en formato normal de 1024 bits)
- Servicios criptográficos
    - Algoritmo de hash SHA-1. Posibilidad de realizarlo completo dentro de la tarjeta o sólo el último bucle, para ganar velocidad
    - Firmas digitales RSA de 1024 bits de tamaño de clave. Posibilidad de realizar el relleno PKCS#1 dentro o fuera de la tarjeta
    - Intercambio de claves de sesión o simétricas entre dos entidades basado en RSA de 1024 bits
    - Cifrado simétrico Triple DES
    - Generación de claves RSA de 1024 bits según el estándar PKCS#1
- Servicios de autenticación
    - Interna o de la tarjeta por la aplicación, RSA
    - De usuario ante la tarjeta, con claves de aplicación
    - De la aplicación ante la tarjeta, RSA o por claves de aplicación
    - Entre entidades remotas. Protocolos one-way y two-way

1.3 Los estándares. ISO 7816-X y PC/SC

Hasta hace relativamente poco tiempo, el uso de tarjetas inteligentes en el entorno del PC se veía dificultado por la ausencia de interoperabilidad a distintos niveles. En primer lugar, la industria carecía de estándares para comunicar el PC con los terminales de tarjetas. Esto hacía difícil crear aplicaciones que pudiesen trabajar con terminales de varios fabricantes. Los intentos por resolver este problema al nivel de las aplicaciones invariablemente suponían un aumento de costes, tanto en desarrollo como en mantenimiento. Además creaba un problema importante para el usuario, ya que un terminal utilizado con una aplicación podía no funcionar con futuras aplicaciones.

En segundo lugar, no había un interface de programación de alto nivel común y ampliamente aceptado para el uso de los terminales. El encapsulamiento de los interfaces puede simplificar enormemente el desarrollo de aplicaciones y reducir costes permitiendo que se comparta el software de acceso a bajo nivel entre múltiples aplicaciones. Además, un interface de alto nivel estandarizado permite a las aplicaciones reducir su dependencia de una implementación específica, permitiendo que dicha aplicación pueda trabajar con futuras implementaciones, es decir, se deben separar las tecnologías de manejo de tarjetas de las aplicaciones que manejan tarjetas, de forma que dichas tecnologías se vuelvan transparentes para las aplicaciones.

Para optimizar el beneficio tanto de la industria como de los usuarios finales, era necesario desarrollar soluciones a estos problemas que soportasen una variedad de entornos operativos y una amplia gama de aplicaciones. Sólo de esta manera se podía animar al desarrollo de aplicaciones con tarjetas inteligentes en el entorno PC a un coste aceptable.

Para promover la interoperabilidad entre tarjetas y terminales de diferentes fabricantes, la ISO desarrolló estándares internacionales que definen las características físicas de las tarjetas inteligentes de contacto. Por ejemplo, el tamaño de las tarjetas está definido en el ISO 7810. Los estándares ISO 7816 y subsiguientes cubren los parámetros de fabricación, las características físicas y eléctricas, la localización de los puntos de contacto, los protocolos de comunicación, el almacenamiento de datos, etc. Es decir, se ocupan del nivel físico y de los protocolos de bajo nivel, pero deja suficiente espacio para que la variación e interpretación de

dichos componentes por parte de diferentes fabricantes haga éstas a menudo incompatibles entre sí aunque todas ellas sean conformes al estándar ISO 7816. Este estándar apenas menciona las capas más altas de programación, aquellas con las que prefiere tratar un programador de aplicaciones que use tarjetas inteligentes.

Además de los estándares físicos y de fabricación se han definido determinados estándares para dominios de aplicación específicos. Entidades de tarjetas de crédito, fabricantes de teléfonos móviles, bancos europeos y americanos son ejemplos de organizaciones que han personalizado los usos y procedimientos de las tarjetas inteligentes para adaptarlas a los servicios que ofrecen. En 1996, Europay, MasterCard y VISA (EMV) definieron una especificación de tarjeta inteligente que adoptó los estándares ISO 7816 y definió algunos tipos de datos y reglas de codificación adicionales para ser usada por la industria de servicios financieros. La industria de telecomunicaciones europea también adoptó los estándares ISO en su especificación de tarjeta inteligente para su Global System for Mobile Communications (GSM) que permite la identificación y autenticación de los usuarios de teléfonos móviles.

A pesar de que todas estas especificaciones (ISO 7816, EMV, GSM) son un paso adelante en la dirección correcta, cada una de ellas es o de muy bajo nivel o específica para unas aplicaciones concretas, por lo que no han tenido un gran soporte por parte del resto de la industria. Ninguna de las especificaciones mencionadas establece elementos de interoperabilidad entre aplicaciones como APIs independientes de los dispositivos, herramientas de desarrollo o compartición de recursos.

Por todo ello en mayo de 1996 se creó el PC/SC Workgroup, una alianza formada por los siguientes fabricantes de ordenadores y tarjetas inteligentes: Bull, Hewlett-Packard, Microsoft, Schlumberger y Siemens Nixdorf. Más tarde se unieron al grupo: Gemplus, IBM, Sun Microsystems, Toshiba y VeriFone. El objetivo principal de este grupo ha sido el desarrollar especificaciones que resolviesen los problemas de interoperabilidad entre las tarjetas y los terminales y la cooperación entre el terminal y el sistema operativo. En diciembre de 1997, el grupo publicó la primera versión de las especificaciones. Dividida en ocho partes, la especificación se extiende desde las características físicas de las tarjetas y los terminales hasta la capa de programación de aplicaciones. Las especificaciones de PC/SC están basadas en los estándares ISO 7816 y son compatibles tanto con los de EMV como con los de GSM. Estas especificaciones han tenido un amplio soporte por parte de la industria y se espera que en el futuro se conviertan en un estándar independiente.

El hecho de tener un modelo estándar acerca de cómo los terminales y tarjetas operan con un ordenador asegura la interoperabilidad entre tarjetas y terminales de diferentes fabricantes. Las APIs independientes del dispositivo aíslan a los desarrolladores de aplicaciones de las diferencias entre implementaciones, lo que reduce el coste de desarrollo debido a cambios en el hardware soportado. La especificación PC/SC se centra en un ordenador personal y en un terminal de tarjetas conectado a él. El componente principal de dicha especificación es el Resource Manager, que debe ser integrado en el sistema operativo del ordenador. Para la regulación de la propia tarjeta, la PC/SC se basa en ISO 7816, sin embargo se puede utilizar con cualquier tipo de tarjeta inteligente.

En el “entorno Microsoft”, esta compañía ha desarrollado los componentes base para el manejo de tarjetas inteligentes basadas en la especificación PC/SC para la totalidad de sus sistemas operativos y dichos componentes ya se encuentran incluidos de forma nativa a partir de Windows 2000.

En el “entorno del software libre” podemos destacar varias iniciativas como GlobalPlatform (www.globalplatform.org) que es una organización independiente cuya misión es diseñar, mantener y promover la adopción de estándares que permitan una infraestructura abierta e interoperable entre las tarjetas inteligentes, dispositivos y sistemas que las usan, que simplifique y acelere el desarrollo, mantenimiento y distribución de aplicaciones.

OpenCard (www.opencard.org) es una especificación que define un framework orientado a objetos que reside en el ordenador al que se conecta la tarjeta. La implementación ha sido desarrollada usando el lenguaje Java y su uso está orientado a desarrollar aplicaciones

también escritas en Java lo que garantiza su portabilidad e interoperabilidad, que son claves para una rápida adopción de la misma. OpenCard proporciona un interface común tanto para los terminales de tarjetas como para las aplicaciones residentes en la tarjeta. Al igual que PC/SC la tarjeta también se regula por el ISO 7816 aunque se podría utilizar cualquier otra. Esta implementación permite la interacción con dispositivos soportados por la especificación PC/SC.

Otra iniciativa interesante es JavaCard (www.javacardforum.org) cuyo objetivo es introducir Java en una tarjeta inteligente para desarrollar aplicaciones. A veces se confunden los nombres OpenCard y JavaCard, probablemente por el hecho de que la implementación de OpenCard está hecha en Java. Sin embargo, la diferencia entre estas dos iniciativas es simple y fundamental: JavaCard trata sobre Java introducido en la propia tarjeta, mientras que OpenCard (al igual que PC/SC) trata del manejo de la tarjeta desde fuera. Una JavaCard será usada conjuntamente con OpenCard y/o con PC/SC. Como OpenCard es un framework Java, esto hará que se integre muy bien con JavaCard.

Una última iniciativa que mencionaré es MUSCLE, que significa Movement for the Use of Smart Cards in a Linux Environment (www.linuxnet.com). MUSCLE es un proyecto que coordina el desarrollo de tarjetas inteligentes y aplicaciones bajo el entorno Linux. El objetivo perseguido es desarrollar un conjunto de drivers, API's y manejadores de recursos para varias tarjetas inteligentes y terminales dentro del entorno GNU.

Por motivos de tiempo y por preferencias personales del autor, a lo largo de este trabajo, siempre que hablemos de la arquitectura de una solución basada en tarjetas inteligentes nos estaremos refiriendo a una basada en la especificación PC/SC. Cuando hablemos de implementación de soluciones nos referiremos exclusivamente a aquellos servicios y utilidades que se encuentran disponibles en el entorno Microsoft.

2. Arquitectura de un sistema de Tarjeta Inteligente[1]

En este apartado veremos un resumen de la especificación PC/SC, describiendo la funcionalidad básica requerida para las tarjetas, terminales y ordenadores que permita la interoperabilidad entre elementos compatibles de diversos fabricantes.

Como hemos visto, la especificación PC/SC fue inicialmente creada para la plataforma Windows y está soportada en todas sus versiones. El creciente uso de tarjetas inteligentes en el entorno de los ordenadores personales y estaciones de trabajo ha dejado patente la necesidad de darles soporte en unas plataformas que no son tan usadas como Windows; plataformas como Macintosh, Solaris, Linux y otros sistemas operativos. Muchas empresas tienen un gran número de máquinas con estas plataformas que se usan para aplicaciones específicas y es necesario darles soporte. Anteriormente, la mayoría de las soluciones en estas plataformas eran propietarias y sólo funcionaban con el hardware del fabricante de la solución.

Como resultado del trabajo desarrollado por el PC/SC Workgroup, existe una documentación que describe cómo se debe comportar un controlador de recursos para tarjetas inteligentes y dispositivos criptográficos independientemente del sistema operativo. Los primeros paquetes de desarrollo para PC/SC incluían documentación de la API que las aplicaciones podían usar para acceder a los terminales de tarjeta inteligente de forma abstracta. El estándar PC/SC proporcionó la oportunidad a otras plataformas de soportar tarjetas inteligentes de una forma compatible con la plataforma Windows.

El grupo MUSCLE creó una pila PC/SC para un entorno no Windows, inicialmente para la plataforma Linux. Posteriormente muchos fabricantes y universidades adoptaron la pila PC/SC MUSCLE (llamada PC/SC Lite) haciendo que estuviera disponible para numerosas plataformas, sin coste alguno e incluyendo el código fuente. Actualmente podemos encontrar implementaciones PC/SC Lite para Linux, Solaris, Mac OS X, BSD y HP-UX.

Para desarrollar una pila de tarjeta inteligente en varias plataformas, primero se necesita un controlador de recursos local y luego se puede construir soporte para servicios remotos como una capa situada por encima de éste. Para proporcionar una comunicación entre procesos válida en varias plataformas se escogieron los sockets tipo POSIX. La especificación PC/SC Lite está escrita con herramientas GNU sobradamente conocidas permitiendo que se use en cualquier plataforma donde estén disponibles las mismas. Está escrita usando las convenciones ANSI C y con pocas interacciones con librerías externas.

Por todo ello, podemos concluir que aunque la especificación PC/SC no es un estándar reconocido por ningún organismo estandarizador, debido al amplio soporte que posee en todo tipo de plataformas, sí es un estándar *de facto*.

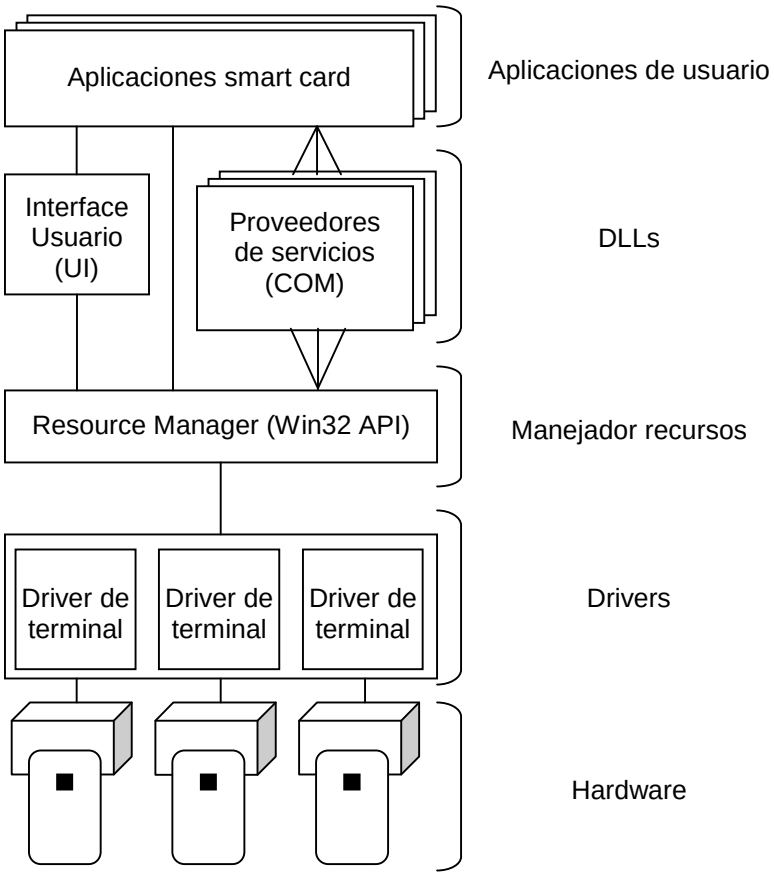
2.1 Descripción de los componentes

Los componentes básicos de un sistema de tarjetas inteligentes basándonos en la especificación PC/SC están formados por tres elementos:

- Un Resource Manager que usa una API de Windows
- Un Interface de Usuario que trabaja con el Resource Manager
- Varios proveedores de servicios base que proporcionan acceso a servicios específicos. Al contrario que la API del Resource Manager, los proveedores de servicios utilizan un interface COM.

La siguiente figura muestra la forma en que se relacionan los mencionados componentes en esta arquitectura:

[1] Referencias: [PCSC05] [JUGU98] [CERE03]



2.2 Tarjetas Inteligentes

El término tarjeta inteligente se usa normalmente para describir a un tipo de dispositivos del tamaño de una tarjeta de crédito, con distintas capacidades, desde tarjetas que simplemente almacenan datos de forma segura hasta las que poseen sofisticados servicios criptográficos, tarjetas sin contacto y tarjetas con circuitos integrados (ICCs). Todas estas tarjetas se diferencian por sus funcionalidades tanto entre sí, como con respecto a las más familiares tarjetas de banda magnética de crédito, débito y para los cajeros automáticos. Las que más nos interesan son las de tipo ICC porque pueden realizar operaciones más sofisticadas, como la firma electrónica y el intercambio de claves.

Para que una tarjeta inteligente se pueda usar bajo la especificación PC/SC implementada en Windows, debe cumplir tanto física como eléctricamente con los estándares ISO 7816-1/-2/-3. También se soportan algunos tipos de tarjeta sin contacto. El objetivo de esta especificación es proporcionar los mismos interfaces y la misma forma de trabajar a nivel aplicación, tanto para las tarjetas de contacto como para las de sin contacto.

Para poder usar una tarjeta inteligente en Windows, primero debemos instalar un programa que nos proporcionará el fabricante porque hasta el momento no hay ningún modelo Plug and Play (PnP) establecido para tarjetas inteligentes. No hay ningún estándar para codificar la cadena de identificación única usada para identificar inequívocamente las tarjetas del mismo tipo. El software de instalación de la tarjeta normalmente instala un proveedor de servicio asociado a la tarjeta que registra sus interfaces en el Resource Manager. El Resource Manager asocia entonces la tarjeta con los interfaces registrados, permitiendo a las aplicaciones acceder a los servicios de la tarjeta basándose en los interfaces que soporta. También se puede asociar una tarjeta con interfaces registrados con anterioridad por otro proveedor de servicios.

2.3 Drivers de terminal de tarjetas

Un terminal de tarjetas es el dispositivo físico a través del cual las tarjetas se comunican con el ordenador. El terminal proporciona la alimentación necesaria al microprocesador al igual que una señal de reloj y una línea de entrada/salida a través de la cual pasar información entre el terminal y la tarjeta. Un terminal puede tener una o varias ranuras para tarjetas y también puede incorporar algunas utilidades como una pantalla o un teclado de introducción de PINs. Los terminales pueden tener implementaciones de muy distinto tipo permitiendo a los fabricantes decidir dónde prefieren alojar la inteligencia del sistema, o bien embebida en el propio dispositivo, o bien en el driver software alojado en el ordenador. En los dispositivos más sencillos, el terminal proporcionará poco más que la conexión eléctrica y la señal de entrada/salida. En configuraciones más elaboradas puede soportar los protocolos de la capa de intercambio de datos definida en el estándar ISO 7816-3.

Un driver de un terminal integra la funcionalidad de dicho terminal en los servicios nativos que proporciona la plataforma Windows y la infraestructura de tarjeta inteligente. El driver proporciona acceso a la funcionalidad específica de un terminal. Las diferencias entre un terminal simple y uno complejo quedan ocultas por el API del driver. Esta es la capa responsable de proporcionar la interoperabilidad necesaria entre distintos terminales. El driver del terminal comunica los eventos de inserción y retirada de una tarjeta al Resource Manager y proporciona la capacidad de comunicación de datos hacia y desde la tarjeta ya sea usando los protocolos de comunicación síncronos o asíncronos.

Los Smart Card Base Components incluyen una librería común para drivers para que sea usada por los desarrolladores y de esta forma simplificar el desarrollo de nuevos drivers. Esta librería compartida soporta el ISO 7816 y funciones del sistema habituales requeridas para la comunicación de datos entre la tarjeta y el terminal. Esto supone una mejora significativa sobre la forma en que hasta entonces se desarrollaban los drivers porque ahora los desarrolladores tienen unos interfaces estándar sobre los que basarse. Estos interfaces estándar permiten que un driver sea desarrollado de una manera uniforme y sea accesible por todas las aplicaciones, en lugar de sólo por unas pocas que saben cómo comunicarse con un terminal específico.

Los terminales de tarjeta inteligente pueden conectarse al ordenador a través de una variedad de puertos de periféricos estándar, como RS-232, PS/2, PCMCIA y USB. Los terminales de tarjeta inteligente se consideran dispositivos Windows estándar y conllevan un descriptor de seguridad y un identificador PnP. Se controlan a través de drivers de dispositivo estándar de Windows y se instalan y desinstalan del sistema usando el asistente de hardware habitual en Windows.

2.4 Smart Card Resource Manager

El Resource Manager es un componente clave de esta arquitectura y se ejecuta como un servicio privilegiado en un único proceso. Todas las peticiones de acceso a tarjetas inteligentes se realizan a través del Resource Manager y son redirigidas hacia el terminal que contiene la tarjeta inteligente requerida. Por lo tanto, el Resource Manager es el responsable de gestionar y controlar todos los accesos de las aplicaciones a cualquier tarjeta inteligente insertada en cualquier terminal conectado al ordenador. El Resource Manager proporciona a una aplicación una conexión directa virtual con la tarjeta requerida.

El Resource Manager realiza tres tareas básicas para manejar el acceso a múltiples terminales y tarjetas. En primer lugar, identifica y monitoriza los recursos. Esto incluye:

- Monitorizar los terminales instalados y poner esta información al servicio de otras aplicaciones.
- Monitorizar los tipos de tarjeta conocidos junto con sus proveedores de servicio asociados e interfaces soportados y poner esta información al servicio de otras aplicaciones.
- Monitorizar los eventos de inserción y retirada para mantener actualizada la información sobre los terminales y tarjetas disponibles.

En segundo lugar, controla la asignación de terminales y recursos a múltiples aplicaciones.

Esto se consigue proporcionando mecanismos para conectarse a un terminal concreto usando los modos de operación exclusivo o compartido.

Finalmente, soporta primitivas transaccionales para el acceso a los servicios disponibles en una tarjeta concreta. Esto es muy importante porque las tarjetas actuales son dispositivos que soportan un solo thread y que a menudo requieren la ejecución de varios comandos para completar una única función. Los controles transaccionales permiten que se ejecuten múltiples comandos sin interrupción, asegurando que la información de estado intermedia no se corrompa.

El Resource Manager es un componente privilegiado en el sentido de que controla el acceso a los dispositivos físicos y hace de intermediario en el paso de comandos y datos entre una aplicación y una tarjeta. Por tanto, es de crucial importancia que sea diseñado para asegurar una separación lógica entre flujos de datos asociados con diferentes procesos. Debe implementarse de manera que mantenga el mismo grado de seguridad y protección que proporciona el sistema operativo. Es decir, el hecho de añadir este componente a un entorno no debe crear nuevas vulnerabilidades.

2.5 Proveedores de servicios

Los proveedores de servicios son los responsables de encapsular la funcionalidad expuesta por un terminal o tarjeta específicos, haciendo ésta accesible a través de interfaces de programación de alto nivel. Estos interfaces pueden ser mejorados y extendidos para soportar las necesidades de un dominio de aplicación determinado.

Una característica importante de esta especificación que debemos hacer notar es que no obliga a que un proveedor de servicios sea un componente monolítico ejecutándose en un solo ordenador. Es decir, que se podría construir un proveedor de servicios como un componente cliente/servidor, lo que permitiría que una aplicación en un servidor aprovechase los interfaces de alto nivel soportados por esta arquitectura.

Toda tarjeta debe tener, al menos, un proveedor de servicios para que las aplicaciones puedan acceder a los servicios de la tarjeta. Puede haber múltiples proveedores de servicios dependiendo del tipo de tarjeta y del fabricante de la misma. Es bastante común que las tarjetas implementen tres tipos de servicios: servicios de almacenamiento, servicios de autenticación y servicios criptográficos. Estos servicios tienen muchas funcionalidades en común para distintas tarjetas, por lo tanto es beneficioso estandarizar interfaces para estos servicios de forma que el desarrollo y mantenimiento de aplicaciones sea más sencillo.

En general, hay dos tipos de proveedores de servicios: criptográficos y no criptográficos. Esta distinción se hace, básicamente, por las restricciones existentes en cuanto a importación y exportación de tecnología criptográfica impuestas por los gobiernos.

Los Proveedores de Servicios Criptográficos (CSP) pueden estar basados sólo en software, como el Microsoft Base Provider CSP que se distribuye con todas las plataformas Windows, o pueden ser parte de una solución basada en hardware en donde las operaciones criptográficas residen en una tarjeta inteligente (o cualquier otro tipo de hardware similar) conectada al ordenador. Sólo deben proporcionar funcionalidades criptográficas a las aplicaciones, el resto de funcionalidades deben ser implementadas en otros proveedores de servicios.

Un CSP asociado con una tarjeta inteligente recibe el nombre de Proveedor de Servicios Criptográficos de Smart Card (SCCP) para distinguirlos de los CSP más generales. Ambos, CSP y SCCP, encapsulan el acceso a las funcionalidades criptográficas –como generación aleatoria de números, generación de claves, firma digital, intercambio de claves y cifrado– a través de interfaces de programación de alto nivel, como la CryptoAPI.

La librería “CeresCSP.dll” es el proveedor de servicios criptográficos implementado para la tarjeta FNMT-RCM. En ella están implementadas las rutinas que utilizará el CryptoAPI de Windows para realizar operaciones criptográficas sobre tarjeta inteligente.

CeresCSP permite a las aplicaciones basadas en CryptoAPI que usen la tarjeta inteligente FNMT-RCM para realizar firmas digitales, cifrar, descifrar, etc. También proporciona los mecanismos para realizar login con tarjeta en Windows 2000 y Windows XP.

Los Proveedores de Servicios de Smart Card (SCSP) exponen los servicios no criptográficos de la tarjeta inteligente a través de interfaces de alto nivel. Un interface de tarjeta inteligente consiste en un conjunto de servicios predefinido, los protocolos necesarios para invocar a esos servicios y cualquier información concerniente al contexto de los servicios. Por ejemplo, debe incorporar el manejo de conexiones a una tarjeta concreta así como el acceso a los servicios de almacenamiento y autenticación. Además puede implementar el acceso a otras características que el fabricante considere necesarias en el dominio de aplicación. El interface de acceso a los servicios de almacenamiento define mecanismos para realizar las siguientes tareas: localizar ficheros por nombre, crear o abrir ficheros, leer y escribir el contenido de ficheros, cerrar un fichero, borrar un fichero y cambiar sus atributos. El interface de autenticación deber permitir: verificar la identidad del dueño de la tarjeta, verificar la tarjeta y que una aplicación se autentique frente a la tarjeta.

Una tarjeta inteligente registra el soporte para un interface a través de una asociación con el Identificador Único Global (GUID) del interface. En la revisión 1 de la especificación esta asociación entre una tarjeta y un interface se realiza en el momento en que la tarjeta se instala en el sistema –normalmente cuando se instala el SCSP. En la revisión 2 dicha asociación se hace de forma dinámica. Una vez que la tarjeta se ha instalado en el sistema, las aplicaciones pueden buscar tarjetas basándose en un interface específico o GUID. Por ejemplo, una tarjeta monedero puede hacerse accesible para las aplicaciones registrando interfaces para acceder a su sistema de pagos.

Como parte de sus Smart Card Base Components, Microsoft proporciona varios proveedores de servicio básicos para realizar operaciones genéricas, como la búsqueda de tarjetas, el manejo de las mismas a través de protocolos de comando y respuesta y el acceso al sistema de ficheros. Estos proveedores de servicio están implementados como objetos COM, lo que permite tanto a los desarrolladores de software como a los fabricantes de tarjetas construir proveedores de servicio de más alto nivel y aplicaciones.

2.6 Aplicaciones que usan Tarjetas Inteligentes

Una aplicación que usa tarjetas inteligentes es un programa software del tipo que sea que se ejecuta en el sistema operativo del ordenador y que hace uso de la funcionalidad proporcionada por una o más tarjetas. Se supone que la aplicación se ejecuta como un proceso en un entorno multiusuario, multitarea, multithread y con múltiples dispositivos. Los componentes definidos en esta especificación proporcionan los mecanismos para enlazar las peticiones de una aplicación con una tarjeta, que normalmente es un entorno de un solo usuario, un solo thread.

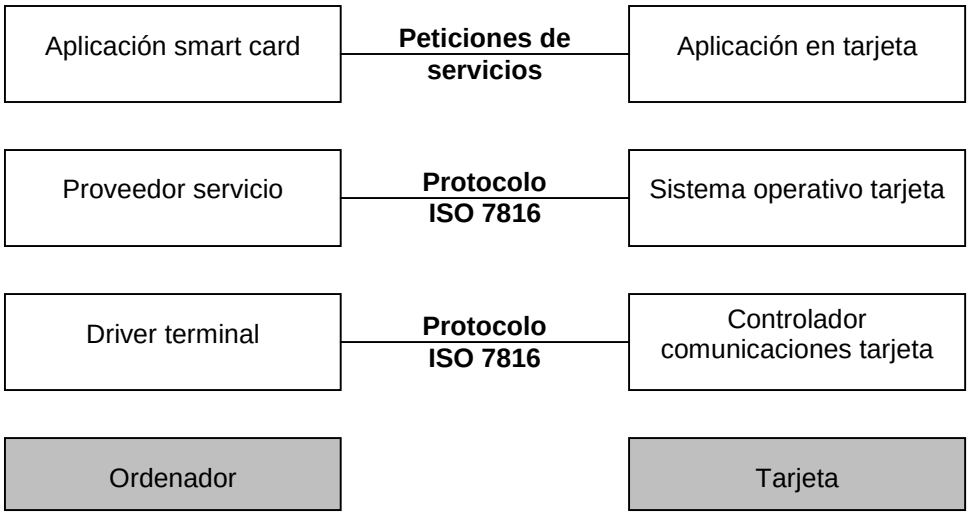
Un desarrollador de aplicaciones tiene tres maneras de usar esta arquitectura:

1. **Usar los SCSP del fabricante**. Cuando los interfaces de los SCSP proporcionados cumplen con los requerimientos de la aplicación, se deben usar los mismos. El fabricante también puede proporcionar interfaces propietarios para manejo de monederos electrónicos, funciones específicas o funciones especiales de seguridad.
2. **Desarrollar un SCSP específico**. Cuando la funcionalidad que se necesita no está definida en los interfaces proporcionados por el fabricante, los desarrolladores pueden construirse su propio SCSP. Éste debe proporcionar, como mínimo, los interfaces estándar definidos por esta especificación.
3. **Acceder al Resource Manager directamente**. Esto es útil cuando la aplicación necesita controlar un terminal o tarjeta a bajo nivel. Este método se reserva para usos muy específicos.

En cualquiera de estos casos, el desarrollador puede acceder al subsistema de manejo de terminales a través de interfaces de alto nivel implementados por el Resource Manager. No se necesita incluir ningún componente con la aplicación. En los casos 1 y 2 porque la

funcionalidad la proporcionan los interfaces del SCSP y en el caso 3 porque se usa el Resource Manager, que como hemos visto es parte del sistema operativo.

Esta arquitectura se puede representar como un protocolo de comunicación punto a punto, como podemos ver en la siguiente figura:



3. Nociones básicas de criptografía[2]

Para lograr entender un poco qué es la criptografía, necesitamos determinar cuales son los problemas que resolvemos gracias a su empleo. Así, los principales problemas que nos permite resolver, junto con un buen uso de las claves y una adecuada regulación legal son:

- La Autenticación
- La Confidencialidad
- La Integridad
- El No Repudio.

La **Autenticación** se define como el proceso por el que se establece la identidad de un software o una persona. En el caso de comunicaciones a través de una red, se trata de establecer de forma fehaciente la identidad de los intervinientes. Este es un punto importante si queremos utilizar la red para algo más que 'chatear', no en vano se dice que los 'chats' son los lugares del mundo donde más hombres usan nombre de mujer ;-). Serán principalmente los mecanismos de la criptografía de clave pública los que nos permitirán abordar con éxito este problema.

La **Confidencialidad** es el proceso mediante el cual logramos que la información únicamente pueda ser accedida por las personas a las que va dirigida o que tienen autorizado su acceso. La información no viaja segura a través de prácticamente ninguna red, será gracias al uso de las técnicas criptográficas, en especial las simétricas, que lograremos esconder la información de miradas indiscretas.

La **Integridad** se refiere a cómo lograr que la información no pueda ser alterada durante la comunicación sin que tengamos conocimiento de ello. Esto es especialmente importante en transacciones comerciales y de tipo financiero.

El **No Repudio** hace referencia al mecanismo necesario para lograr que nadie pueda negar la autoría de los mensajes que ha emitido (no repudio en origen). Este concepto se extiende para imposibilitar la negación de la recepción y acceso a un mensaje, al que efectivamente se ha accedido (no repudio en destino).

La criptografía (simétrica y asimétrica) conjuntamente con el cuidadoso manejo de las claves y una legislación adecuada permitirán resolver satisfactoriamente los problemas que acaban de plantearse. La criptografía se divide en dos grandes ramas, la criptografía de clave privada o simétrica y la criptografía de clave pública o asimétrica.

Ya no es posible controlar un grupo más o menos numeroso de empleados: en los negocios electrónicos se requiere controlar la seguridad en los accesos de usuarios bajo los que no se tiene control administrativo, como proveedores, clientes, consumidores... La PKI (Public Key Infrastructure) es, sobre todo, infraestructura, que las aplicaciones tanto orientadas a Internet como orientadas al 'mundo interior' (bases de datos, sistemas operativos) están tan sólo comenzando a aprovechar.

Bien, ya tenemos una PKI... ¿Y qué? ¿Qué aporta de cara al control de accesos? La tecnología de clave pública ofrece grandes ventajas en cuanto a autenticación fuerte de usuarios (su combinación con dispositivos como las tarjetas inteligentes, de hecho, constituye el estado del arte en cuanto a autenticación de usuarios). Protocolos como SSL o IPSEC se apoyan en la PKI para ofrecer servicios añadidos de confidencialidad e integridad en las comunicaciones. La cuestión es: ¿tienen las PKI algo que ofrecer en cuanto al control de accesos?

Los certificados de atributos no son sino la respuesta de clave pública a los 'certificados de atributos de privilegio' usados en el pasado. Los certificados de clave pública X.509 proporcionan evidencia de la identidad de una persona. Pero en entornos de comercio electrónico, se precisa más información que la mera identidad, en especial cuando las partes involucradas en una transacción no han tenido contacto previo. En este caso, la información

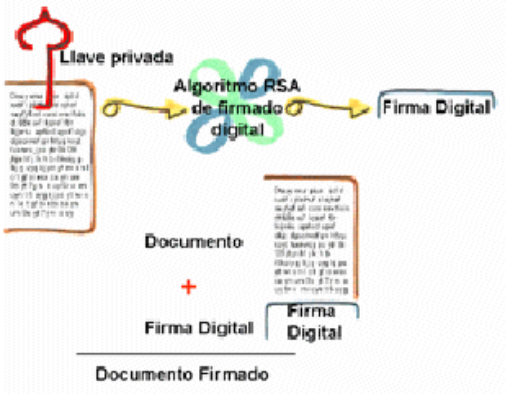
[2] Referencias: [FER105] [FER205] [MEND05] [ANGE00] [RAEF03]

sobre los atributos de privilegio de una persona (por ejemplo, su capacidad de firmar un contrato, o su límite de crédito) es mucho más relevante que su mera identidad.

3.1 Firma digital

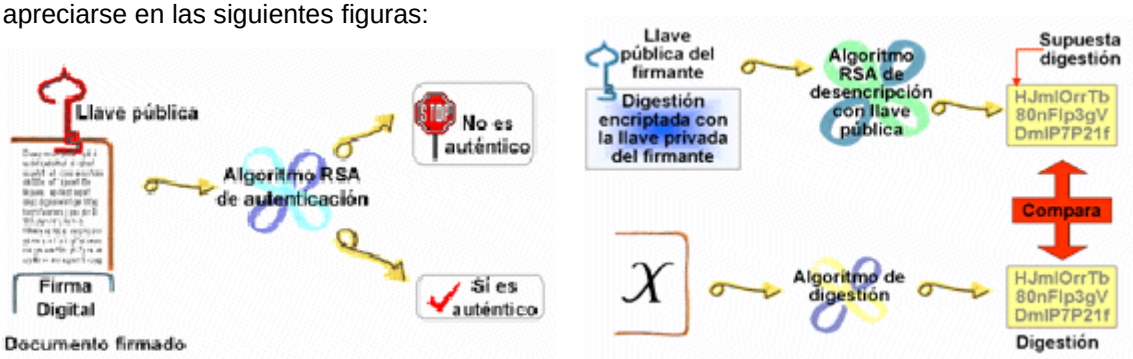
Una Firma Digital es un bloque de caracteres que se introduce o acompaña a un documento o fichero y que sirve para acreditar quién lo ha originado (autenticación) y que no se ha producido ninguna manipulación en su contenido posterior a su firma (integridad).

Para firmar un documento veremos que el firmante debe utilizar su clave secreta, a la que supuestamente sólo él tiene acceso, lo que impide que posteriormente pueda negar la autoría de su firma (no revocación en origen), ya que cualquier persona puede verificar la validez y autoría de una firma, siempre que disponga de la clave pública del firmante. El proceso de firma digital puede verse representado en la siguiente figura:



El software del firmante aplica, de forma transparente para el usuario, una función *Hash* (SHA-1, MD5, etc.) sobre el fichero a firmar, obteniendo un resumen (*digest*) de dicho documento de longitud fija y específico para él, de tal forma que cualquier mínimo cambio en el documento habría dado lugar a un extracto diferente.

El resumen obtenido se somete a continuación a un procedimiento de cifrado mediante un algoritmo asimétrico (RSA, DSA, etc.) con la clave privada o secreta del firmante, para lo cual éste tendrá que introducir su clave de acceso a su clave secreta. El resumen cifrado constituye la firma digital y se añade al documento, o bien se crea un fichero con la firma anexo al documento firmado. El procedimiento de verificación de la validez de la Firma Digital, puede apreciarse en las siguientes figuras:



Para poder llevarlo a cabo, se necesita disponer de la clave pública del firmante. El software del receptor descifra el resumen cifrado que constituye la Firma Digital, de forma transparente al usuario, utilizando para ello la clave pública del remitente. Como resultado de ello obtiene un bloque de caracteres que supuestamente son el resumen del documento firmado. A continuación, calcula el resumen *Hash* correspondiente al documento supuestamente firmado, mediante el mismo algoritmo que se aplicó en origen. Si el resultado coincide exactamente con el bloque de caracteres obtenidos en la operación anterior, la firma se considerará válida. Si existiera la menor diferencia, la firma deberá considerarse inválida, ya que cualquier

modificación del documento enviado, de la propia firma o bien de la clave pública que se utiliza para su descifrado será detectada mediante este proceso, tal y como se muestra en la siguiente figura:

Este esquema cuenta con un punto débil, consustancial a los sistemas de cifrado de clave pública y otro más derivado de su generalización.

El primero consiste en que la firma digital permite comprobar la relación existente entre un mensaje y la clave secreta utilizada para realizar la firma, pero en principio no podemos estar seguros que dicha clave secreta corresponda realmente a la persona o entidad que dice poseerla. Es decir, necesitamos a alguien que garantice la relación entre la clave utilizada y la identidad real del poseedor.

Ante este problema surgen dos enfoques de solución. El primero de ellos se basa en la creación de una jerarquía de autoridades o entidades de registro, que actuarían como terceras partes de confianza en la certificación de la identidad del poseedor de un determinado par de claves. Lo que realiza esta entidad es la firma (con su propia clave secreta) de la clave pública que quiere certificar, así como cierta información adicional, que conformarán lo que se denomina certificado digital. De esta forma se garantiza la conexión existente entre una determinada clave y su propietario real.

El segundo enfoque consiste en la creación de una red de confianza, de forma que sean los propios usuarios los que otorguen confianza a las claves en función del conocimiento de su origen y de la confianza que les merezcan aquellos que firman a su vez otras claves públicas. Este es el enfoque de algunos programas como PGP.

Si se observan con detalle ambos modelos podemos ver que el segundo es más general y contiene al primero. La jerarquía de entidades de certificación constituye un árbol de confianza y, de hecho, un árbol no es más que un tipo de red donde cada nodo tiene un único padre. La diferencia fundamental ente ambos enfoques, consiste en que en la primera aproximación hay una centralización de dicha función (más apropiada para entornos comerciales), mientras que en la segunda, la confianza se otorga de forma descentralizada (más orientada al uso personal).

El segundo de los problemas de la firma digital surge con su extensión. Si necesitamos verificar la firma de varios remitentes, tendremos que tener acceso a las claves públicas de cada uno de ellos. Surge, pues, la necesidad de contar con algún lugar donde poder hallar dichas claves. De forma que no puedan ser suplantadas. Esta tarea de gestionar las claves públicas, o más bien los certificados, es la que se atribuye a las autoridades o entidades de certificación.

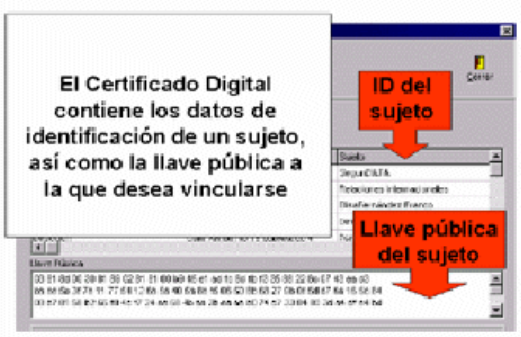
Aunque existan entidades que asuman ambas funciones, la de registro y la de certificación, tales como Verisign o Thawte a escala mundial o ACE y FESTE en España, lo cierto es que se trata de funciones diferentes que también podrían desarrollarse de forma separada, tal y como estaba previsto en el modelo de CERES de la Real Fábrica Nacional de la Moneda y Timbre (FNMT). En CERES, al menos de forma teórica y en su concepción más general, el Organismo Autónomo de Correos y Telégrafos (OACT) actuaría como entidad de registro y la propia fábrica como entidad de certificación. De hecho, en los modelos de funcionamiento actuales en el sector público, ambas funciones se separan, al menos parcialmente. Tanto la Agencia Estatal de la Administración Tributaria (AEAT), como la Tesorería General de la Seguridad

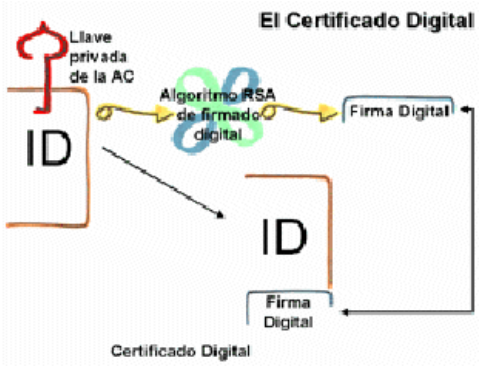
Social, actúan como entidades de registro en sus respectivos procedimientos electrónicos con el ciudadano y utilizan los servicios de certificación de la FNMT.

Los procedimientos de registro parten de la generación de un precertificado que, según el caso, puede generar el propio interesado con su navegador (al menos la generación del par o pares de claves asimétricas). Este precertificado (que contiene al menos la clave pública que quiere certificarse) se envía firmado a la entidad de registro, o bien a la propia entidad de certificación (como en el caso de la AEAT), obteniendo un número identificativo de la transacción realizada (un resumen o MAC). La acreditación ante la entidad de registro normalmente exige la presencia física de la persona y la presentación de la documentación acreditativa de su personalidad y capacidades. La entidad de registro verifica la documentación presentada y si está todo conforme envía un mensaje firmado a la entidad de certificación habilitando el certificado correspondiente al MAC aportado, de forma que dicha entidad informa al usuario que puede descargar su certificado firmado por la entidad cuando quiera y con las adecuadas medidas de seguridad. Esta somera descripción del procedimiento de registro no es exhaustiva, ni corresponde a la única posibilidad de implantación, sino que pretende ser ilustrativa de alguno de los procedimientos que se realizan hoy en día.

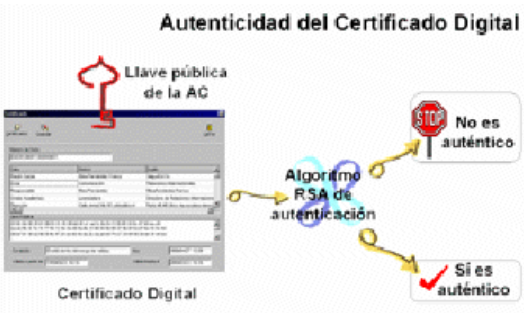
3.2 Certificado digital

Hemos visto la necesidad de contar con la identidad certificada de la clave pública para una adecuada autenticación de la Firma Digital. Igualmente, se ha visto la necesidad de conocer algunos parámetros, como la función *Hash* utilizada, su versión etc. para poder realizar adecuadamente el procedimiento de verificación de la Firma Digital. Todo ello nos conduce a la necesidad de gestionar no tan solo la clave pública de una persona, sino un conjunto más amplio de informaciones acerca de la identificación del poseedor de la clave y de la entidad de certificación, el período de validez del certificado, los parámetros de su generación (algoritmos, versiones, etc.), así como otra serie de atributos inherentes al poseedor de la clave y que puedan ser de utilidad para otras aplicaciones que necesiten de una autenticación fuerte del usuario. Toda esta información estructurada, estandarizada (X.509 v.3) y firmada por una entidad de certificación es lo que va a constituir nuestro Certificado Digital, tal y como se aprecia en las siguientes figuras:

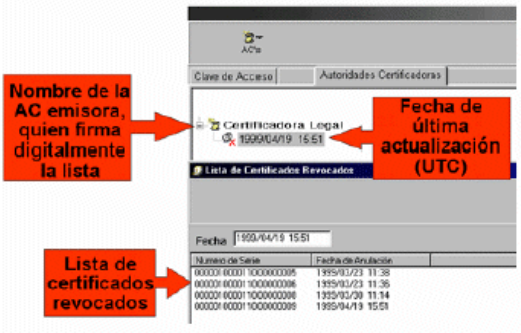




Gracias a la introducción de este nuevo elemento, es posible anexar al documento firmado una copia de nuestro certificado digital de forma que la comprobación de la firma del documento será más segura, ya que el receptor podrá comprobar la validez del certificado previamente a comprobar la validez de la firma, tal y como podemos ver en las siguientes figuras:

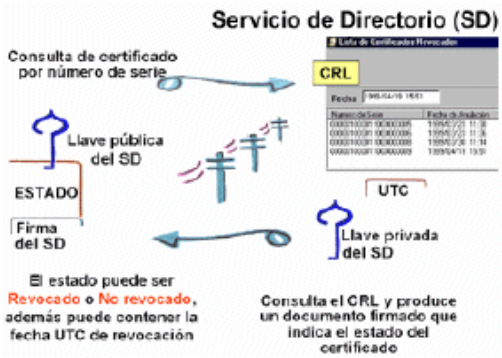


Sin embargo, dado que la vida no es estática, nuestros certificados digitales tampoco pueden serlo, por lo que debemos introducir en ellos conceptos como el período de validez, motivado por la evolución tecnológica, así como la capacidad de revocación de un certificado digital por parte de su legítimo propietario, dado que siempre existe la posibilidad de pérdida o sustracción de la clave secreta que le da sentido. Fruto de la necesidad anteriormente expuesta, se va a requerir de nuestra entidad de certificación un nuevo servicio que nos permita verificar si el certificado que tenemos o que nos ha llegado, se encuentra actualmente en vigor. Para ello las entidades de certificación cuentan con dos mecanismos. El primero consiste en la emisión de listas de revocación de certificados (CRL es su acrónimo en inglés) cuyo formato se puede apreciar en la siguiente figura:



Se trata, pues, de una lista firmada por la entidad de certificación en la que aparecen los números de serie de los certificados que han sido revocados, junto con su fecha de revocación. De esta forma cualquiera puede comprobar si el certificado correspondiente a la firma que se comprueba estaba en vigor en el momento de su recepción.

Sin embargo, este enfoque cuenta con dos problemas. El primero es la gestión de las listas de revocación que se hacen cada vez más voluminosas. El segundo es el desfase temporal existente siempre entre el momento de emisión de una lista y el de su consulta. Como este elemento es crítico en ciertos entornos como el financiero, surge la necesidad de proporcionar un servicio de consulta de directorio en línea, que permita a cualquiera enviar un mensaje de consulta acerca de la validez de un determinado certificado digital identificado por su número de serie y obtener como respuesta un mensaje firmado por la entidad de certificación con la indicación del estado de dicho certificado y la fecha de la consulta, tal y como podemos ver en la siguiente figura:

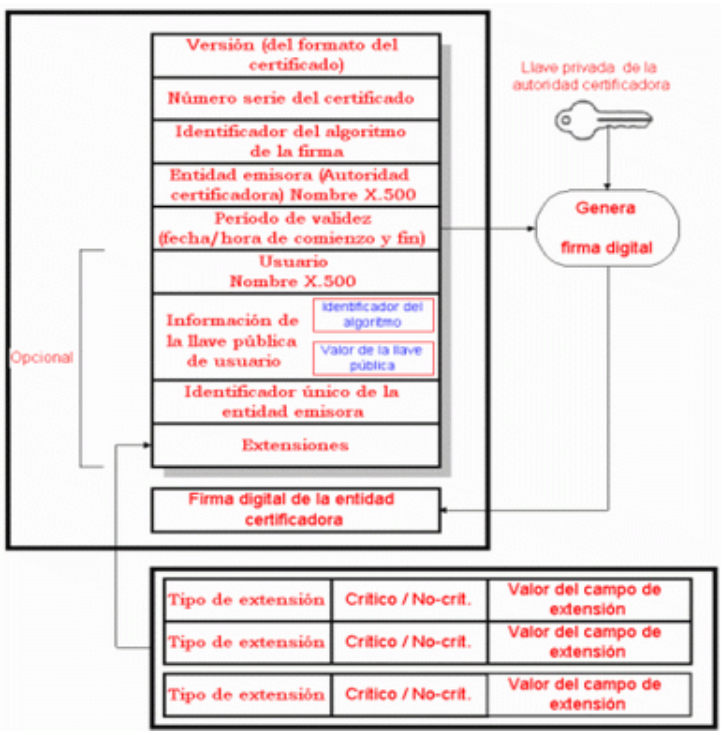


Como vemos, la realidad va complicando cada vez más el esquema inicial de intercambio de documentos. De hecho, aún con la inclusión de los certificados y los servicios de consulta de directorio de las entidades de certificación, el emisor del documento sigue sin poder demostrar que el receptor lo ha recibido y accedido a su contenido, con lo que queda en situación de debilidad frente al receptor. Para lograr una cierta situación de equilibrio necesitamos la participación de una tercera parte de confianza, que podría ser la misma entidad de certificación o, en una versión más débil, un sistema automático de acuse de recibo. Cualquiera de ellos deberá proporcionar al emisor del documento un resumen del documento recibido firmado por el receptor (y/o la tercera parte de confianza) que incluya la fecha de recepción (*time stamping*). Un planteamiento completo de la solución de este problema sobrepasa el objetivo de este documento. La potencia de estas herramientas criptográficas es tan grande

que existe una fuerte preocupación en los gobiernos por su utilización para encubrir o realizar actos delictivos con su ayuda. Esta es la razón por la que muchas entidades de certificación opten por mecanismos de doble par de claves. Un par de claves se garantiza que únicamente está en posesión del usuario y tan solo puede utilizarse para firmar documentos. El segundo par de claves cuenta con una copia que se deposita normalmente en la entidad de certificación (*key scrow*) y que se utiliza para garantizar la confidencialidad en la transmisión de la información.

El estándar internacionalmente aceptado para Certificados Digitales es el denominado X.509, en su versión 3. La primera versión de X.509 apareció en 1988 y fue publicada como el formato X.509v1, siendo la propuesta más antigua para una infraestructura de clave pública (PKI) a nivel mundial. Esto, junto con su origen ISO/ITU han hecho de X.509 el PKI más ampliamente utilizado. Más tarde fue ampliada en 1993 por la versión 2 únicamente en dos campos, identificando de forma única el emisor y usuario del certificado. La versión 3 introduce cambios significativos en el estándar. El cambio fundamental es el hacer el formato de los certificados y las CRLs extensible. Ahora los que implementen X.509 pueden definir el contenido de los certificados como crean conveniente. Además, se han definido extensiones estándares para proveer una funcionalidad mejorada.

En la siguiente figura podemos apreciar la estructura de un certificado X.509v3:



Los principales campos de un certificado X.509 son:

- Versión del certificado. Identifica la versión del X.509 que define los elementos del certificado. Normalmente será versión 3.
- Número de serie. Es asignado por el emisor del certificado, por lo que es único.
- Identificador del algoritmo de firma. Sirve única y exclusivamente para identificar el algoritmo usado para firmar el paquete X.509.
- Autoridad certificadora. Es el nombre de la autoridad que emite el certificado. De acuerdo con el estándar X.500 dicho nombre es único.
- Período de validez. El período de tiempo durante el cual el certificado es válido.
- Nombre del propietario. Es el nombre de la entidad cuya clave pública se está firmando y debe reconocerse como auténtica para el certificado. Según el estándar X.500 dicho nombre es único.
- Clave pública. Es la clave pública del propietario del certificado que se debe reconocer como auténtica, más el identificador del algoritmo utilizado y más parámetros si son necesarios.

- Firma digital. Es la firma digital de los datos del certificado con la clave privada de la autoridad certificadora.

X.509 proporciona tres procedimientos alternativos para la autenticación en peticiones de servicio, mensajes o envío de información. En todos estos procedimientos, se supone que las dos partes conocen la clave pública de la otra, bien porque la han obtenido de un directorio, o bien porque en el mensaje inicial va incluida.

Autenticación en una vía. El terminal genera un número aleatorio y lo envía a la tarjeta. Esto es lo que llamamos "el desafío". La tarjeta cifra el número que recibe usando una clave conocida por el terminal y la tarjeta. La seguridad del procedimiento depende de esta clave, ya que sólo el poseedor de dicha clave puede generar la respuesta correcta. Después la tarjeta envía esa respuesta al terminal. Ésta es la respuesta al desafío. En el terminal se comparan ambos números y si son iguales la tarjeta es auténtica. En la práctica para evitar que todas las tarjetas tengan la misma clave, lo cual es un problema de seguridad, se cifra parte del número de serie de la tarjeta usando una clave maestra y el resultado es lo que se usa como la clave de autenticación específica de la tarjeta. El envío de información de A a B, define:

- La identidad de A y que el mensaje fue generado por A
- Que el mensaje estaba dirigido a B
- La integridad y unicidad del mensaje.

Autenticación en dos vías. Se usan dos autenticaciones unilaterales sucesivas, una para cada parte de la comunicación para obtener la autenticación mutua. Para minimizar el tiempo utilizado ambas comunicaciones se simultanean. El terminal le pide a la tarjeta su número de serie y un número aleatorio. El terminal genera su número aleatorio y cifra ambos con la clave específica de la tarjeta. La tarjeta descifra el mensaje y comprueba que en él esté el número aleatorio que había enviado, en cuyo caso confirma que el terminal tiene su clave específica, lo que le autentica frente a la tarjeta. La tarjeta intercambia ambos números y los cifra con su clave enviándolos al terminal. Define además de los anteriores:

- La identidad de B, y que el mensaje fue generado por B
- Que el mensaje estaba dirigido a A
- La integridad y unicidad del segundo mensaje.

Autenticación en tres vías. La autenticación de tres vías elimina la necesidad de que el destino y el iniciador tengan sus relojes sincronizados. Después de pasar por la autenticación de dos vías, el iniciador envía entonces un mensaje a la respuesta del destino incluyendo el nuevo testigo contenido en la respuesta original. Después de verificar que los valores del testigo son idénticos, ya no hay necesidad de verificar las marcas de tiempo.

4. Soluciones basadas en Tarjetas Inteligentes[3]

Al efectuar una operación comercial por Internet se presentan nuevos problemas, por ejemplo cómo saber que la tienda virtual existe verdaderamente, una vez hecho el pedido cómo saber que no se cambia la información, cuando se envía el número de tarjeta de crédito cómo saber si éste permanecerá privado, en fin, para el comerciante también se presentan problemas similares, cómo saber que el cliente es honesto y no envía información falsa, etc. Todos estos problemas pueden ser resueltos de manera satisfactoria si se implementan protocolos de comunicación segura usando criptografía. En la siguiente sección nos dedicamos a describir como es que estos protocolos resuelven los problemas planteados.

Dado que mejoran las soluciones basadas sólo en software, como autenticación de clientes y mensajería segura, las tarjetas inteligentes permiten que una nueva clase de aplicaciones se posicionen para sacar ventaja de futuras oportunidades en el mundo de la emergente economía digital global. Las tarjetas inteligentes ofrecen a los desarrolladores de aplicaciones un mecanismo seguro para proporcionar mejores soluciones, tanto para la empresa como para el consumidor.

Las posibles aplicaciones para las tarjetas inteligentes incluyen identificación, sistemas de control de accesos a zonas o máquinas restringidas, almacenamiento seguro de datos, etc. Las tarjetas multifuncionales incorporan varias aplicaciones en una sola tarjeta. Los modernos sistemas operativos de tarjetas inteligentes permiten cargar nuevas aplicaciones en la tarjeta una vez que le ha sido entregada al usuario sin comprometer la seguridad de las demás aplicaciones. Esta flexibilidad abre un mundo nuevo de áreas de aplicación. Por ejemplo, se hacen indispensables módulos personales de seguridad si se quiere que el comercio y los pagos realizados a través de Internet sean seguros. Por ese motivo se están desarrollando especificaciones para aplicaciones de Internet seguras utilizando tarjetas inteligentes en todo el mundo. En pocos años podemos esperar ver cada PC equipado con un terminal de tarjetas inteligentes.

4.1 Securización de mensajes

El correo electrónico seguro es una de las aplicaciones más interesantes de la criptografía de clave pública porque permite a los usuarios intercambiar información de manera confidencial y con la confianza de que la integridad de la información se ha mantenido intacta durante la transmisión. Usando cualquiera de los programas de gestión de correo electrónico más populares hoy en día, el usuario puede escoger un certificado de clave pública de una entidad de confianza para firmar digitalmente y descifrar mensajes seguros. Cuando un usuario publica su certificado en un directorio público en el rango de la empresa o en Internet, otros usuarios de la empresa o de toda la red pueden enviar mensajes cifrados a ese usuario y viceversa.

Una tarjeta inteligente añade un nivel de integridad a las aplicaciones de correo seguro porque almacena la clave privada en la tarjeta, protegida por un PIN. Para que la clave privada se vea comprometida y poder enviar un mensaje firmado con ella a alguien, el intruso tendría que ser capaz de obtener la tarjeta inteligente y el PIN. El PIN podría ser, en un futuro no muy lejano, sustituido por una plantilla biométrica de la huella digital del usuario y de esta forma se mejorarían los criterios de no repudio de un correo firmado digitalmente.

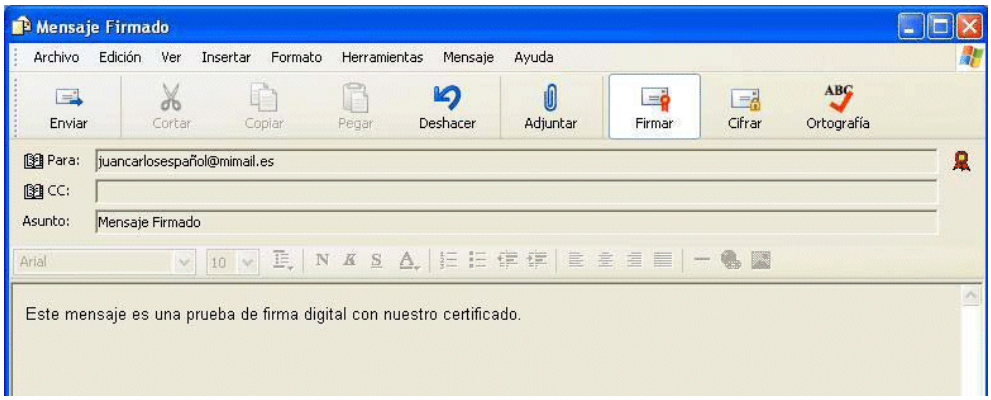
El intercambio seguro de información se basa en el uso de las claves públicas y privadas de usuarios. Las claves públicas son accesibles a todo el universo de usuarios y se envían junto con el certificado digital. En cambio las claves privadas son secretas y no pueden ser extraídas de la tarjeta. Outlook Express utiliza CryptoAPI, con lo que para realizar operaciones criptográficas usará la librería CeresCSP.dll. Para utilizar todas las posibilidades del correo seguro entre dos usuarios es necesario que cada uno tenga el certificado del otro con sus claves públicas. La manera de obtenerlo es recibiendo un mensaje firmado, quedando así registrado en la libreta de direcciones.

[3] Referencias: [CERE03] [ANGE00]

Para realizar una firma digital, el usuario utilizará su clave privada que se encuentra almacenada, junto con el resto de componentes del perfil criptográfico, en la tarjeta FNMT-RCM. Esa clave no puede ser extraída, por lo que no pueden existir copias de ella. Cuando queramos firmar un mensaje, la aplicación de correo nos solicitará la tarjeta cuyo número de serie coincida con el de aquella que contiene el certificado asociado a esa cuenta. Además, será necesario presentar el PIN de la tarjeta para acceder y utilizar la clave privada. De esta manera, si la tarjeta cayese en manos de otro individuo no podría realizar ninguna firma ya que no debería conocer su PIN.

Si queremos enviar un **correo firmado** a otro usuario, crearemos un nuevo mensaje de la forma habitual. Una vez que hayamos escrito el mensaje tenemos varias formas de indicar que queremos firmarlo. Podemos acceder al menú “Herramientas” y ahí seleccionar la opción “Firmar digitalmente”, o bien pulsar el botón “Firmar” de la barra de herramientas.

En la siguiente figura vemos un mensaje que va a ser firmado digitalmente utilizando el certificado que se seleccionó previamente para esa cuenta. En la imagen vemos que ha quedado pulsado el botón de “Firmar” y ha aparecido en la parte derecha un icono mostrando un pequeño sello rojo, indicando que el mensaje va a ser firmado antes de ser enviado. Cuando finalicemos el mensaje lo enviaremos normalmente y la aplicación de correo realizará las operaciones de firma.

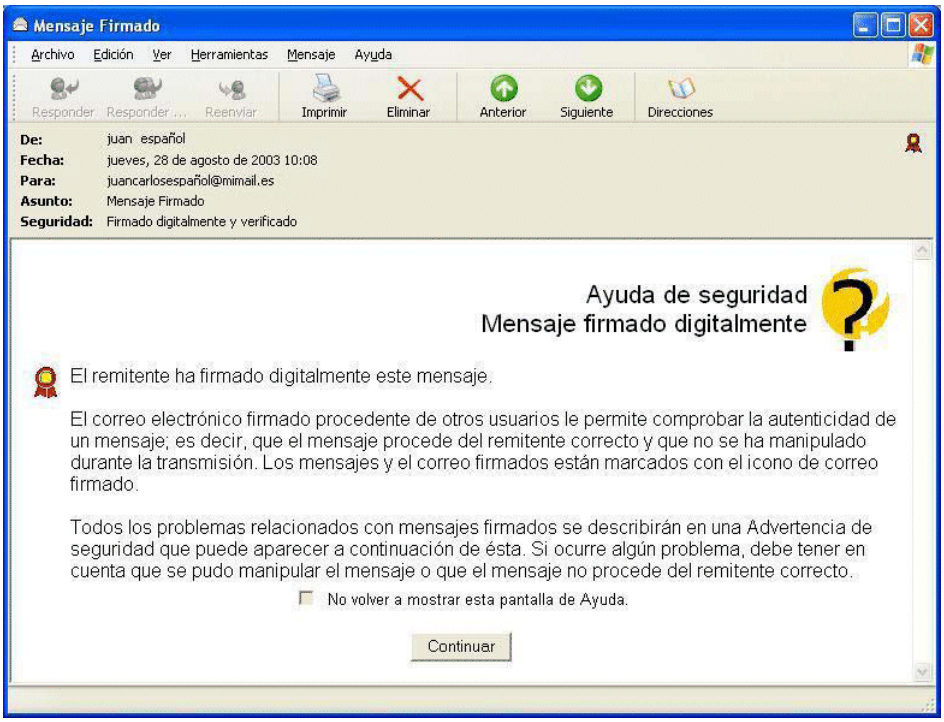


Como dijimos antes, para poder firmar digitalmente un mensaje es necesario usar nuestra clave privada, almacenada en la tarjeta, y presentar el PIN para tener acceso a ella. Por esto es imprescindible que la tarjeta que contiene el certificado se encuentre insertada en el lector. En caso de que no sea así aparecerá en pantalla una ventana solicitando la tarjeta correcta, cuyo número de serie debe coincidir con el indicado.

Cuando la tarjeta esté disponible se solicitará su PIN. Será entonces cuando se realicen las operaciones de firma digital y se envíe el mensaje al destinatario, incluyendo el certificado del emisor.

En el cliente de correo del receptor aparecerá un nuevo mensaje con un icono que indica que dicho mensaje está firmado digitalmente. Al incluirse el certificado en el propio mensaje, cuando lo abramos se usará para verificar la firma e indicarnos si es correcta o no.

Si abrimos el mensaje firmado aparecerá una pantalla previa en la que Outlook Express nos indica que el mensaje ha sido firmado digitalmente. Tras pulsar el botón de “Continuar”, se usará el certificado para comprobar la autenticidad de la firma. Si puede verificar la firma recibida, el mensaje aparecerá normalmente. Si no, nos advertirá que no pudo ser verificada, aunque nos mostrará el texto del mensaje.

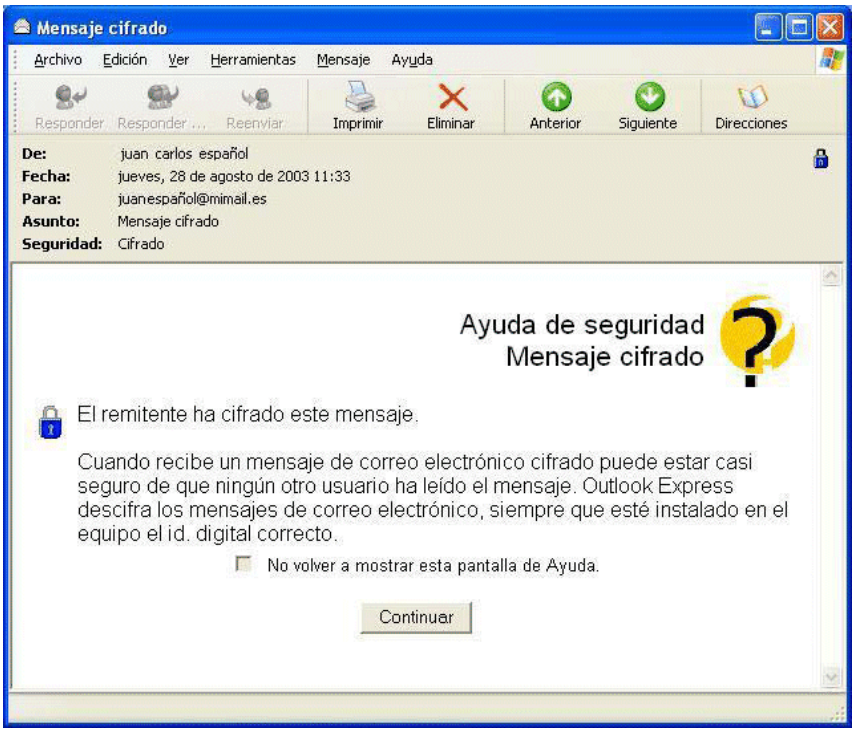


Todo lo anterior era referente a la firma digital. Ahora comenzaremos con el **cifrado y el descifrado de mensajes**. Sabemos que para poder cifrar un mensaje debemos utilizar la clave pública del receptor. Esa clave pública se envía junto con el certificado, por lo que hasta que no lo tengamos no podremos enviarle un mensaje cifrado a ese usuario. Una manera de obtener el certificado del receptor es recibir previamente un mensaje firmado por él y que lo incluya. De esta manera al añadirse a nuestra libreta de contactos, quedará reflejado el certificado asociado a ese usuario y que se usará para las operaciones criptográficas que lo requieran.

Cuando queramos cifrar un mensaje actuaremos normalmente, pero antes de enviarlo pulsaremos el botón de “Cifrar”, o bien lo indicaremos desde el menú de “Herramientas” del mensaje.

En el mensaje aparecerá un pequeño icono de un candado en la parte derecha que indica que al enviar se va a realizar el cifrado del mensaje. Al contrario que con la firma digital, ahora no es necesario tener insertada nuestra tarjeta inteligente ya que la clave que se va a usar es la del certificado del receptor.

Cuando el receptor del mensaje reciba el correo cifrado, le aparecerá marcado con el icono correspondiente. Cuando se abra el mensaje nos aparecerá una advertencia indicándonos que ese mensaje está cifrado.



Para poder abrirlo necesitaremos nuestra tarjeta FNMT-RCM en la que se encuentra nuestra clave privada, necesaria para descifrar el mensaje.

Cuando queramos enviar un correo que esté a la vez cifrado y firmado realizaremos las mismas operaciones descritas anteriormente. Al realizarse una operación de firma digital y otra de cifrado, será necesario tener insertada la tarjeta con nuestra clave privada y además tener instalado el certificado del receptor del mensaje. Antes de enviar el correo pulsaremos los botones de “Firmar” y “Cifrar”, mostrándose en este caso los iconos de las dos operaciones.

Al recibir el mensaje veremos que el icono que aparece junto a él es solamente el de cifrado. Esto es debido a que el descifrado es la primera operación que se llevará a cabo, para después proceder con la verificación de la firma digital. Para abrirlo necesitaremos nuestra tarjeta FNMT-RCM con nuestro certificado almacenado para descifrar el mensaje, además del certificado del emisor para poder verificar su firma digital.

4.2 Autenticación de cliente

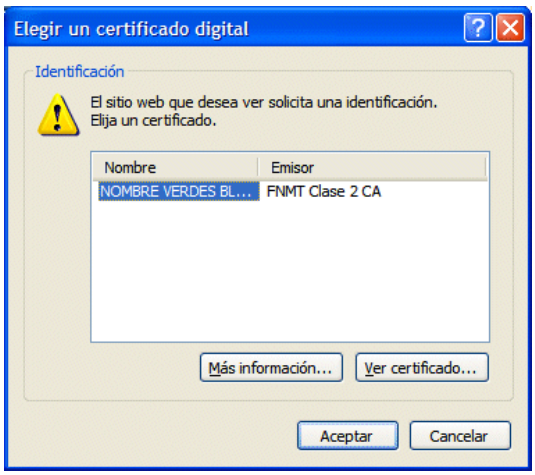
La autenticación de cliente implica la identificación y la validación de la identidad del usuario frente a un servidor remoto estableciendo para ello un canal de comunicación seguro. Normalmente se usa un protocolo seguro, como Secure Sockets Layer (SSL) o Transport Layer Security (TLS), conjuntamente con un certificado de clave pública proporcionado por el cliente y en el que se confía, para identificar al cliente frente al servidor. El cliente puede ser cualquier navegador web y el servidor cualquiera que soporte SSL/TLS.

La sesión segura se establece usando autenticación de clave pública con intercambio de claves para obtener una clave de sesión única que podrá ser usada para asegurar la integridad de los datos y la confidencialidad a lo largo de la sesión. Se puede conseguir un mayor grado de autenticación si se asocia el certificado con una cuenta de usuario o grupo en la que previamente se han establecido privilegios de acceso.

La tarjeta inteligente mejora el proceso de autenticación mediante clave pública actuando como un almacén seguro para la clave privada y como una herramienta criptográfica para realizar operaciones de firma digital e intercambio de claves.

Para que un certificado almacenado en la tarjeta FNMT-RCM sea accesible a las aplicaciones

de Microsoft, debe estar también instalado en el sistema. Cuando una aplicación Web requiera una conexión segura, pasará el control al navegador, quien nos mostrará la ventana de certificados instalados, donde podremos seleccionar aquel con el que queremos conectarnos.



Después, el navegador comprobará si ese certificado está en software o si, por el contrario, está asociado a alguna tarjeta inteligente. En este último caso nos pedirá que introduzcamos la tarjeta con el número de serie correspondiente.

Una vez introducida la tarjeta correcta en el terminal, la aplicación pasará a realizar las operaciones necesarias para establecer la comunicación segura. Para ello utilizará las claves almacenadas en la tarjeta FNMT-RCM, así que de nuevo nos solicitará el PIN.

En caso de no autenticarnos contra la tarjeta o que el certificado no esté correctamente instalado en la misma, el navegador no será capaz de establecer la conexión y mostrará un mensaje de error informándonos de ello. Si la conexión se realiza correctamente, en la barra de estado inferior del navegador aparecerá un icono de un pequeño candado que indica que hay establecida una conexión segura.

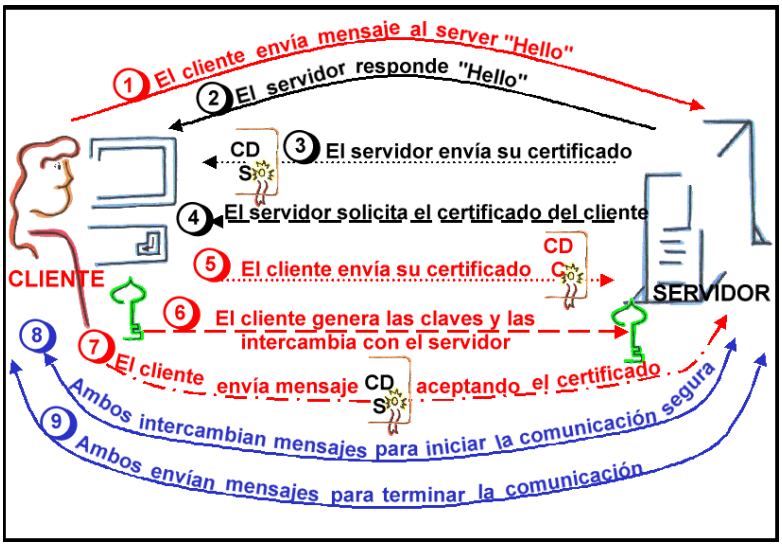
SSL es el protocolo de comunicación segura más conocido y usado actualmente, SSL actúa en la capa de comunicación y es como un túnel que protege a toda la información enviada y recibida. SSL es usado en gran cantidad de aplicaciones que requieren proteger la comunicación. Con SSL se pueden usar diferentes algoritmos para las diferentes aplicaciones, por ejemplo usa DES, TDES, RC2, RC4, MD5, SHA-1, DH y RSA. Cuando una comunicación está bajo SSL la información que se cifra es:

- La URL del documento requerido
- El contenido del documento requerido
- El contenido de cualquier formulario requerido
- Las “cookies” enviadas del navegador al servidor
- Las “cookies” enviadas del servidor al navegador
- El contenido de las cabeceras de los http

La versión más actual de SSL es la v3, existe otro protocolo parecido a SSL sólo que está desarrollado por IETF que se denomina TLS (Transport Layer Security Protocol) y difiere en que usa un conjunto un poco más amplio de algoritmos criptográficos. Por otra parte existe también SSL plus, un protocolo que extiende las capacidades de SSL y su principal característica es que es interoperable con RSA, DSA/DH y CE (Criptografía Elíptica).

Este protocolo está especialmente diseñado para asegurar las transacciones por Internet que se pagan con tarjeta de crédito. Esto es debido a que una gran cantidad de transacciones de compra por Internet son efectuadas con tarjeta de crédito, por otro lado SSL deja al descubierto alguna información sensible cuando se usa para lo mismo.

El procedimiento que se lleva a cabo para establecer una comunicación segura con SSL es el siguiente:



1. El cliente (navegador) envía un mensaje de saludo al servidor “ClientHello”
2. El servidor responde con un mensaje “ServerHello”
3. El servidor envía su certificado
4. El servidor solicita el certificado del cliente
5. El cliente envía su certificado: si es válido continúa la comunicación, si no para o sigue la comunicación sin certificado del cliente
6. El cliente envía un mensaje “ClientKeyExchange” solicitando un intercambio de claves simétricas si es el caso
7. El cliente envía un mensaje “CertificateVerify” si se ha verificado el certificado del servidor, en caso de que el cliente esté en estado de autenticado
8. Ambos, cliente y servidor, envían un mensaje “ChangeCipherSpec” que significa el comienzo de la comunicación segura
9. Al término de la comunicación ambos envían el mensaje “finished” con lo que termina la comunicación segura, este mensaje consiste en un intercambio del *hash* de toda la conversación, de manera que ambos están seguros que los mensajes fueron recibidos intactos (íntegros).

4.3 Logon en Windows

Windows 2000 y Windows XP soportan la autenticación de usuario mediante clave pública y usan un certificado X.509 v3 almacenado en una tarjeta inteligente junto con su clave privada. En lugar de una contraseña el usuario introduce su PIN en el sistema de identificación, este PIN se usa para autenticar al usuario frente a la tarjeta.

Conectarse a una red con una tarjeta inteligente proporciona un mecanismo seguro de autenticar un usuario en un dominio porque se usa una identificación basada en criptografía junto con una prueba de propiedad. Por ejemplo, si un atacante obtiene la contraseña de un usuario, esa persona puede suplantar la identidad del usuario en la red simplemente usando esa contraseña. Mucha gente utiliza contraseñas que puede recordar fácilmente, lo que hace a las contraseñas inherentemente débiles y susceptibles a ataques.

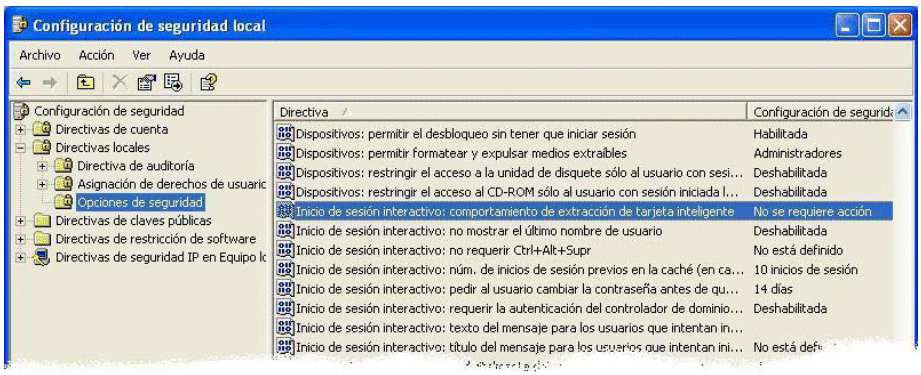
En el caso de una tarjeta inteligente, ese mismo atacante tendría que obtener tanto la tarjeta del usuario como su PIN para poder suplantarlo. Esta combinación hace que el ataque sea menos probable porque de entrada hay que tener acceso físico a la tarjeta del usuario. Otro beneficio añadido es que una tarjeta inteligente quedará bloqueada después de que se introduzca erróneamente el PIN varias veces, lo que hace extremadamente difícil un ataque basado en diccionario contra la tarjeta (hay que aclarar que el PIN no tiene por qué ser una serie de números; también se pueden usar otros caracteres alfanuméricos). Los ataques contra

una tarjeta no pasarán inadvertidos, porque el atacante debe estar en posesión de dicha tarjeta y su legítimo propietario la echará en falta.

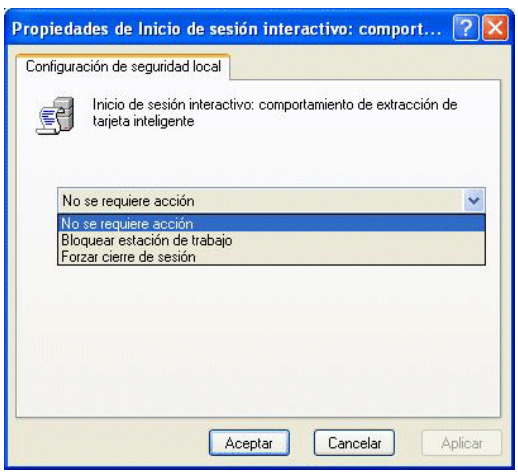
Al iniciar una nueva sesión, el sistema operativo nos muestra una ventana en la que nos indica las acciones que podemos realizar para arrancarla. Si pulsamos Ctrl+Alt+Supr se mostrará el habitual cuadro de identificación de usuario, dominio y contraseña. En cambio, si insertamos la tarjeta aparecerá una ventana en la que se nos solicitará que introduzcamos su PIN.

Para conseguir logon con tarjeta debemos estar en posesión de un certificado emitido por Active Directory de Microsoft. Si tuviésemos más de un certificado en la tarjeta se comprobaría si su certificado por defecto permite realizar el *logon*. En caso de que no sea así, se mostrará una ventana en la que deberemos seleccionar el certificado con el que autenticarnos. Una vez validado el usuario, se iniciará la sesión normalmente.

Al solicitar este tipo de arranque con tarjeta aparece la duda de saber qué ocurre cuando ésta es extraída. Es decir, si el equipo debe quedar bloqueado, si la sesión debe cerrarse, si no debe ocurrir nada, etc. Estas acciones posteriores a la extracción de la tarjeta son configurables. Para modificarlo debemos acudir al "Panel de Control" y seleccionar "Herramientas Administrativas". Cuando se abra la ventana de las herramientas administrativas aparecerán una serie de opciones de configuración. La que nos interesa en este caso es la de: "Inicio de sesión interactivo: comportamiento de extracción de tarjeta inteligente". En la siguiente figura podemos ver la pantalla de configuración que debemos modificar:



Aquí podremos indicar qué acción queremos que se lleve a cabo al extraer la tarjeta con la que hicimos *logon*. Estas opciones son tres: o no ejecutar ninguna acción, o bloquear la estación de trabajo, o forzar el reinicio de la sesión. En la figura vemos cómo podemos seleccionar entre las tres.



A la hora de realizar *logon* con tarjeta en Windows 2000, Windows NT o Windows XP, el certificado que se utilizará para la autenticación será el propio de Microsoft. La aplicación "ccmng.exe" permite seleccionar el certificado que queremos establecer por defecto en la

tarjeta FNMT-RCM. De esta manera evitaremos tener que seleccionarlo para el *logon* cada vez que queramos iniciar una sesión.

Cuando ejecutemos esta aplicación aparecerá una ventana solicitándonos que insertemos la tarjeta en el lector. En cualquier caso la aplicación continúa con la ejecución normal, solicitándonos el PIN de la tarjeta.

Una vez tecleado el PIN, la aplicación comprobará si la tarjeta está insertada en el lector. En caso de que no sea así, se nos mostrará una ventana de error y la aplicación se detendrá. Si hemos introducido el PIN correctamente, aparecerá en pantalla una última ventana que nos permitirá seleccionar qué certificado queremos dejar por defecto en la tarjeta. Simplemente debemos pulsar en el que hayamos elegido y pulsar el botón de “Aceptar”.

Una vez hecho esto, el certificado seleccionado quedará asignado como certificado por defecto para la tarjeta.

Durante un *logon* con tarjeta inteligente, se recupera el certificado de clave pública del usuario de la tarjeta a través de un proceso seguro y se verifica que sea válido y que proceda de una autoridad de confianza. Durante el proceso de autenticación se envía a la tarjeta un desafío, basado en la clave pública contenida en el certificado, para verificar que la tarjeta posee y puede usar la correspondiente clave privada. Una vez que se verifica el par clave pública-privada, se usa la identidad de usuario que contiene el certificado para referenciar un objeto de usuario almacenado en el Active Directory para construir un *token* y enviar al cliente un Ticket-Granting Ticket (TGT). El mecanismo de clave pública se ha integrado con la implementación de Microsoft de Kerberos.

Para poder implementar este mecanismo en nuestro entorno debemos asegurarnos de cumplir con unos requisitos. En cuanto al hardware, los dispositivos deben cumplir con la especificación PC/SC. En cuanto al software debemos tener un servidor con los siguientes componentes instalados:

- El Certificate Authentication Service. Con este servicio, Windows puede crear certificados que se instalarán en la tarjeta inteligente. Un servidor corriendo este servicio se denomina Certificate Authority (CA)
- Microsoft Active Directory. Este es el servicio de directorio compatible con LDAPv3 donde se almacenarán los certificados.
- Un proveedor de servicios criptográficos (CSP). Como hemos visto el CSP permite la comunicación con los dispositivos y debe estar firmado por Microsoft o no funcionará. Nos será proporcionado por el fabricante.

Como hemos dicho el protocolo Kerberos es el mecanismo de autenticación usado en

Windows. En el centro del protocolo tenemos un servidor de confianza llamado Key Distribution Center (KDC). Cuando el usuario se conecta a la red, el KDC verifica la identidad del mismo y proporciona unas credenciales llamadas “tickets”, una por cada servicio de red que quiera usar el usuario. Cada ticket presenta al usuario frente al servicio apropiado y opcionalmente lleva información que indica los privilegios del usuario en el servicio.

La implementación de Microsoft del protocolo utiliza extensiones que permiten el uso de las tarjetas inteligentes, lo que proporciona la doble ventaja de un proceso de autenticación reforzado y un acceso sencillo a la PKI. Normalmente, Kerberos utiliza cifrado de clave simétrica. Sin embargo el certificado de la tarjeta se basa en cifrado asimétrico.

El proceso de *logon* con tarjeta inteligente sigue los siguientes pasos:

1. Cuando el usuario inserta una tarjeta inteligente, el servicio de *logon* de Windows (WINLOGON) envía este evento al Graphical Identification and Authentication (GINA).
2. Al usuario se le pide que introduzca el PIN.
3. El GINA envía el PIN a la Local Security Authority (LSA).
4. La LSA usa el PIN para acceder a la tarjeta y extraer el certificado con la clave pública del usuario.
5. El servicio Kerberos envía el certificado de usuario firmado con la clave privada del usuario al KDC.
6. El KDC compara la identidad del certificado con los objetos de usuario del directorio. El KDC también verifica la firma del certificado para asegurarse de que ha sido expedido por una CA en la que se confía.
7. El KDC cifra la clave de sesión y el TGT con la clave pública del usuario. Este paso asegura que sólo el cliente con la clave privada apropiada puede descifrar la clave de sesión.
8. El cliente descifra la clave de sesión y presenta el TGT. A partir de este momento toda comunicación con Kerberos utiliza cifrado simétrico.

5. Desarrollo de software para Tarjetas Inteligentes[4]

En nuestro caso cuando hablemos de software para tarjetas inteligentes, nos estaremos refiriendo a software anfitrión que se ejecutará en un ordenador y accede a tarjetas inteligentes quizás para incorporarlas a un sistema más complejo. No nos referiremos por tanto a software que se ejecute en la propia tarjeta inteligente, apartado que podría ser merecedor por sí mismo de un trabajo independiente.

Hablaremos, por tanto, de software necesario para que el sistema soporte el uso de una tarjeta específica y sobre todo de aplicaciones de usuario final. Este software normalmente estará escrito en lenguajes de programación de alto nivel y utilizará librerías comerciales o estándar para acceder a diferentes dispositivos de tarjetas, diferentes tarjetas e incluso diferentes tipos de tarjetas. Al contrario que la mayoría del software, que se supone va a ser ejecutado en un entorno de confianza, el software que trata con tarjetas inteligentes asume que el contexto en el que se ejecuta es hostil y no se puede confiar en él. Hasta que haya pruebas que evidencien lo contrario el software anfitrión no confiará en la tarjeta y la tarjeta no confiará en el software que intenta usarla.

Un proveedor de servicios es el mecanismo a través del cual el conjunto de funcionalidades específicas de las tarjetas inteligentes (en forma de API) se hacen accesibles para las aplicaciones. Para cada tarjeta debe haber al menos un proveedor de servicios y es a través de él como una aplicación accede a los datos o servicios específicos de una tarjeta. Los siguientes tres tipos de servicios se encuentran implementados habitualmente en una tarjeta:

- Servicios de ficheros
- Servicios de autenticación
- Servicios criptográficos.

Estos servicios, cuando se hallan presentes tienen mucha funcionalidad en común entre las distintas tarjetas. Consecuentemente es beneficioso estandarizar interfaces a esos servicios de forma que el desarrollo y mantenimiento de aplicaciones sea más sencillo.

El proveedor de servicios tiene, por definición, un conocimiento profundo de la tarjeta a la que proporciona acceso. La implementación de las APIs puede proceder de diversas fuentes, entre las cuales están:

- El fabricante de la tarjeta que desea proporcionar acceso a las mismas desde el entorno PC. Proporcionando un proveedor de servicios permite que el desarrollo de software que acceda a la tarjeta pueda llevarse a cabo por parte de desarrolladores sin demasiada experiencia en la tecnología de tarjetas.
- El distribuidor de la tarjeta que puede querer un proveedor de servicios “personalizado” encima del proporcionado por el fabricante.
- Un suministrador de aplicaciones de tarjeta que quiere definir el nivel de funcionalidad que debe proporcionar la tarjeta para soportar la aplicación. Definiendo la API permite que los distribuidores de tarjetas proporcionen la tarjeta junto con un proveedor de servicios que implemente la API definida por el suministrador de aplicaciones.
- Una o más partes interesadas en un dominio específico que quieren facilitar el desarrollo tanto de aplicaciones como de tarjetas que soporten dichas aplicaciones en el ámbito de su dominio de interés.

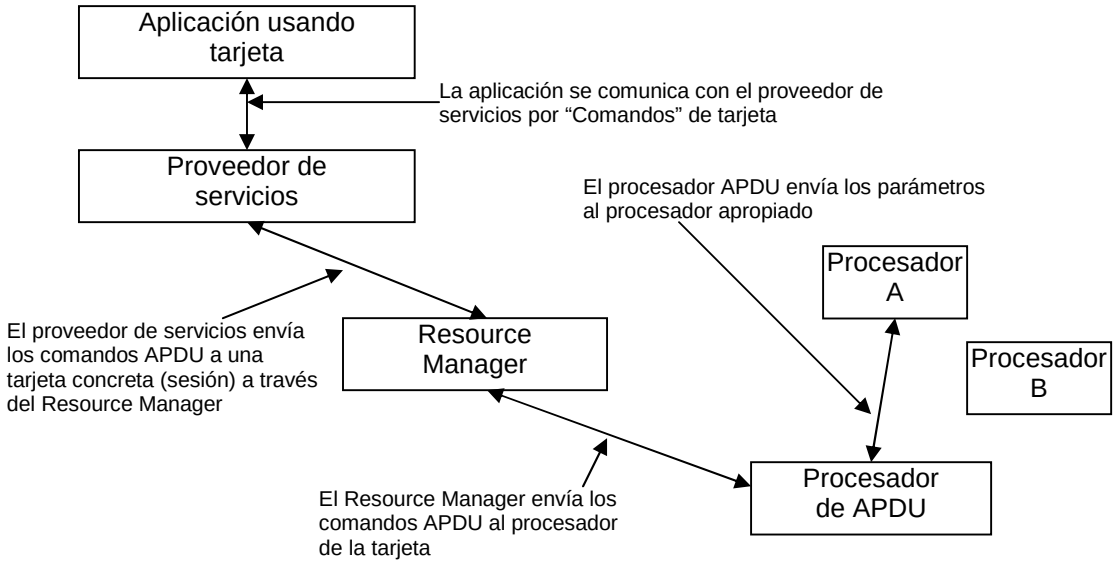
Los proveedores de servicio se construyen a partir de los servicios expuestos por el Resource Manager. Esto proporciona a los desarrolladores de proveedores de servicios un conjunto conocido de interfaces de bajo nivel que pueden usar para simplificar su desarrollo. Además pueden usar los mecanismos de compartición de dispositivos y las primitivas de transacción proporcionadas por el Resource Manager para asegurar una correcta operación de la tarjeta. Como hemos visto, los proveedores de servicio pueden hacer uso a su vez de otros proveedores de servicio para simplificar el desarrollo y la reutilización de código.

Desde el punto de vista del desarrollador de aplicaciones, tal y como vimos en el capítulo 2 al

[4] Referencias: [JUGU98] [RAEF03] [CHAU04] [GAL103] [GAL203]

describir los componentes del sistema, hay tres mecanismos para acceder a los servicios que soportan las tarjetas inteligentes: la CryptoAPI, el componente Scard COM y la Win32 API. El mecanismo elegido depende del tipo de aplicación a desarrollar y de las capacidades de una tarjeta inteligente específica.

En la siguiente figura podemos ver un esquema general de cómo sería la comunicación aplicación-tarjeta en esta arquitectura:



5.1 Win32 API y Scard COM

Las Win32 APIs son las APIs de nivel básico para acceder a las tarjetas inteligentes y requieren de un profundo conocimiento tanto del sistema operativo Windows como de las tarjetas inteligentes para que se puedan usar de modo efectivo. A cambio también proporcionan la mayor flexibilidad para que la aplicación controle los terminales, tarjetas y componentes relacionados. Para los desarrolladores que necesiten tener el máximo control sobre el uso que hace la aplicación de la tarjeta inteligente, esta extensión de las APIs básicas proporciona los interfaces necesarios para controlar la interacción con los dispositivos de tarjeta inteligente.

Normalmente cada proveedor de soluciones desarrolla su propia estructura de ficheros y datos acorde con sus necesidades particulares. Conocer dicha estructura es muy complicado ya que dichos proveedores mantienen esas estructuras en secreto junto con las operaciones necesarias para recuperar la información correctamente y sólo la proporcionan a aquellas personas debidamente autorizadas. Por lo tanto usar las APIs Win32 aparte de necesitar un gran esfuerzo de desarrollo obliga a estar en posesión de información reservada lo que hace prácticamente imposible su utilización para el desarrollo de aplicaciones de amplia difusión. Es por esto que podemos afirmar que dichas APIs son utilizadas de forma casi exclusiva por los fabricantes de tarjetas inteligentes o de soluciones para desarrollar los proveedores de servicios necesarios. Dicho esto centraremos nuestra atención en los demás mecanismos que nos permiten un uso más sencillo de las funcionalidades de las tarjetas, es decir, las APIs estándar para acceder tanto a funcionalidades criptográficas como no criptográficas: Scard COM y CryptoAPI.

El componente Scard COM es una implementación del interface no criptográfico que proporciona Microsoft para permitir un acceso genérico a los servicios basados en tarjetas inteligentes desde aplicaciones escritas en diferentes lenguajes, como C, Visual C++, Java o Visual Basic. Se compone de un conjunto base de objetos de interface COM que los desarrolladores pueden usar para construir interfaces más potentes para usar luego en sus aplicaciones. El desarrollador puede usar herramientas de desarrollo estándar para construir aplicaciones y proveedores de servicio que manejen tarjetas inteligentes.

En general, el desarrollador de aplicaciones no necesita conocer los detalles de cómo funciona una tarjeta inteligente en particular para poder acceder a sus servicios a través de COM. Esta abstracción agiliza el desarrollo de aplicaciones, ahorrando tanto tiempo como costes de desarrollo, y evitando que las aplicaciones queden obsoletas debido a los cambios que se produzcan en el diseño de una tarjeta.

Scard COM expone los servicios no criptográficos de una tarjeta inteligente a una aplicación a través de proveedores de servicios que soportan interfaces específicos. Un interface de tarjeta inteligente consiste en una serie de servicios predefinidos, los protocolos necesarios para invocar dichos servicios, y cualquier otra información relativa al contexto de los servicios.

Como parte de los Smart Card Base Components, Microsoft proporciona varios proveedores de servicios básicos para realizar operaciones genéricas, como búsqueda de tarjetas, manejo de los comandos y respuestas APDU (Application Protocol Data Unit), y acceso al sistema de ficheros de la tarjeta.

A continuación vamos a ver cómo definiríamos el interface para acceder en el entorno .NET a las funciones que nos proporciona WinSCard, usando C#:

```
internal static extern uint SCardEstablishContext(
                uint dwScope,
                IntPtr pvReserved1,
                IntPtr pvReserved2,
                out IntPtr phContext);

internal static extern uint SCardIsValidContext(
                IntPtr hContext);

internal static extern uint SCardReleaseContext(
                IntPtr hContext);

internal static extern uint SCardConnect(
                IntPtr hContext,
                String szReader,
                uint dwShareMode,
                uint dwPreferredProtocols,
                out IntPtr phCard,
                out uint pdwActiveProtocol);

internal static extern uint SCardReconnect(
                IntPtr hCard,
                uint dwShareMode,
                uint dwPreferredProtocols,
                uint dwInitialization,
                out uint pdwActiveProtocol);

internal static extern uint SCardDisconnect(
                IntPtr hCard,
                uint dwDisposition);

internal static extern uint SCardBeginTransaction(
                IntPtr hCard);

internal static extern uint SCardEndTransaction(
                IntPtr hCard,
                uint dwDisposition);

internal static extern uint SCardStatus(
                IntPtr hCard,
                StringBuilder szReaderName,
                ref uint pcchReaderLen,
                out uint pdwState,
                out uint pdwProtocol,
                [Out] byte[] pbAtr,
                ref uint pcbAtrLen);

internal static extern uint SCardGetStatusChange(
                IntPtr hContext,
                uint dwTimeout,
                [In, Out] SCARD_READERSTATE[] rgReaderStates,
```

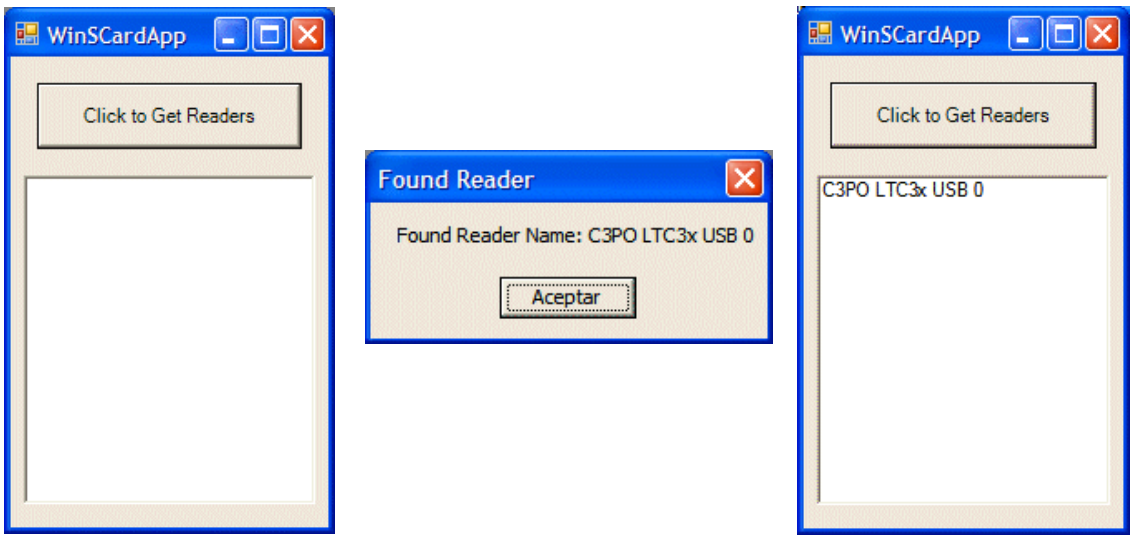
```
                uint cReaders);

internal static extern uint SCardControl(
                IntPtr hCard,
                uint dwControlCode,
                [In] byte[] lpInBuffer,
                uint nInBufferSize,
                [In, Out] byte[] lpOutBuffer,
                uint nOutBufferSize,
                out uint lpBytesReturned);

internal static extern uint SCardTransmit(
                IntPtr hCard,
                SCARD_IO_REQUEST pioSendPci,
                [In] byte[] pbSendBuffer,
                uint cbSendLength,
                SCARD_IO_REQUEST pioRecvPci,
                [In, Out] byte[] pbRecvBuffer,
                ref uint pcbRecvLength);

internal static extern uint SCardGetAttrib(
                IntPtr hCard,
                uint dwAttrId,
                ref IntPtr pbAttr,
                ref uint pcbAttrLen);

internal static extern uint SCardSetAttrib(
                IntPtr hCard,
                uint dwAttrId,
                [In] byte[] pbAttr,
                uint cbAttrLen);

internal static extern uint SCardListReaderGroups(
                IntPtr hContext,
                ref IntPtr pmszGroups,
                ref uint pcchGroups);

internal static extern uint SCardListReaders(
                IntPtr hContext,
                String mszGroups,
                ref IntPtr pmszReaders,
                ref uint pcchReaders);

internal static extern uint SCardFreeMemory(
                IntPtr hContext,
                IntPtr pvMem);

internal static extern uint SCardCancel(
                IntPtr hContext);
```

Veremos ahora cómo escribir una sencilla aplicación que utilizando el interface anterior nos permite buscar los terminales de tarjeta inteligente que tenemos conectados en el sistema y nos muestra el nombre de cada uno de ellos.

```
private int nContext;     //Card reader context handle - DWORD
private System.Windows.Forms.ListBox listBox1;   //Connection handle-DWORD

private void button1_Click(object sender, System.EventArgs e)
{
   //El primer paso para usar tarjetas inteligentes es SCardEstablishContext()
   uint nContext  = 2;       //system
   int  nNotUsed1 = 0;
   int  nNotUsed2 = 0;
   this.nContext  = 0;       //handle to context - SCARDCONTEXT

   int nRetVal1 = SCardEstablishContext(nContext, nNotUsed1, nNotUsed2,
                                        ref this.nContext);
   if (nRetVal1 != 0)
   {
      MessageBox.Show("SCardEstablishContext() ha devuelto un error");
      return;
   }

   //Luego convertiremos los strings delimitados por nulos a string array
   char[] delimiter = new char[1];
```

```
   delimiter[0] = Convert.ToChar(0);

   //Necesitamos llamar a SCardListReaderGroups() para encontrar los grupos a usar
   string cGroupList = ""+Convert.ToChar(0);
   int nGroupCount = -1;      //SCARD_AUTOALLOCATE
   int nRetVal2 = SCardListReaderGroups(this.nContext, ref cGroupList,
                                        ref nGroupCount);
   if (nRetVal2 != 0)
   {
      MessageBox.Show("SCardListReaderGroups() ha devuelto un error");
      return;
   }

   string[] cGroups = cGroupList.Split(delimiter);

   string cReaderList = ""+Convert.ToChar(0);
   int nReaderCount = -1;

   //Obtenemos la lista de terminales
   int nRetVal4 = SCardListReaders(this.nContext, cGroups[0], ref cReaderList,
                                   ref nReaderCount);
   if (nRetVal4 != 0)
   {
      MessageBox.Show("SCardListReaders() ha devuelto un error");
      return;
   }

   string[] cReaders = cReaderList.Split(delimiter);

   listBox1.Items.Clear();
   foreach (string cName in cReaders)
   {
      MessageBox.Show("Found Reader Name: "+cName,"Found Reader");
      listBox1.Items.Add(cName);
   }
}
```

En las siguientes figuras podemos ver el resultado de ejecutar esta aplicación:



5.2 CryptoAPI

Uno de los usos principales de las tarjetas inteligentes, como hemos visto a lo largo de este trabajo, es proporcionar servicios criptográficos para dar soporte a una infraestructura de seguridad. En particular, son el mejor lugar para almacenar claves privadas y realizar operaciones de firma digital que usan dichas claves. Hoy en día podemos decir que existen principalmente dos módulos criptográficos:

- CryptoAPI en las plataformas Windows
- PKCS-11 en plataformas Windows y otras, como Linux.

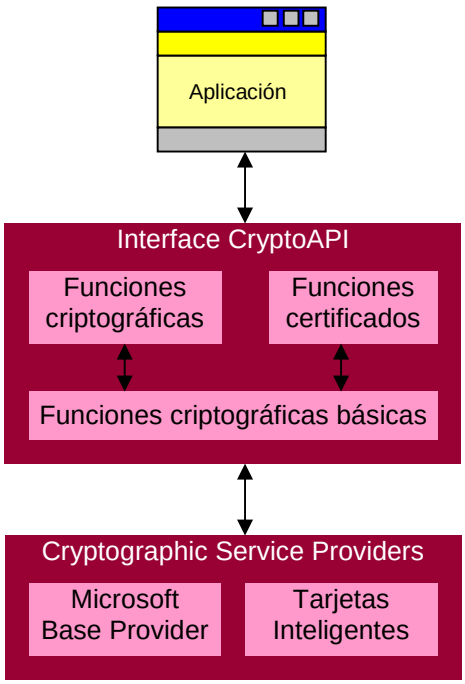
Ambos módulos proporcionan servicios criptográficos con independencia del uso de una tarjeta inteligente. Sin embargo, ambos permiten también el uso de tarjetas inteligentes como elementos de seguridad personalizados. En nuestro caso estudiaremos sólo la CryptoAPI.

CryptoAPI es la API criptográfica de Microsoft. La CryptoAPI está diseñada para abstraer los detalles de las funcionalidades criptográficas, como los algoritmos de encriptación, de forma que las aplicaciones puedan usar criptografía de forma sencilla. Esto se consigue con una arquitectura a dos niveles. En el nivel superior está la CryptoAPI propiamente dicha que ofrece un número de funciones reducido que las aplicaciones pueden utilizar para realizar operaciones criptográficas. Estas funciones permiten el cifrado de bloques de datos así como el cálculo de firmas digitales para dichos datos. Muchas aplicaciones Windows como navegadores de Internet y programas de correo electrónico (como hemos visto en el Capítulo 4) utilizan los servicios de esta API para implementar funciones de seguridad.

Al igual que con otras APIs estandarizadas lo que se busca es aislar las aplicaciones de los cambios que puedan surgir en la implementación de las mismas. En el caso de servicios criptográficos existen varias formas en que estos servicios pueden cambiar. En primer lugar en muchos países existen restricciones a la exportación de métodos criptográficos. En algunos casos no se puede utilizar ningún tipo de criptografía y en otros la calidad de la misma se ve mermada. En casos como este, es útil poder soportar lo que llamamos “criptografía fuerte” en algunos casos, “criptografía débil” en otros o ninguna pero sin que la aplicación tenga que ser consciente de las diferencias.

La segunda capa de esta arquitectura está formada por módulos reemplazables llamados Cryptographic Service Providers (CSPs). Los CSPs pueden estar compuestos sólo por software o pueden ser parte de una solución basada en hardware en donde la funcionalidad criptográfica reside en una tarjeta inteligente, o cualquier otro dispositivo conectado al ordenador.

En la siguiente figura podemos ver una perspectiva completa del modelo de programación que nos ofrece la CryptoAPI:



La CryptoAPI puede utilizar cualquier CSP para realizar sus operaciones criptográficas evitando de esta manera las diferencias entre los CSP desde el punto de vista de la aplicación. Esto nos permite utilizar una gran variedad de tarjetas para las operaciones criptográficas dado que, por supuesto, nos interesa poder utilizar tarjetas de distintos fabricantes. Un CSP está asociado con un tipo específico de tarjetas inteligentes, implementando la relación entre las funciones criptográficas expuestas por la CryptoAPI y los comandos de bajo nivel accesibles a

través de la Win32 API para tarjetas inteligentes. Por tanto, el CSP gobierna las operaciones criptográficas específicas de la tarjeta.

El interface CryptoAPI se compone de tres tipos de funcionalidades: las funciones de certificados, las funciones criptográficas y los servicios de bajo nivel. Entender las funciones criptográficas es el primer paso a dar para familiarizarse con el API. Entre éstas están las funciones para crear y usar claves y para cifrar y descifrar información. Las funciones de certificados permiten extraer, almacenar y verificar certificados adjuntos a documentos y enumerar los que están almacenados en la máquina. A un nivel más bajo hay otras funciones cuyos argumentos requieren un CSP específico. A menos que sea una necesidad crítica para la aplicación, no se deben acceder los CSP directamente y debe evitarse utilizar las particularidades de un proveedor determinado. Hay aplicaciones que necesitan su propio CSP por varias razones, entre las cuales está la necesidad de utilizar mecanismos de seguridad especiales.

Vamos a ver un ejemplo de utilización de la CryptoAPI. Igual que en el ejemplo que vimos para la WinSCard, lo primero que tendremos que hacer será importar aquellas funciones de la librería que vamos a necesitar (para no ser repetitivos obviaremos este paso). A continuación vamos a describir un código que realiza las siguientes funciones:

- Lee un certificado X.509 de los almacenes de certificados y filtra por el SubjectName.
- Extrae el módulo y el exponente de la clave RSA pública del certificado.
- Genera una clave simétrica 3DES secreta y aleatoria para el cifrado junto con el Vector de Inicialización.
- Cifra un fichero con la clave 3DES y lo guarda.
- Cifra la clave 3DES y el VI usando la clave pública RSA del certificado y los guarda en un fichero.

El programa pregunta al usuario por una cadena para el SubjectName del certificado. Normalmente los certificados de otras personas se guardan en el almacén “ADDRESSBOOK” (también llamado “Otras Personas”), aunque también buscamos en el almacén “MY”. Para cifrar algo para nosotros mismos debemos tener un certificado X.509 válido en el almacén “MY” (también llamado “Personal”).

```
private X509Certificate GetRecipientStoreCert(String searchstr) {
   X509Certificate cert = null;
   IntPtr hSysStore = IntPtr.Zero;
   IntPtr hCertCntxt = IntPtr.Zero;
   string[] searchstores = {"ADDRESSBOOK", "MY"};

   uint openflags =   CERT_SYSTEM_STORE_CURRENT_USER |
                      CERT_STORE_READONLY_FLAG       |
                      CERT_STORE_OPEN_EXISTING_FLAG;

   foreach(String store in searchstores)
   {
      hSysStore = Win32.CertOpenStore("System", ENCODING_TYPE, IntPtr.Zero, openflags,
                                       store);
      if(hSysStore == IntPtr.Zero){
         Console.WriteLine("No se ha podido abrir el almacén {0}", store);
         continue;
      }
      hCertCntxt=Win32.CertFindCertificateInStore(
          hSysStore,
          ENCODING_TYPE,
          0,
          CERT_FIND_SUBJECT_STR,
          searchstr ,
          IntPtr.Zero) ;

      if(hCertCntxt != IntPtr.Zero){
         cert = new X509Certificate(hCertCntxt);
         Console.WriteLine("\nEncontrado certificado en almacén {0} con SubjectName
                           \"{1}\"", store, searchstr);
         Console.WriteLine("SubjectName:\t{0}", cert.GetName());
         break;
      }
   } // end foreach
```

El primer certificado que se encuentra se almacena en la variable **X509Certificate cert**. Si se encuentra un certificado válido se llama a **GetCertPublicKey()**. Para extraer la clave pública, el módulo y el exponente, se debe decodificar la clave pública codificada en ASN.1 a una estructura PUBLICKEYBLOB de CryptoAPI llamando a **CryptDecodeObject()** con el parámetro RSA_CSP_PUBLICKEYBLOB. Después se parsea esa estructura para obtener los parámetros de la clave pública. El módulo, el exponente y el tamaño de la clave se asignan a variables. Las estructuras de CryptoAPI normalmente representan los datos como *little-endian*, pero los parámetros RSAParameters de .NET son *big-endian*, por eso tenemos que darles la vuelta.

```
private bool GetCertPublicKey(X509Certificate cert)
{
   byte[] publickeyblob ;
   byte[] encodedpubkey = cert.GetPublicKey(); //ASN.1 encoded public key
   uint blobbytes=0;

   if(Win32.CryptDecodeObject(ENCODING_TYPE, RSA_CSP_PUBLICKEYBLOB, encodedpubkey,
     (uint)encodedpubkey.Length, 0, null, ref blobbytes))
   {
      publickeyblob = new byte[blobbytes];
      Win32.CryptDecodeObject(ENCODING_TYPE, RSA_CSP_PUBLICKEYBLOB, encodedpubkey,
        (uint)encodedpubkey.Length, 0, publickeyblob, ref blobbytes)
   }
   else{
      Console.WriteLine("Imposible decodificar la clave publica del certificado");
      return false;
   }

   PUBKEYBLOBHEADERS pkheaders = new PUBKEYBLOBHEADERS();
   int headerslength = Marshal.SizeOf(pkheaders);
   IntPtr buffer = Marshal.AllocHGlobal(headerslength);
   Marshal.Copy( publickeyblob, 0, buffer, headerslength );
   pkheaders = (PUBKEYBLOBHEADERS) Marshal.PtrToStructure(buffer,
               typeof(PUBKEYBLOBHEADERS) );
   Marshal.FreeHGlobal(buffer);

   //-----  Averiguar tamaño de clave publica en bits -------------
   this.certkeysize = pkheaders.bitlen;

   //-----  Averiguar el exponente -------------
   byte[] exponent = BitConverter.GetBytes(pkheaders.pubexp); //little-endian
   Array.Reverse(exponent);    //convertir a big-endian
   this.certkeyexponent = exponent;

   //-----  Averiguar el módulo  -------------
   int modulusbytes = (int)pkheaders.bitlen/8 ;
   byte[] modulus = new byte[modulusbytes];
   try{
      Array.Copy(publickeyblob, headerslength, modulus, 0, modulusbytes);
      Array.Reverse(modulus);
      this.certkeymodulus = modulus;
   }
   catch(Exception){
      Console.WriteLine("Problema sacando el módulo de la clave");
      return false;
   }
   return true;
}
```

Una vez obtenidas las propiedades de la clave se llama a la función **TripleDESEncrypt()** para generar una clave aleatoria y un VI para luego cifrar el contenido del fichero y guardar la salida en otro fichero.

```
private bool TripleDESEncrypt(String content, String encContent, String encKeyfile,
String encIVfile)
{
   FileStream fin = new FileStream(content, FileMode.Open, FileAccess.Read);
   FileStream fout = new FileStream(encContent, FileMode.OpenOrCreate,
                     FileAccess.Write);
   byte[] buff = new byte[1000]; //buffer de cifrado
   int lenread;
   byte[] encdata;
```

```
   try{
      TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
      CryptoStream encStream = new CryptoStream(fout, tdes.CreateEncryptor(),
                               CryptoStreamMode.Write);
      Console.WriteLine("\nCifrando contenido ... ");

      //hacer el cifrado ...
      while( (lenread = fin.Read(buff, 0, 1000))>0)
         encStream.Write(buff, 0, lenread);
      encStream.Close();
```

Finalmente, se inicializa una instancia de RSAParameters con el exponente y el módulo de la clave. Se crea una instancia de RSACryptoServiceProvider y se inicializa con el objeto RSAParameters, luego se cifran la clave 3DES y el VI en ficheros de salida.

```
private byte[] DoRSAEncrypt(byte[] keydata, byte[] modulus, byte[] exponent)
{
   byte[] protectedkey = null;
   try{
      //Inicializar RSAKeyInfo con los parametros
      RSAParameters RSAKeyInfo = new RSAParameters();
      RSAKeyInfo.Modulus = modulus;
      RSAKeyInfo.Exponent = exponent;

      //Inicializar RSACryptoServiceProvider
      RSACryptoServiceProvider oRSA = new RSACryptoServiceProvider();
      oRSA.ImportParameters(RSAKeyInfo);
      protectedkey = oRSA.Encrypt(keydata, false);
   }
   catch(CryptographicException){
      return null ;
   }
   return protectedkey;
}
```

5.3 CAPICOM

Poder llevar a cabo con éxito incluso las tareas criptográficas más sencillas usando la CryptoAPI requiere tener un profundo conocimiento tanto de las APIs de Windows como del lenguaje utilizado. Se necesita una considerable curva de aprendizaje para conseguir manejar con soltura los conceptos clave de la criptografía, como las codificaciones (ASN.1 – Abstract Syntax Notation) y las estructuras de datos que se deben pasar como parámetros a muchas funciones de CryptoAPI. La CryptoAPI se introdujo en 1996, pero poco a poco se ha ido extendiendo, de forma inevitable, la necesidad de usar criptografía, por lo tanto es importante que el acceso a la programación criptográfica esté disponible en un amplio abanico de posibilidades (Visual Basic, Windows Script Host, lenguajes .NET, etc.).

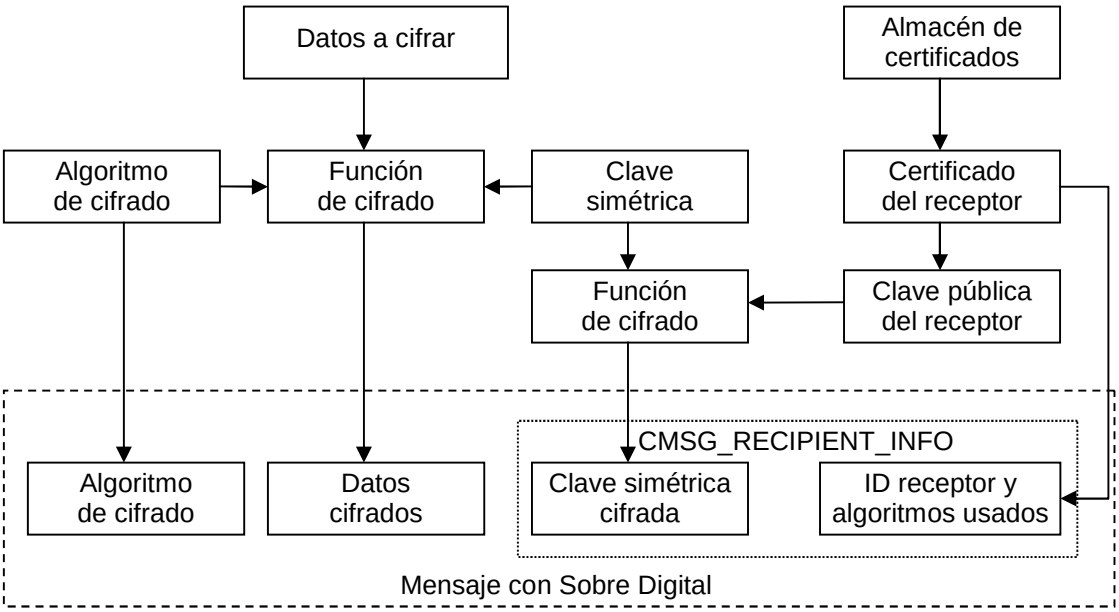
En 2001 se introdujo CAPICOM y fue diseñada para permitir un acceso sencillo a la funcionalidad proporcionada por CryptoAPI, en particular a las tareas relacionadas con certificados. Los programadores pueden así desarrollar aplicaciones y utilidades sin complicarse con la CryptoAPI y llevando a cabo las principales tareas criptográficas con mucho menos código que el que se requiere para realizar la misma tarea directamente con CryptoAPI.

Por ejemplo, el siguiente código genera y muestra el sobre digital de un mensaje cifrado en base64:

```
Set oEnvData = CreateObject("CAPICOM.EnvelopedData")
oEnvData.Content = "Tu sueldo se incrementa en: 2000 Loonies"
Message = oEnvData.Encrypt
WScript.Echo "Tu mensaje ensobrado" & vbCrLF & Message
```

Se crea un objeto **CAPICOM.EnvelopedData**, su propiedad **Content** se inicializa con una simple cadena de texto y luego se llama al método **Encrypt()** del objeto **EnvelopedData** usando parámetros por defecto. Se genera automáticamente una clave simétrica secreta de forma aleatoria con un algoritmo de cifrado por defecto y un tamaño de clave por defecto. La clave secreta se usa para cifrar el contenido. Al usuario se le presenta un diálogo con el contenido del almacén de certificados **AddressBook** (“Otras personas” en el navegador) para

que seleccione un receptor del mensaje. Se extrae automáticamente la clave pública RSA del receptor a partir del certificado seleccionado y se usa para cifrar la clave simétrica secreta. Podemos ver el diagrama completo en la siguiente figura:



Este sencillo ejemplo nos muestra cómo CAPICOM encapsula la mayor parte de la complejidad de CryptoAPI. Sin embargo, CAPICOM posee funcionalidad suficiente, a través de sus propiedades y métodos para ajustar parámetros criptográficos importantes. Por ejemplo, en el código anterior, sería deseable cambiar el algoritmo usado para generar la clave simétrica junto con la longitud de la clave usada. La propiedad **EnvelopedData.Algorithm** nos permite especificar el nombre del algoritmo que deseamos y la longitud de la clave a emplear. Además se pueden especificar con la propiedad **EnvelopedData.Recipients,** que representa una colección de certificados de destinatarios, los destinatarios del mensaje, en vez de usar el diálogo del almacén por defecto del sistema.

Con la introducción en 2002 de CAPICOM v2, se añadieron un conjunto de funcionalidades que incluyen funciones de búsqueda en almacenes de certificados potentes y sencillas, soporte para sobres digitales CMS/PKCS#7 y soporte para firmas Authenticode. La tecnología Authenticode permite autenticación de código durante una descarga, control sobre qué código se puede ejecutar, control sobre quién puede extender las aplicaciones, etc.

A fin de comparar la sencillez de manejo de CAPICOM con respecto al manejo de la CryptoAPI directamente, vamos a ver un ejemplo similar al visto en el apartado anterior. Se lee el contenido de un fichero y se abre un almacén de certificados que contiene los certificados de los destinatarios del mensaje. Se añaden todos los certificados en el almacén como destinatarios, se crea el sobre digital y dicho sobre se escribe a un fichero.

```
Sub Envelope(ByVal InFile As String, ByVal OutFile As String, ByVal StoreName As String)

   On Error GoTo ErrorHandler

   'Abrir el fichero de entrada y leer el mensaje a ensobrar.
   Dim Text As String
   Open InFile For Input As #1
   Input #1, Text
   Close #1
   If Len(Text) < 1 Then
      MsgBox "No hay ningún mensaje."
      Exit Sub
   End If

   'Abrir el almacén que contiene los certificados de los destinatarios.
   Dim CertStore As New Store
   CertStore.Open CAPICOM_CURRENT_USER_STORE, StoreName, CAPICOM_STORE_OPEN_READ_ONLY
```

```
   'Comprobar que el almacén no esté vacío.
   If CertStore.Certificates.Count < 1 Then
      MsgBox "No hay ningún certificado disponible."
      Set CertStore = Nothing
      Exit Sub
   End If

   'Declarar e inicializar un objeto EnvelopedData
   Dim EnvMessage As New EnvelopedData
   EnvMessage.Content = Text
   Dim I As Integer
   For I = 1 To CertStore.Certificates.Count
      EnvMessage.Recipients.Add CertStore.Certificates.Item(I)
   Next I

   'Escoger un algoritmo de cifrado y la longitud de la clave
   Envmessage.Algorithm.Name = ENCRYPTION_ALGORITHM_RC4
   Envmessage.Algorithm.KeyLength = KEY_LENGTH_128_BITS

   'Declaramos la variable que contendrá el mensaje ensobrado.
   Dim EnvelopedMessage As String

   'Cifrar el mensaje a EnvelopedMessage. El mensaje ensobrado contiene todo
   'lo que el receptor necesitará para descifrar el mensaje, incluyendo el
   'algoritmo usado y la longitud de la clave.
   EnvelopedMessage = EnvMessage.Encrypt

   'Comprobar la longitud para asegurarnos que el cifrado ha funcionado.
   If Len(EnvelopedMessage) < 1 Then
      MsgBox "No se ha cifrado el mensaje."
   Else
      MsgBox "El mensaje tiene " & Len(EnvelopedMessage) & " caracteres"
      'Abrir el fichero de salida y encribir el mensaje cifrado.
      Open OutFile For Output As #2
      Write #2, EnvelopedMessage
      Close #2
      MsgBox "Se ha grabado el mensaje en el fichero"
   End If

   'Liberamos los objetos usados.
   Set Envmessage = Nothing
   Set CertStore = Nothing
   Exit Sub

ErrorHandler:
   If Err.Number > 0 Then
      MsgBox "Error de Visual Basic : " & Err.Description
   Else
      MsgBox "Error de CAPICOM : " & Err.Number
   End If
End Sub
```

6. Ejemplos de aplicaciones

En este apartado expondré brevemente algunas aplicaciones que he desarrollado a modo de ejemplo de lo que se podría hacer con la utilización de certificados digitales y tarjetas inteligentes.

El fin principal del desarrollo de estas aplicaciones era conocer de primera mano los problemas a los que se enfrenta un programador a la hora de tener que dar soluciones basadas en estas tecnologías.

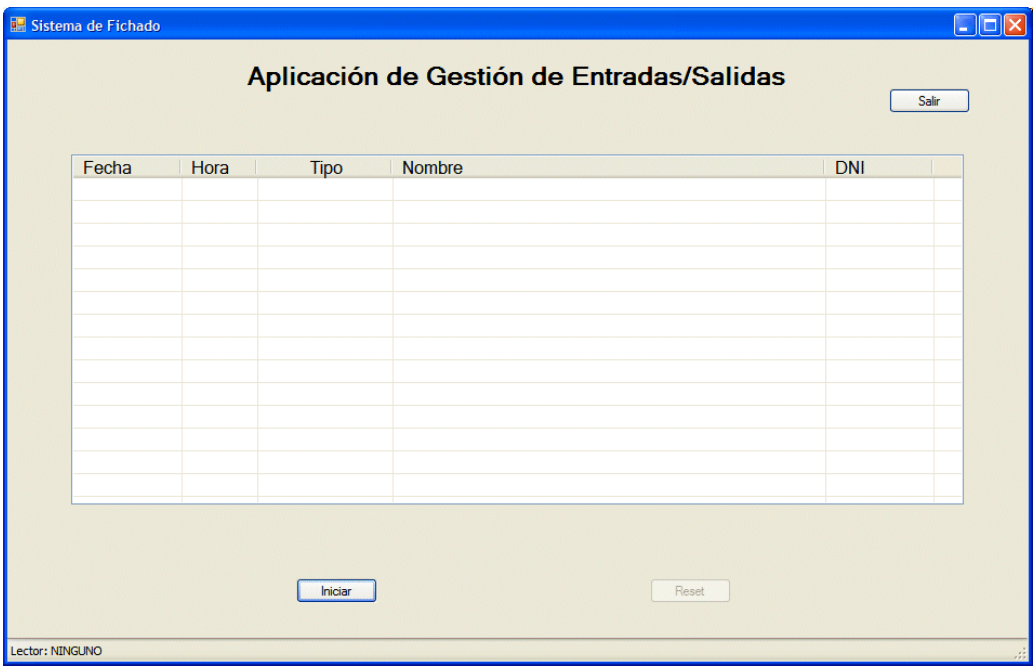
Se han escogido estas aplicaciones porque he considerado que son bastante representativas de problemas reales a los que se podría dar una solución satisfactoria mediante el empleo de tarjetas inteligentes pero, evidentemente, tan sólo son la punta de lanza de aquello que se podría lograr con el uso de las mismas.

Es necesario aclarar que dado que estas aplicaciones se han desarrollado con fines demostrativos se pretendía demostrar el resultado de su uso por varios individuos. Para ello era necesario estar en posesión de varios certificados digitales. Por ello se han creado certificados digitales ficticios, en el sentido de que no tienen ninguna validez legal al no estar expedidos por una autoridad certificadora válida, pero correctamente construídos según el estandar X.509v3, es decir, salvo su carencia de validez son exactos a los legalmente conseguidos. Por ese motivo, al insertar la tarjeta inteligente se le pide al usuario que seleccione el certificado que desea usar, caso que en la vida real no ocurriría ya que el usuario sólo tendría su certificado almacenado en la tarjeta.

6.1 Sistema de fichado

En esta primera aplicación he implementado el clásico control de entradas/salidas de empleados a sus puestos de trabajo. Salvo que en esta ocasión en vez de utilizar las clásicas cartulinas de fichado donde se estampan las horas de entrada y salida, se utilizan tarjetas inteligentes y los movimientos quedan almacenados en una base de datos.

El interfaz de la aplicación está compuesto por un *listbox* en el que se van mostrando los diversos movimientos que se producen y que podemos ver en la siguiente figura:

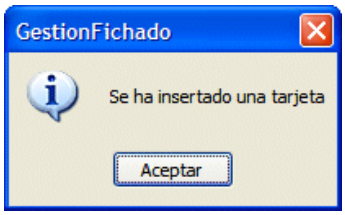


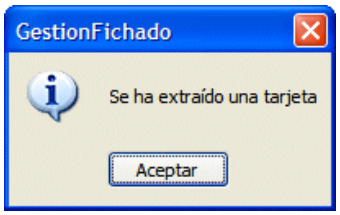
Cuando pulsamos el botón “Iniciar”, la aplicación busca los terminales conectados al sistema, escoge el primero que encuentra y muestra su nombre en la barra de estado. A partir de ese momento se queda esperando la inserción de alguna tarjeta en dicho terminal.

La detección de un cambio de estado en el terminal es esencial para el funcionamiento de cualquier aplicación. Lo ideal sería que el terminal pudiese avisar a nuestra aplicación mediante un evento cada vez que se produce un cambio de estado. Desgraciadamente no existe tal posibilidad, así que tendrá que ser nuestra aplicación la que, realizando consultas al terminal, determine si se ha producido un cambio de estado o no.

El componente Scard COM nos proporciona acceso a las funciones no criptográficas del terminal. Entre ellas están algunas que nos permiten detectar el estado del terminal a través de diversos parámetros. Sin embargo, recordemos que esta es una API común estándar y que puede que no todas las funciones estén implementadas por el software del fabricante. En el terminal usado en nuestro caso así ocurre.

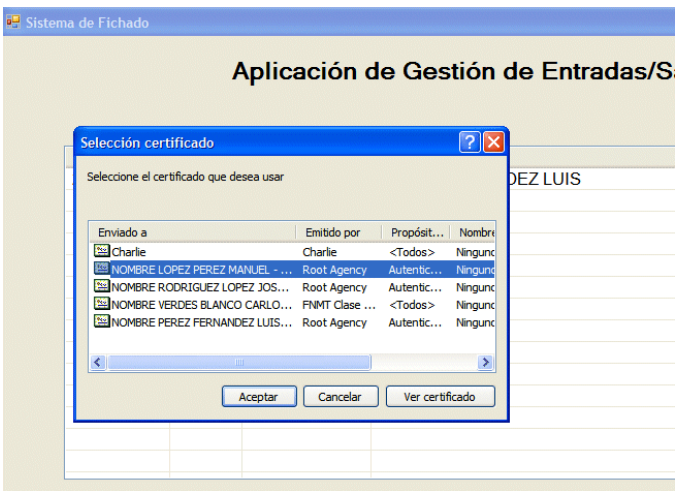
Así que nos decantamos por un método que nos resulte infalible independientemente del terminal usado. Consiste en tratar de conectarse a la tarjeta como si fuesemos a realizar alguna operación con ella. En caso de que exista una tarjeta insertada en el terminal se nos devuelve un identificador de dicha conexión y si no, se nos devuelve un error. Con esto conseguimos desarrollar un módulo que nos avise de un cambio de estado:





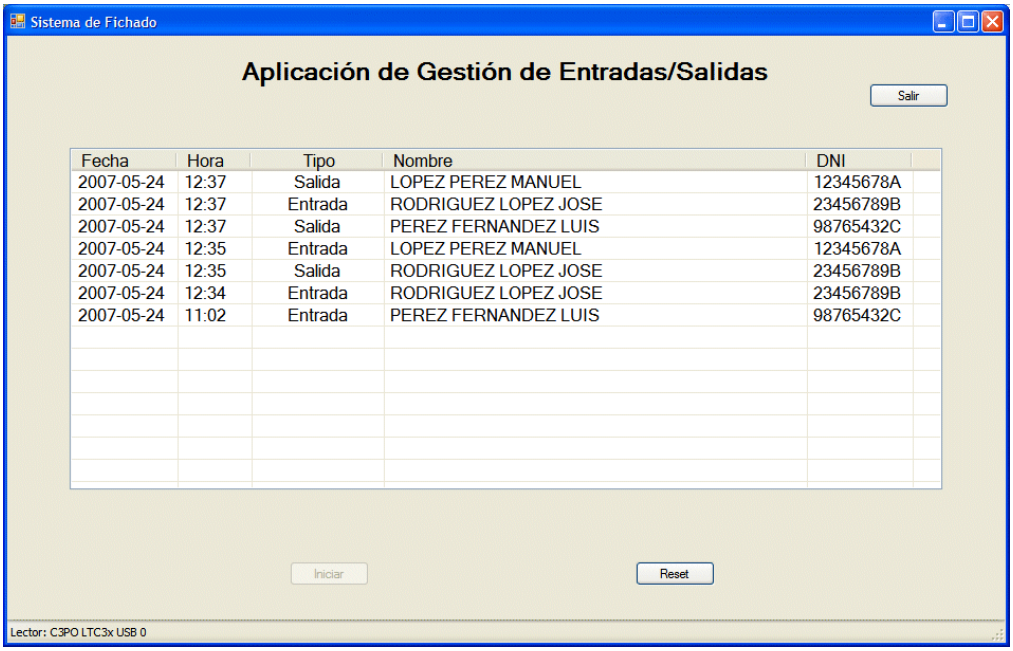
El problema reside en que dicha comprobación de estado debe hacerse en un bucle y esto nos impide atender a los eventos que se produzcan en el formulario principal. Por lo tanto la comprobación tenemos que implementarla como un *thread* aparte que sólo se dedique a eso y nos avise cuando hay un cambio para que el formulario principal pueda actuar en consecuencia.

Una vez que se inserta una tarjeta se le pide al usuario que escoja uno de los certificados que hemos creado para la demostración:



A continuación se extraen del certificado los datos que nos interesan y se consulta el estado del empleado en la base de datos. Esto nos dará el tipo de movimiento que debemos registrar: entrada o salida que, junto con los demás datos almacenamos en el registro.

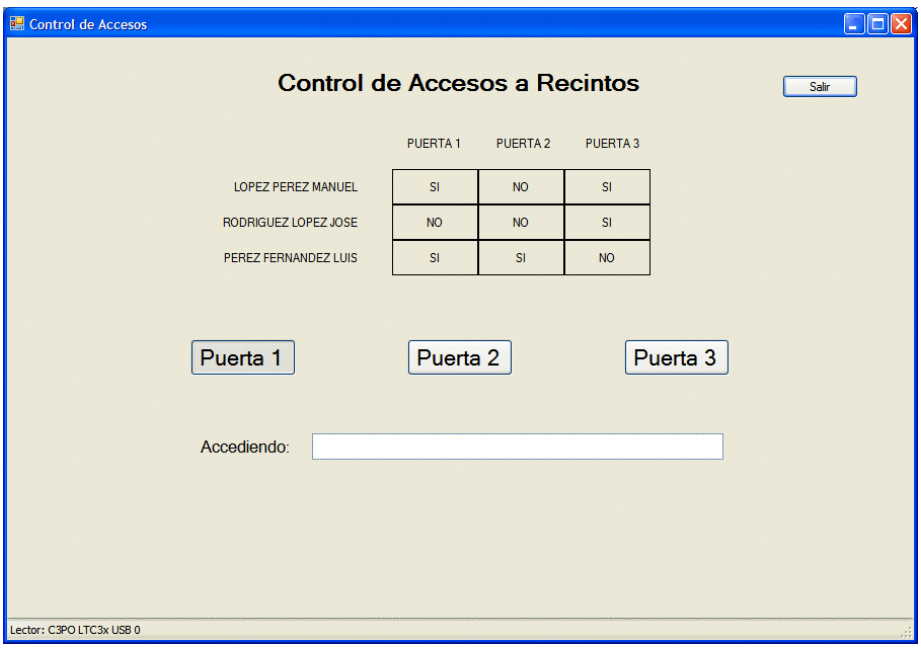
En la siguiente figura vemos un ejemplo de utilización de la aplicación:



6.2 Sistema de acceso a recintos

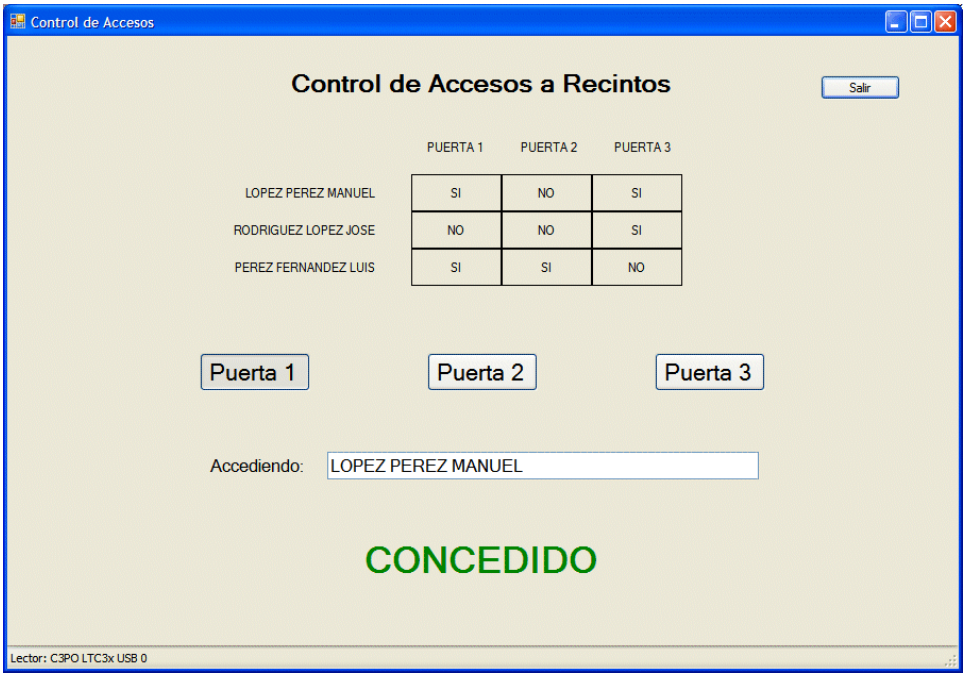
En esta aplicación vamos a simular un sistema de autorización/denegación de acceso a diversos recintos. Para ello tendremos una base de datos donde indicaremos a qué recintos tiene acceso cada usuario.

En nuestro caso nos vamos a limitar a tres posibles usuarios y a tres supuestas puertas de acceso a recintos que vamos a simular con la pulsación del botón correspondiente. En la siguiente figura vemos el entorno de simulación:

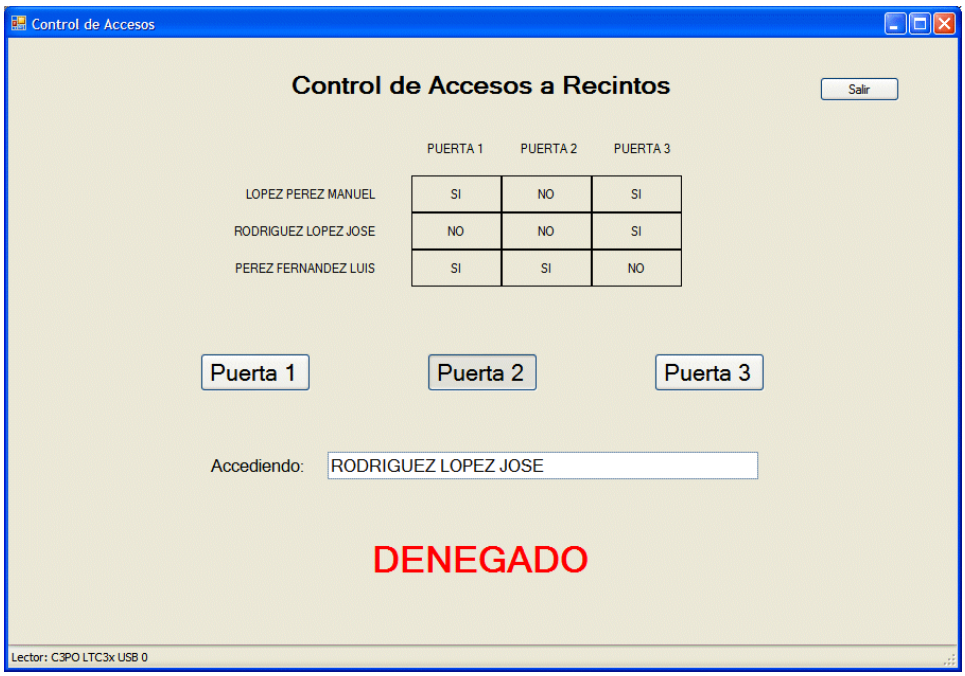


Para operar pulsaremos el botón de la puerta cuyo acceso queremos simular y a continuación insertamos una tarjeta, elegimos el certificado de la persona que intenta el acceso y en pantalla nos mostrará el resultado de dicho intento.

En la siguiente figura vemos cómo se concede acceso a un usuario autorizado:



Y en esta otra vemos lo que ocurre si se intenta el acceso a un recinto sin autorización:



Si pensamos a un nivel global, es decir, si nuestro objetivo es automatizar todo lo posible las tareas habituales a fin de que la vida sea más cómoda para los usuarios, veremos estas dos primeras aplicaciones como el comienzo de una cadena que puede simplificar el trabajo y mejorar la imagen de nuestra empresa.

Por ejemplo, si cuando un empleado ficha su entrada/salida guardamos ese estado en una

base de datos podemos hacer que dicha información sea consultable por los demás miembros de la empresa de una forma sencilla. De esta forma podríamos evitarnos muchas llamadas inútiles, saber si se puede convocar una reunión con todos los asistentes necesarios, o saber si a una persona podemos enviarle una tarea urgente o en caso de que no se encuentre desviarla a otra persona. Esto podría ser de ayuda también si tenemos implantado un sistema de gestión de expedientes a la hora de saber a quién debe asignar las tareas pendientes.

También podríamos enlazarlo con una centralita digital programable de forma que si alguien llama a un empleado y éste no se encuentra presente se le informe de este hecho mediante una locución grabada o, si así se ha decidido, se desvíe la llamada hacia otro interlocutor.

Pensemos en el caso de un empleado que puede tener varias extensiones telefónicas porque su trabajo le obliga a desplazarse, incluso entre distintos edificios. Con el sistema de acceso a recintos podemos tener registrado el último acceso que ha hecho el empleado y en función de ello desviar todas las llamadas entrantes en sus extensiones a la que está en dicho recinto, con lo que se ahorraría mucho tiempo intentando localizar a esa persona.

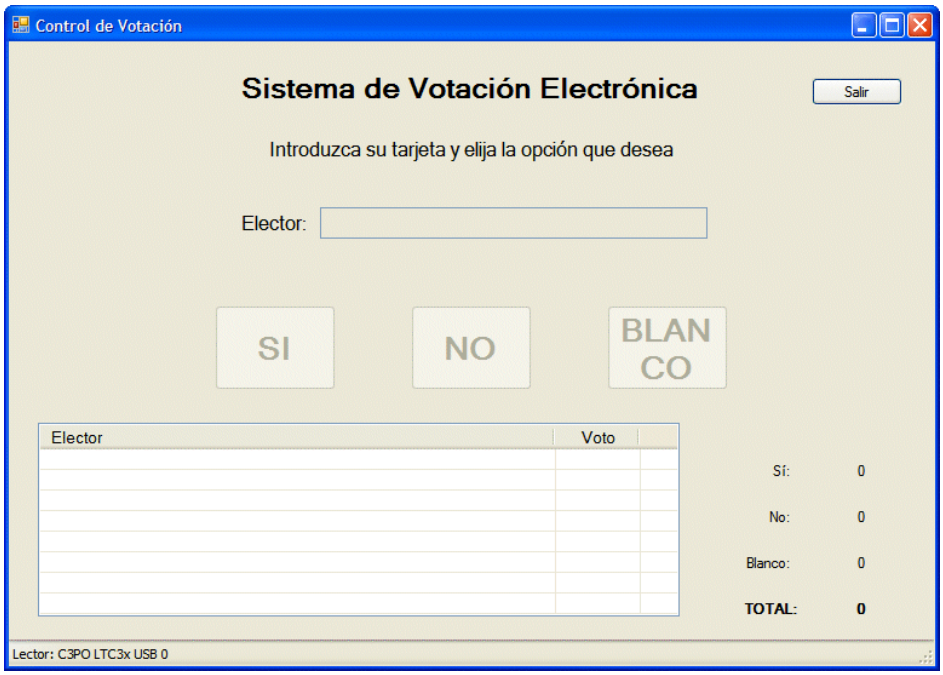
También, si la seguridad así lo requiere, podemos registrar los intentos de acceso por parte del personal a zonas restringidas a fin de investigar el motivo de dichos intentos, etc. El límite lo pone nuestra imaginación y, como siempre, los medios de los que dispongamos.

6.3 Sistema de votación

En este ejemplo se ha implementado un sencillo sistema para votar de forma electrónica y que ahorraría mucho tiempo, ya que en un sistema tradicional debe comprobarse uno a uno y a mano que el votante está autorizado a ejercer el voto.

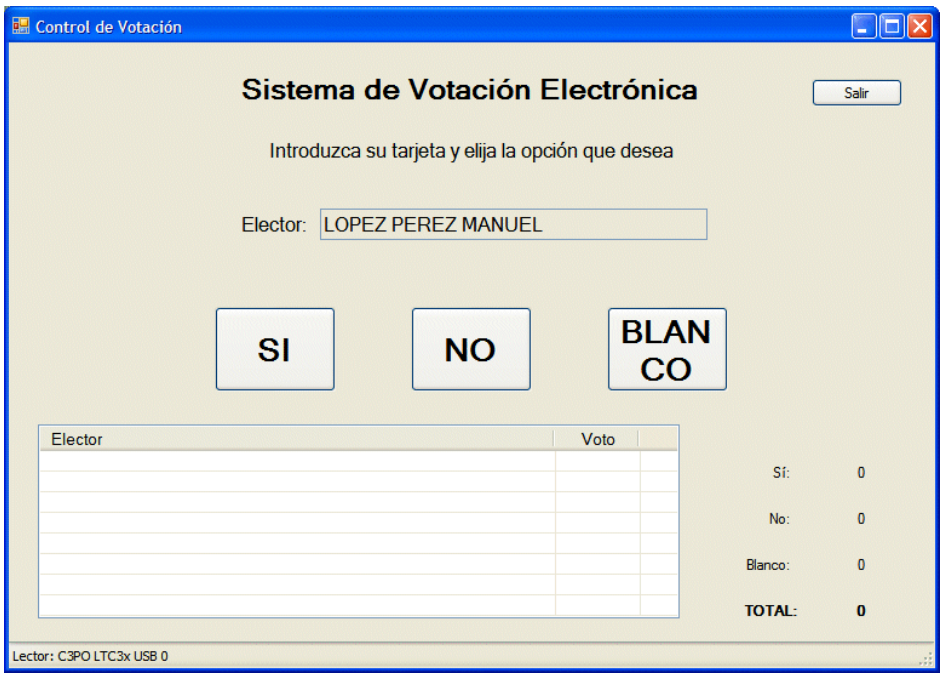
En este caso, simplemente prepararíamos una base de datos con los votantes autorizados y ellos mismos ejercerían el voto siendo el sistema el encargado de autorizarles a emitir su voto o no, en función de que su certificado esté autorizado en la base de datos.

En la siguiente figura vemos cómo sería el sistema:

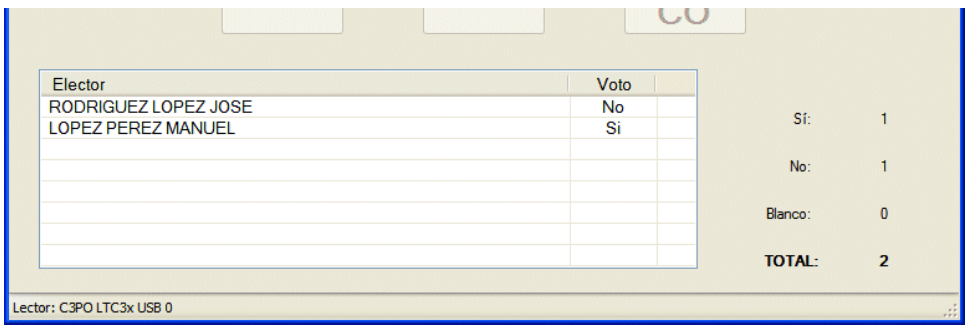


Para operar con el sistema, el elector introduciría su tarjeta de la que extraeríamos los datos de su certificado y comprobaríamos que éste es válido para ejercer el voto, identificando inequívocamente al votante frente al sistema:

En caso de que esté autorizado a votar, se habilitan los botones a tal efecto pidiéndole al usuario confirmación de su voto y permitiéndole cambiar el mismo hasta el momento en que esté seguro:



Tras ejercer su voto, éste quedará reflejado en un registro y el recuento de votos se hace de forma automática e instantánea. Evidentemente en caso de que el voto sea secreto, en dicho registro sólo se almacenará el hecho de haber ejercido el voto, y no el voto en sí.



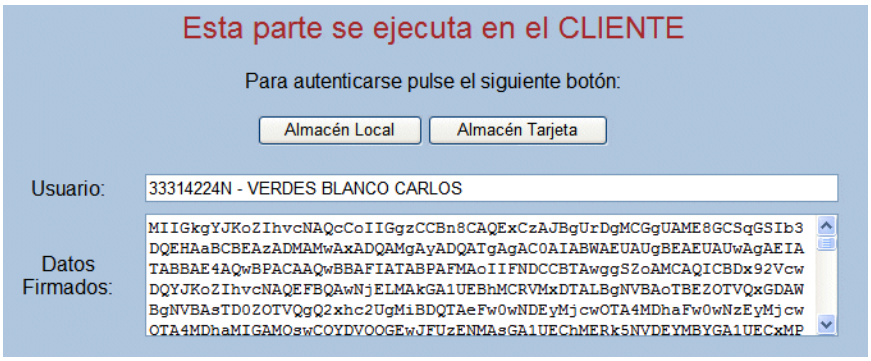
6.4 Autenticación frente a una página web

En este último ejemplo de utilización, vamos a utilizar nuestro certificado digital para autenticarnos frente a un servidor web.

En este caso cuando queremos autenticarnos pulsamos el botón correspondiente al almacén donde se encuentre nuestro certificado. Hay que decir que, dado que comprobaremos la autenticidad del certificado, éste tiene que ser uno válido en todos los sentidos, es decir, los generados por nosotros a efectos de prueba no servirán en este caso ya que no han sido emitidos por una entidad de confianza.

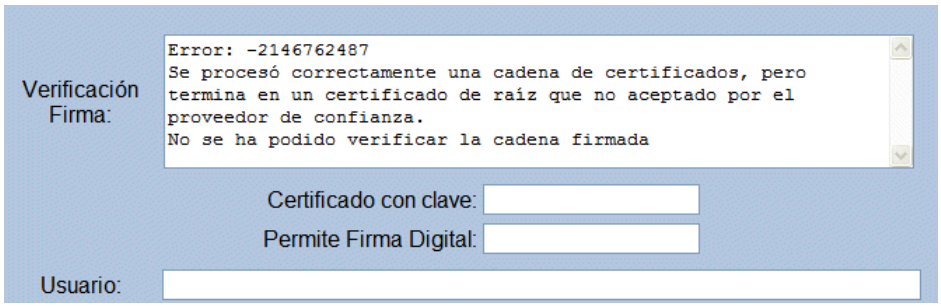
Una vez elegido el certificado para autenticarnos vamos a firmar con su clave privada unos datos a nuestra elección, en nuestro caso hemos elegido el NIF y su nombre completo. En la siguiente figura vemos cómo se nos solicita el PIN para poder realizar la firma:

Una vez hecho esto, se firman los datos elegidos con la clave privada del usuario y todo esto, datos firmados junto con el certificado del usuario se envían al servidor a fin de que pueda comprobar su autenticidad.



Para simular la recepción de estos datos por el servidor pulsamos el botón “Validar” y en ese momento se hace la validación del certificado recibido, la validación de la firma incluída y la extracción de los datos que se han firmado. El resultado de todas estas operaciones lo reflejamos en la siguiente figura:

Si todo ha sido correcto, en el servidor tenemos el certificado validado del usuario del cual podemos extraer la información que necesitemos para comprobar si tiene autorización para conectarse al servidor o no. En caso de que se produzca algún error durante el proceso también lo veremos refrejado de la siguiente manera:



7. Conclusiones

Como declara el Art. 6 de la Declaración Universal de Derechos Humanos, "*Todo ser humano tiene derecho, en todas partes, al reconocimiento de su personalidad jurídica*". Por tanto la identidad personal es un derecho de toda persona y los Estados tienen la obligación de establecer los mecanismos adecuados para facilitársela.

Según la definición de la Real Academia Española de la Lengua, "*La identidad es el conjunto de rasgos propios de un individuo o de una colectividad que los caracterizan frente a los demás*", y también "*Conciencia que una persona tiene de ser ella misma y distinta a las demás*". Como se puede apreciar la identidad personal es un concepto importante que toma aún más valor en la actual Sociedad de la Información. De esta forma se entiende la necesidad de establecer los medios y mecanismos más adecuados para otorgar esta identidad personal a cada individuo.

Otorgar identidad personal adquiere una nueva dimensión cuando se trata de establecerla para un uso no presencial en medios telemáticos. Y aunque la identidad siempre es física, es necesario establecer mecanismos y procedimientos electrónicos para verificarla en estos nuevos ámbitos.

El desarrollo de la Sociedad de la Información y la difusión de los efectos positivos que de ella se derivan exigen la generalización de la confianza de las personas en las comunicaciones telemáticas.

Como respuesta a esta necesidad se han creado instrumentos capaces de acreditar la identidad de los intervinientes en las comunicaciones electrónicas y asegurar la procedencia y la integridad de los mensajes intercambiados. Estos instrumentos son los certificados digitales, que son la adaptación de la identificación tradicional a la nueva realidad de una sociedad interconectada por redes de comunicaciones. De este modo, cada persona podrá realizar múltiples gestiones de forma segura a través de medios telemáticos y asegurando la identidad de los participantes en la comunicación.

Las tarjetas inteligentes nos proporcionan un mayor grado de seguridad ante la utilización de certificados digitales. El sistema operativo del chip está diseñado para que ningún software malicioso pueda extraer información sensible del mismo. Las operaciones más delicadas, como pueden ser la generación de claves privadas o de firmas electrónicas, se realizan siempre en el interior del chip de la tarjeta, y nunca se descarga esta responsabilidad a aplicaciones informáticas del ordenador.

Por tanto podemos estar seguros de que nadie puede robarnos información de la tarjeta inteligente para después realizar usos fraudulentos en cualquier momento.

No obstante, existen dos momentos delicados mientras se usa una tarjeta inteligente: cuando se introduce el PIN en el ordenador para que viaje hasta la tarjeta y cuando se firma un documento electrónico (se envía un resumen de éste a la tarjeta).

En el peor de los casos, podemos imaginar que un software malicioso copiara el PIN en ese momento. Para poder hacer un uso fraudulento necesitarían también robarnos la tarjeta inteligente, pues a diferencia de las tarjetas de crédito, es necesario tener físicamente la tarjeta y su PIN asociado para hacer un mal uso del mismo.

El otro momento delicado es el de la firma electrónica cuando estamos realizando una gestión a través de Internet. Por ejemplo, imaginemos que nos conectamos al portal web del Ministerio de Educación para rellenar una solicitud de becas. En un momento determinado la aplicación informática nos pedirá que firmemos de forma electrónica dicha solicitud y nos pedirá el PIN de la tarjeta. Un software malicioso podría presentar al chip de la tarjeta otro documento diferente al de la solicitud de beca para, una vez firmado, recogerlo a través del portal al que nos hemos conectado. En este caso, para evitar firmar algo diferente de lo que esperábamos, es necesario

asegurarnos de estar conectados al portal del Ministerio de Educación y no a otro. Esto es posible, pues todas las Instituciones que nos soliciten firmar electrónicamente algún documento deben hacerlo a través de una página segura: la dirección será una https (en lugar de una http) y aparece un candado o una llave en nuestro navegador indicándonos que es un lugar seguro.

Por tanto, se deben tomar precauciones cuando nos conectamos a Internet para realizar transacciones, es decir, cumplir con unos mínimos requisitos de seguridad.

En mi opinión creo que en un futuro cercano veremos una proliferación de tarjetas inteligentes para distintos usos ya que representan una tecnología segura, conveniente, permiten almacenar y procesar distintos tipos de datos y todo ello con un coste aceptable. La utilización de estas tecnologías tiene múltiples ventajas como la eliminación de muchos costes (papel, toner, mensajería), la reducción de espacio físico para almacenamiento junto con una disminución de los tiempos de búsqueda de documentos.

Desde el punto de vista informático se han ido dando cada vez más pasos encaminados a que se pueda hacer uso de las tarjetas inteligentes sin que ello suponga un gran esfuerzo de programación ni necesite de un amplio conocimiento de las mismas. Un ejemplo de ello es el reciente lanzamiento de Microsoft, su sistema operativo Windows Vista incluye el sistema criptográfico CNG (Cryptography Next Generation) que es la nueva API criptográfica que sustituye a la existente CryptoAPI y que proporciona nuevas funcionalidades además de dar soporte a nuevos algoritmos como la criptografía de curva elíptica.

8. Referencias

[PCSC05] **Interoperability Specification for ICCs and Personal Computer Systems**, PC/SC Workgroup 2005.

[CERE03] **Manual de usuario CERES**, Fábrica Nacional de Moneda y Timbre 2003.

[RAEF03] **Smart Card Handbook**, *Rankl W. y Effing W.*, Wiley & Sons 2003.

[EVER94] **Smart Card Tutorial**, *Everett D.*, Smart Card News 1992-94

[JUGU98] **Smart Card Developer's Kit**, *Jurgensen T. y Guthery S.*, Macmillan 1998.

[CHAU04] **Smart Cards in .NET**, *Chauhan D.*, ASP Free 2004.

[GAL103] **Encryption in .NET with CryptoAPI Certificate Stores**, *Gallant M.I.*, Microsoft 2003.

[GAL203] **Extending .NET Cryptography with CAPICOM and P/Invoke**, *Gallant M.I.*, Microsoft 2003.

[BEYE02] **Get Up and Running with .NET Cryptography Providers**, *Beyer D.*, ASP Free 2002.

[FER105] **El cifrado. Algoritmos de cifrado simétricos y asimétricos. La función hash. El notariado**, *Fernández Muñoz J.,* Temario de ingreso en el CSSTI de la Administración del Estado 2005.

[FER205] **Servicios de autenticación: El rol de los certificados digitales**, *Fernandez Muñoz J.*, Temario de ingreso en el CSSTI de la Administración del Estado 2005.

[MEND05] **El ABC de los documentos electrónicos seguros**, *Mendívil I.*, SeguriData 2005.

[ANGE00] **Criptografía para principiantes**, *Angel Angel J.J.*, SeguriDATA 2000.

[GIRA01] **Introduction to Smart Cards**, *Giraud J.L.*, MacCrypto 2001.

[LICO01] **Mitos y realidades sobre las tarjetas inteligentes**, *Licón A.*, Schlumberger 2001.

[MICR05] **Smart Card Deployment at Microsoft**, *Technical White Paper*, Microsoft 2005.