.Universidad del Valle de México :: Rectoría Institucional. Episteme No. 11. Año 3, abril - septiembre 2007
Dirección Institucional de Investigación e Innovación Tecnológica

enfoque interdisciplinario

Aplicaciones electrónicas seguras basadas en tarjetas inteligentes

Luis Adrián Lizama Pérez
Roberto León Oramas
UVM-Querétaro

Resumen
En este artículo se describen dos aplicaciones que fueron desarrolladas para elevar la seguridad de los servicios de redes y de comercio electrónico, por medio de tecnologías que utilizan el pequeño procesador de una tarjeta de plástico conocida como tarjeta inteligente. Además se analizan las ventajas de incluir en la tarjeta alguna tecnología biométrica, es decir, la comprobación de alguna característica morfológica del usuario.

. . . . . .

. . . . . .

Introducción

La importancia de la seguridad de la información digital va en aumento. En la era de la interconectividad y del comercio electrónico, surgen nuevas formas de delito digital debido a la posibilidad de suplantar la identidad de los usuarios, la pérdida de confidencialidad en las comunicaciones y la integridad de la información transmitida. Estas amenazas causan graves pérdidas económicas [11].

Una forma de reducir los riesgos de perder la información sensitiva, como pueden ser números confidenciales o el número de tarjeta de crédito, consiste en procesar estos datos en un sistema portátil, el cual puede ser una tarjeta personal con procesador (también llamado *chip*). Actualmente los procesadores de las tarjetas inteligentes son capaces de ejecutar algunos algoritmos criptográficos. En las secciones siguientes vamos a presentar dos aplicaciones que utilizan tarjetas inteligentes: el primero es un protocolo de autenticación para redes locales llamado *Kerberos* y el segundo es un *monedero electrónico* basado en el protocolo de comercio de Transacciones Electrónicas Seguras (SET).

arriba

. . . . . .

Kerberos

Kerberos fue diseñado en el Tecnológico de Massachusetts como un medio para que las contraseñas de los usuarios no tengan que viajar en la red, donde posibles intrusos pueden capturarlas. En su lugar, Kerberos proporciona credencial es (también llamadas tickets) a los usuarios, que los obtienen de un Servidor de Distribución de Tickets. Estos tickets le proporcionan a los usuarios acceso a los *hosts* (estaciones de trabajo) y servicios de red. Los tickets se envían encriptados por la red, de tal forma que si alguien los captura no pueda usarlos [1].

Limitaciones de Kerberos

Kerberos contiene algunos problemas de seguridad, debido principalmente a su dependencia sobre las contraseñas que son seleccionadas por los usuarios. En el esquema que se propone en [1] la contraseña se reemplaza por un dato biométrico a partir del cual se obtiene la plantilla biométrica del usuario, que es almacenada y usada por el procesador de la tarjeta. Así, cuando un usuario intenta obtener acceso (a tiempo de *login*) es confrontado por la interfaz biométrica del sistema de autentificación.

Si la lectura biométrica del usuario que denominamos Pb' y la plantilla biométrica almacenada en la tarjeta Pb, son similares de acuerdo con el algoritmo de comparación biométrica, se sigue que la llave Ku, también guardada en la tarjeta, se puede usar para decriptar el ticket de autentificación de Kerberos. Por lo tanto, el usuario queda autentificado y podrá obtener autorización para recibir algún servicio de red, por ejemplo: *ftp, telnet, rsh*. Con esto queda previsto que ante pérdida o robo de tarjeta, ningún intruso podrá tener acceso a los servicios de red, ya que implica obtener simultáneamente la tarjeta, los datos biométricos del usuario, y la llave de seguridad del usuario de Kerberos Ku. Adicionalmente, el usuario debe ingresar el NIP de la tarjeta antes de obtener los servicios deseados [3].

Kerberos y las tarjetas inteligentes

Se han reportado problemas inherentes a Kerberos que no pueden evitarse sin el uso de *hardware* de propósito especial, sin importar el diseño del protocolo [1][3]. En el protocolo de Kerberos, la llave del cliente, que denotaremos *Kc*, es compartida por el usuario y el *KDC* (Centro de Distribución de Llaves de Kerberos). *Kc* es derivada del *password* del usuario, después de aplicar una función *hash* sobre éste. La estación de trabajo lee el password, lo convierte en *Kc*, y lo usa para decriptar el ticket de autentificación *TGT*. El protocolo aparece ilustrado en la Figura 1, donde se observa la fase de autentificación de Kerberos.

Diseño del protocolo de intercambio de mensajes

Los objetivos de diseño del protocolo quedan de la siguiente manera: 1) Usar bits aleatorios para generar *Kc* . 2) Almacenar la llave del usuario *Kc* en una smart card. 3) Activar la smart card por medio del PIN del usuario. 4) Almacenar la plantilla biométrica maestra del usuario *Pb* en una smartcard y ejecutar el mecanismo de correlación *on card*. 5) Decriptar el *TGT* en una smart card. 6) No modificar el *KDC* de Kerberos. La Figura 1, muestra el esquema general del protocolo de pre-autentificación. Cuando un usuario intenta obtener acceso a una estación de trabajo, inserta su smart card, y también introduce su PIN.

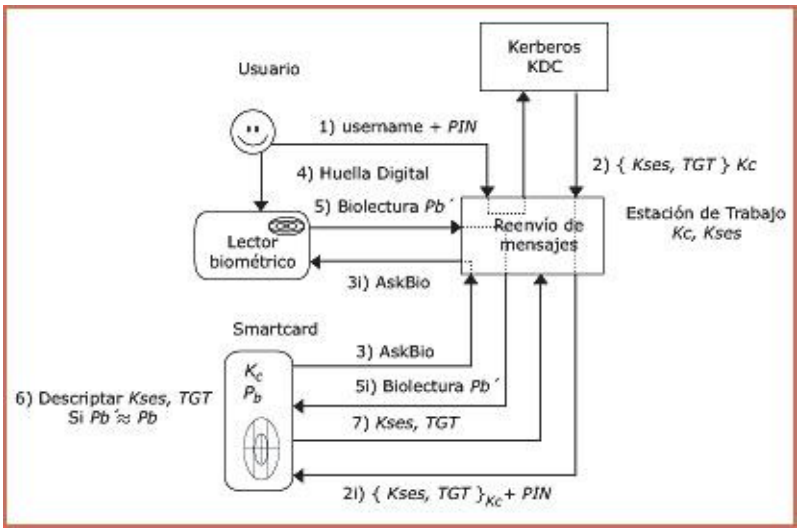

Figura 1. El Protocolo de Pre-Autentificación Smart-PIN Biométrica

Por su parte, la estación envía una solicitud al *KDC*. 2) El *KDC* genera un *TGT* y una llave de sesión *K ses*, los encripta con *Kc*, y los envía a la estación de trabajo, quien los reenvía a la smart card (para su decripción y almacenamiento del *TGT*) junto con el PIN del usuario. 3) La smart card verifica el PIN del usuario y solicita al usuario su identificación biométrica *Pb'*. 4) El usuario ingresa su dato biométrico por medio del sensor biométrico. La estación de trabajo recibe la lectura biométrica y la reenvía a la smart card. 6) El chipcard compara la lectura *Pb'* con la plantilla biométrica del usuario *Pb* almacenada en la tarjeta. Si la validación es exitosa, el chipcard procede a decriptar el *TGT* con la llave *Kc* también guardada en la tarjeta, si el *TGT* es decriptado correctamente, el usuario es autentificado (mediante *K ses*) y el acceso es permitido. El protocolo propuesto satisface las metas del diseño: *Kc* es una llave aleatoria, la tarjeta es activada por medio del PIN del usuario, la plantilla biométrica maestra *Pb* y la llave del usuario *Kc* son almacenadas en la tarjeta, el ticket es decriptado en la tarjeta y el *KDC* no sufre modificaciones.

Resultados obtenidos

La Interfaz para la Programación de Aplicaciones Biométricas de la Java Card nos brinda una interfaz para correlación de datos biométricos y registro de plantillas maestras [6]. Debido a que no está disponible la API Biométrica, nos sujetamos a un prototipo basado en el diseño descrito en el apartado anterior. Para el resto de la implementación construimos un prototipo de Kerberos escrito en Java (mediante la extensión criptográfica de Java) y basado en el diseño previamente modelado y validado [1]. Para ello se instaló el kit Cyberflex para Linux, procesador intel x86, Linux x86: RedHat 6.1x86, así como la interfaz de comunicación PCSC-lite y los drivers del smart reader de Towitoko [7]-[10]. Se utilizó la tarjeta Cyberflex Access 16K de SchlumbergerSema y con estos recursos se activó el demonio pcscserver para la comunicación con la tarjeta:

>run pcscserver >run Xcard >run pay

Por medio del programa de comunicación de datos con la tarjeta *pay*, incluído en el paquete *Cyberflex i386_linux2*, se probaron comandos APDU para la prueba del *Cardlet*. Escribimos el código Java JCE JPCSC del Emulador de Kerberos: *KDC_module.java* es el centro de distribución de llaves de Kerberos, C_module.java es el cliente de Kerberos y el *Cardlet* dentro de la tarjeta es *Smart.java. KDC_module* y *C_module* se comunican a través de un canal seguro por medio de las funciones criptográficas de JCE. A su vez, *C_module* se comunica con *Smart* por medio de la librería JPCSC de IBM. Aquí la clase *Context* habilita las funciones PCSC relacionadas con la conexión/desconexión de los servicios PCSC y los *card readers*.

Trabajos futuros

Una tecnología reciente desarrollada por IBM se basa en un coprocesador criptográfico, que puede elevar la seguridad del servidor de Kerberos. Un coprocesador seguro es un dispositivo computacional confiable para la ejecución correcta de software, a pesar de posibles ataques físicos. Por ejemplo el coprocesador IBM 4758 es un dispositivo PCI card resistente y sensitivo a infiltraciones, la tarjeta detecta intentos de apertura, intentos de penetración, ataques de temperatura y ataques de radiación. Para un análisis de seguridad de esta tecnología incorporada a Kerberos [1].

arriba

. . . . . .

Desarrollo de un *monedero electrónico seguro*

Las soluciones informáticas que se basan en monederos electrónicos, utilizan la tecnología de las tarjetas inteligentes para guardar y procesar las operaciones sobre saldos de dinero digital a partir de un depósito inicial. El prototipo que hemos desarrollado funciona como sistema de pago electrónico, protegido contra ataques sobre confidencialidad e integridad de los datos, ya que éstos se transfieren encriptados desde un procesador portátil embebido en la tarjeta de plástico, el cual es capaz de ejecutar código Java, una especificación que se ha desarrollado en la industria orientada a la programación generalizada de dispositivos portátiles.

Hoy en día muchos dispositivos, como celulares y electrodomésticos, pueden programarse utilizando las especificaciones del lenguaje Java. Otros tipos de tarjetas operan con sistemas operativos propietarios y también pueden implementar aplicaciones protegidas contra robo o clonación, pero estas soluciones son privadas (por tanto desconocidas) y algunas de ellas requieren *hardware* adicional (lectores de tarjetas con teclado o *keypads*) para su operación [2].

Limitaciones de los monederos

Una variedad de aplicaciones de comercio electrónico se está desarrollando sobre las ventajas de las tarjetas inteligentes, principalmente, debido a la necesidad de un ambiente portátil de computación. Sin embargo el desarrollo de estas aplicaciones queda limitado si no se implementan mecanismos específicos para la protección de los datos guardados en su memoria y transferidos a través de una red de computadoras.

Las amenazas aumentan debido a *crackers* que roban información sensitiva de los usuarios a través de Internet, haciendo que cada vez sean más frecuentes las pérdidas por fraude electrónico. A pesar de lo anterior, las tarjetas inteligentes pueden servir para implementar un sistema de autentificación que al combinarse con protocolos de comercio electrónico, pueden elevar la seguridad de las comunicaciones a través del sistema y de la red. En este trabajo presentamos la implementación de un prototipo para el funcionamiento de un monedero electrónico seguro, que puede servir para el desarrollo de otras aplicaciones basadas en el uso de estas tarjetas con procesador.

SET (Protocolo de Transacciones Electrónicas Seguras) es una especificación de seguridad abierta diseñada para proteger las transacciones con tarjetas de crédito en Internet. SET hace seguras las transacciones en línea mediante el uso de certificados digitales. Con SET se puede verificar que tanto clientes como vendedores estén autorizados para realizar o aceptar un pago electrónico.

A pesar de lo anterior, es posible explotar algunas vulnerabilidades de SET para la captura de datos importantes del usuario: Si un adversario tiene acceso privilegiado a la estación de trabajo del cliente, es relativamente simple instalar un programa que capture su número de tarjeta de crédito. Por lo tanto, al instalar los programas que procesan estos datos en la memoria de una tarjeta, fuera de la estación de trabajo, podemos reducir este riesgo, evitando la suplantación de los programas por medio de Caballos de Troya y la consecuente pérdida de datos sensitivos.

Mediante la utilización de la tecnología de tarjetas inteligentes dentro de un protocolo de transacciones electrónicas, se puede fortalecer la seguridad del sistema de pagos a través de la red. Además es posible implementar una versión simple del mismo para el desarrollo de un prototipo y de la infraestructura necesaria para efectuar los pagos electrónicos.

El protocolo SET

En el protocolo SET, la llave Ks del comprador, se utiliza para encriptar los datos que van dirigidos al Banco, los cuales no deben ser vistos por el vendedor. En lo que sigue vamos a referirnos a estos datos por su abreviatura: PI, DS y OIMD. El primer componente es la Información de Pago (PI), donde van los detalles de la compra y el número de tarjeta de crédito del comprador. El segundo componente constituye la Firma Dual (DS, el mecanismo más estratégico de este protocolo) y el último de ellos es la huella de la Información de la Orden (OIMD).

El mensaje encriptado E Ks [PI, DS, OIM] se envía al vendedor junto con la llave Ks, que a su vez es encriptada pero con la llave pública del Banco, o sea Kub, de modo que sólo el Banco puede obtener la llave Ks. Junto a estos componentes se transfieren otros que serán necesarios para su procesamiento (Véase la Fig. 2).

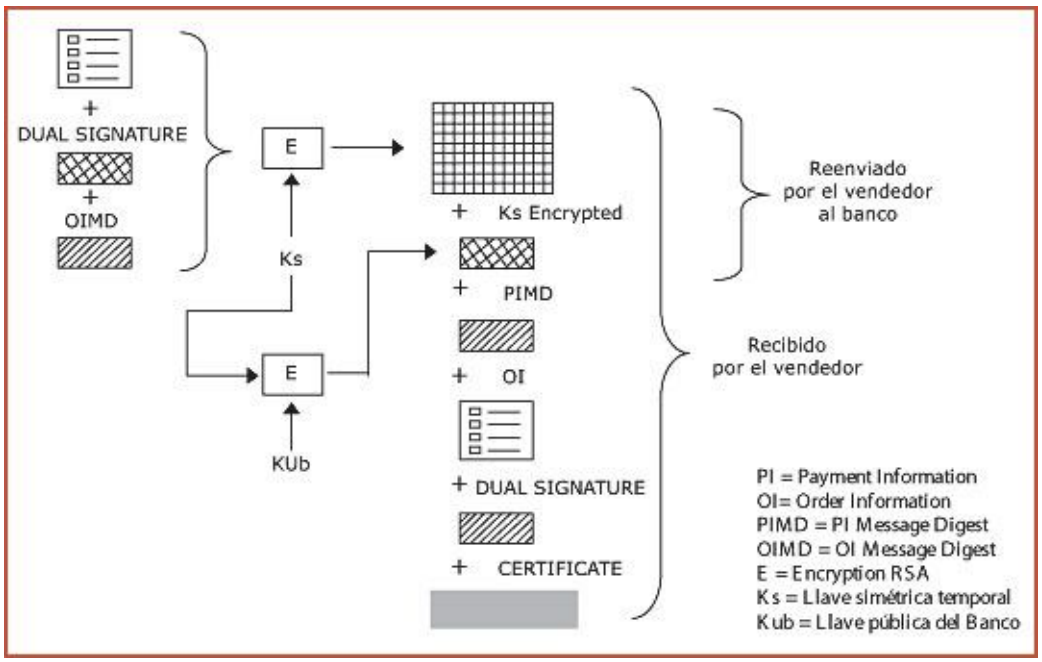

Figura 2. El comprador envía la solicitud de compra

Un adversario puede instalar un programa que capture PI. Para resolver este problema es preferible calcular fuera de la estación de trabajo (véase [2]) la huella SHA-1 de la Información de Pago (PIMD) y encriptar {PI, DS, OIMD} por medio del procesador de la tarjeta. Por otra parte, las computadoras no pueden almacenar información de manera completamente segura. La información en un sistema se puede almacenar en memoria o en disco duro, pero ninguno es suficientemente seguro. Una llave (en este caso Ks) en la memoria de la estación de trabajo del comprador es vulnerable si la estación no está protegida o no es confiable. Una llave en el disco duro también es vulnerable porque un adversario puede lograr el acceso a él y leerlo o escribirlo. Una llave en memoria también es vulnerable si un adversario escanea la memoria. Por lo tanto, es preferible contar con un dispositivo de almacenamiento y procesamiento seguro fuera de la estación de trabajo.

En la Figura 3, presentamos nuestra versión del esquema anterior: Básicamente proponemos instalar las llaves Ks y KUb en el interior de la tarjeta. También proponemos transferir los procesos de encripción de PI y Ks a la tarjeta y, dado que PI se introduce directamente en ella, aquí se calculará la huella *SHA-1* de PI (es decir *PIMD*). Después se envía ésta a la Estación de trabajo, donde finalmente se calcula la Firma Dual (DS) a partir de PIMD. Hemos mantenido un enfoque de diseño orientado a reducir el número de operaciones en la tarjeta debido a sus limitaciones intrínsecas, sin embargo, el crecimiento de esta tecnología permitirá calcular *DS* completamente en la tarjeta.

Una vez que se integra la solicitud del Comprador, se envía al Vendedor para su verificación, quien a su vez reenvía algunos componentes al Banco. Para realizar la verificación, el Vendedor calcula la huella SHA-1 de la Orden (es decir OIMD), después la concatena al PIMD que ha recibido y calcula otra vez el hash SHA-1, entonces obtenemos *POMD*. Por otra parte, el Vendedor obtiene la llave pública del Comprador (KUc), porque esta llave se encuentra en el Certificado del Comprador. Por lo tanto, procede a decriptar DS con esta llave a fin de obtener POMD. Por último compara este resultado con POMD', si ambos son iguales el Vendedor ha verificado al Comprador.

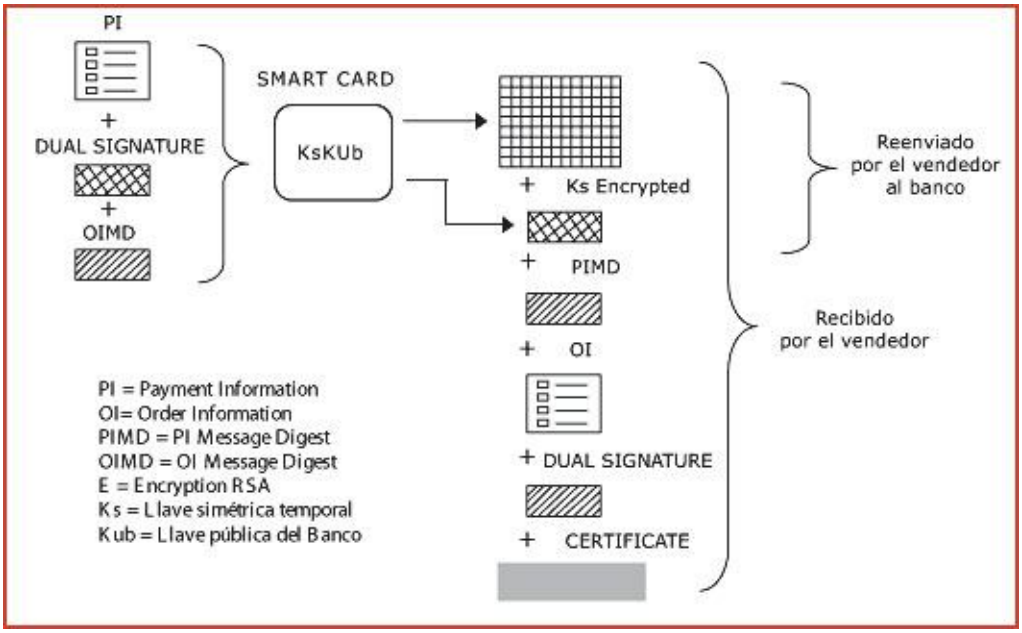

Figura 3. La solicitud del comprador por medio de una tarjeta

El vendedor no tiene acceso a la llave Ks que protege a PI, porque está encriptada con la llave pública del Banco y solamente el Banco puede obtener Ks mediante KRb (la llave privada del Banco). Un proceso de verificación similar se ejecuta en la computadora del Banco. En este diseño observamos que la Información de Pago (PI) que incluye el número de tarjeta de crédito queda protegida desde su almacenamiento hasta su transferencia por la red. Más aún, es preferible mantener el número de la tarjeta de crédito preinstalado en la tarjeta sin necesidad de que sea introducido de manera explícita por medio de un teclado ó *keypad*.

Diseño

La fortaleza de SET radica en el cifrado de datos con algoritmos de llave pública y la creación/verificación de DS y huella digital (MD), sin embargo, en un equipo conectado a Internet se corre el riesgo de que alguien obtenga estos datos antes de ser procesados y pueda, entre otras cosas obtener el número de cuenta y NIP del comprador. Por ello se requiere de dispositivos portátiles que puedan realizar el proceso de cifrado, así como, la creación de DS y MD [2].

La Figura 4, muestra el esquema general del sistema. En 1) una vez iniciada la operación para realizar el pago, el usuario ingresa el OI. Aquí el comprador entra a la página de un vendedor e ingresa al carrito de compras. El comprador selecciona los productos que va a adquirir y pagará de manera electrónica. Terminada la selección de los productos a adquirirse el comprador inserta su tarjeta en el lector de la estación de trabajo y pulsa el botón pagar en el carrito de compras. El carrito de compras inicia peticiones de acceso a la Tarjeta. El carrito de compras pide se ingrese PIN (esta sería una barrera más, y sin exponer el NIP del comprador). El comprador ingresa PIN y pulsa el botón Aceptar. El carrito de compras envía PIN a la Tarjeta para ser validado y completar el proceso de acceso a ésta. La Tarjeta recibe, valida y acepta PIN.

En 2) la tarjeta calcula PIMD y en 3) envía PIMD y PIN_válido al carrito de compras, el cual al recibirlos calcula en 4) el OIMD y DS; después los envía a la tarjeta en 5), para que realice las encripciones: en 6) [PI, DS, OIMD] con Ks, en 7) [Ks] con KUb (la llave pública del Banco) y en 8) [DS] con KRc (la llave privada del cliente); en 9) se envía el resultado de las encripciones de 6), 7) y 8) al carrito de compras para que en 10) sean encriptados con KUv (la llave pública del vendedor) junto con el OI, el PIMD y el Certificado del Cliente; enviándose al vendedor en 11) el paquete KUv[6), 7), PIMD, OI, 8), Certificado del Cliente]. De este modo los componentes K s [PI, DS, OIMD], PIMD, K Ub [K s] y K Rc [DS] provienen de la tarjeta. El certificado del Cliente (el certificado del comprador) puede estar también en la tarjeta.

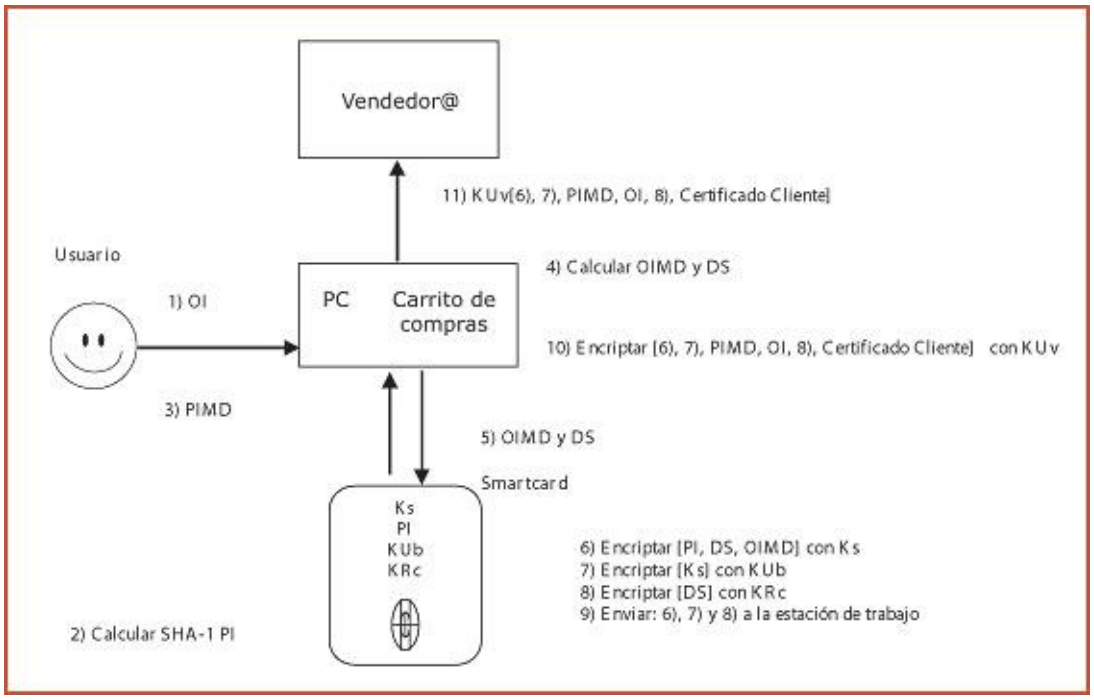

Figura 4. SET smartcard integrado

El sistema del prototipo está formado por tres aplicaciones: una aplicación Servidor que registra las transacciones y verifica la información procedente de la Tarjeta. Una aplicación cliente (host) que administra las transacciones con la smart card y reenvía al Servidor los datos de verificación de la smart card. Una aplicación para la smart card (cardlet) que recibe, valida y procesa las peticiones del host y le notifica el estado Final del proceso. Toda la información que se envía al Servidor desde la smart card va codificada (cifrada o encriptada) utilizando el Estándar de Encripción de Datos (DES). Se realizaron pruebas de compatibilidad entre la tecnología Java de las tarjetas, la computadora y la comunicación a través de la red, tomando como referencia las características criptográficas que el fabricante enuncia. En la Tabla 1, se da el nombre de la prueba y descripción.

Tabla 1.- Lista de pruebas y descripción realizadas

Prueba	Descripción
Creación de llaves DES, 3DES y RSA.	Comprueba la creación de llaves en la Javacard y determina las longitudes permitidas de las mismas.
Pruebas de cifrado/descifrado en la Javacard.	Realiza el cifrado y descifrado en la Javacard y determina los pasos requeridos para ello
Criptografía Javacard con llaves DES, 3DES y RSA java.	Verifica si las llaves creadas en Java para PC son compatibles con los mecanismos criptográficos de la Javacard
Cifrado/descifrado Java PC y Javacard	Prueba que el cifrado en Java para PC se logre descifrar en la Javacard ; posteriormente se cifró en la Javacard y descifró en Java PC.

Una vez construido el *Applet*, se compila con *java* y se obtiene un *.class*; a este se le procesa con *mksolo* ; el cual genera un *.bin*; siendo este el que se carga a la tarjeta. El Cardlet cargado en la Tarjeta, encripta PI y Ks, y calcular SHA-1 de PI. Para ello éste debe ser cargado en la tarjeta y registrado en el JCRE de la misma. Cargado el Cardlet en la tarjeta, y haberse seleccionado, se mantiene en espera de un APDU entrante, al entrar uno comprueba si es para el, y verifica el Número de Identificación Personal (PIN) [7]-[10].

Recolección y análisis de datos

Con las pruebas preliminares se determinó el mecanismo criptográfico a implantarse en el prototipo, así como las características del protocolo de comunicación. En particular, la capacidad limitada de las tarjetas en su memoria de 16K no permitió incorporar la criptografía de llave pública RSA. Por esta razón se utilizó DES como

mecanismo de codificación en modo ECB y el mecanismo MAC (Código de Autenticación de Mensajes) para verificar la integridad de datos. El protocolo de comunicación de datos se implementó considerando la información del usuario que accede al servicio (véase [2]), quedando de la siguiente manera:

1. Cuando el usuario se presenta en el módulo de servicio, el personal de administración utiliza la aplicación host para activar la aplicación Smartcard de la tarjeta por medio del lector conectado. Entonces la aplicación host le requiere a ésta su identificador para que sea verificada por la aplicación Servidor. La aplicación host recibe el mensaje {Num, {IDCARD, MAC} kCard } kServ . Donde Num es el identificador del usuario.
2. La aplicación host reenvía el mensaje recibido, es decir: {Núm, {IDCARD,MAC} kCard} kServ a la aplicación Servidor para que ésta verifique la tarjeta del usuario.
3. La aplicación Servidor recibe {Núm, {IDCARD,MAC} kCard} kServ, descifra el paquete, verifica la integridad de los datos por medio de MAC y devuelve el mensaje *Tarjeta_Aceptada* junto con los datos del propietario a la aplicación host.
4. La aplicación host recibe el mensaje *Tarjeta_Aceptada*, muestra los datos y solicita que el usuario ingrese su número de identificación personal PIN.
5. El usuario ingresa su PIN y la aplicación host lo envía a la aplicación Smartcard para ser validado.
6. La aplicación Smartcard recibe el PIN y lo valida.
7. La aplicación Smartcard devuelve al host el mensaje *PIN_Correcto*.
8. La aplicación host recibe *PIN_Correcto*, muestra la interfaz para seleccionar la operación (incremento ó decremento) y un campo para asignar el monto.
9. Se selecciona la operación, se asigna el monto y se pulsa el botón *Aceptar*.
10. La aplicación host envía a la aplicación Smartcard el mensaje {Acción, Monto}. Donde Acción es la operación y Monto es el valor en pesos de la operación.
11. La aplicación Smartcard recibe {Acción, Monto}, y procesa la Acción aplicando el Monto.
12. La aplicación Smartcard devuelve {Núm, {Acción, Monto, MAC} kCard } kServ a la aplicación host.
13. La aplicación host recibe {Num, {Accion, Monto, MAC} kCard } kServ y lo envía al Servidor.
14. La aplicación Servidor recibe {Núm, {Acción, Monto, MAC} kCard } kServ , descifra el paquete, verifica su MAC y registra la Acción junto con el Monto, en la base de datos de las tarjetas de los usuarios registrados.
15. Finalizadas las transacciones, la aplicación host desactiva la aplicación Smartcard.

Resultados obtenidos

En el desarrollo del proyecto se usaron dos computadoras conectadas en red: Una para ejecutar la aplicación host y otra para la aplicación Servidor, un lector *Towitoko* de tarjetas, 5 tarjetas de prueba *Cryptoflex* de 16K. Todo el software que se utilizó es de distribución libre: *Cyberflex* versión 2.1 para Linux, PCSC y Java (versiones 1.2 y 1.4). Los resultados obtenidos muestran la funcionalidad del prototipo y la validez del protocolo de transacciones. El prototipo funciona en plataforma Linux por lo que aún se estudia su viabilidad para implantar una red de puntos de acceso al servicio. A continuación se enlistan los principales resultados que obtuvimos:

- Se comprobó que existe compatibilidad entre las llaves generadas con JCE de la PC y las llaves generadas con Javacard.
DES: Hay compatibilidad; las llaves tienen 8 bytes de longitud y pueden ser creadas en la computadora o la tarjeta.
3DES: No hay compatibilidad, en el host son de 24 bytes y en la smart card son de 16 bytes.
RSA: No se encontró en la smart card la clase para crear estas llaves, se crearon en el host y se cargaron al cardlet inicializando el cifrador. En el *host* se creó la llave de menor tamaño posible equivalente a 512 bits.
- Existe compatibilidad del algoritmo DES en el modo ECB para el Cifrado en el *host* y el descifrado en la Javacard.
- Existe compatibilidad criptográfica entre el cifrado de la Javacard y el descifrado en el host: Falla DES al cifrar en la Javacard y descifrar en el host, el error fue de relleno (padding). Este error fue corregido haciendo los paquetes a cifrarse en Javacard en múltiplos de 8 e indicando en JCE de la PC que no usara *padding*.
- No se completaron las pruebas del cifrado/descifrado con algoritmo 3DES por incompatibilidad de las llaves entre el host y la Javacard.
- El monedero soporta un balance máximo de 32767, las transacciones se hacen en base a montos con números enteros.
- Actualmente se está terminando una librería de comunicación con la tarjeta que permita utilizar el puerto USB de una computadora portátil, ya que una gran variedad de modelos no incluye el puerto COM. Esta librería, la cual ha sido implementada en Java permite sustituir al programa de comunicación Pay del CITI.

arriba

. . . . . .

Conclusiones

En el primer esquema se sugiere la integración de la tecnología biométrica con las tarjetas inteligentes en el protocolo de Kerberos. La principal ventaja radica en la confiabilidad del protocolo: si un intruso intercepta una cadena de datos no podrá implementar un ataque por repetición debido al dispositivo bio-inteligente que se encuentra conectado a la estación de trabajo. Tampoco podrá reconstruir la llave Kb a partir de estos datos, dada la cantidad de entropía de la llave, formada por bytes aleatorios que la hacen prácticamente inalcanzable por medio de búsqueda exhaustiva, a pesar de los fragmentos de información que un atacante pueda disponer acerca de la misma. A diferencia de la mayoría de los protocolos de seguridad, aquí no se asume la seguridad de componentes de bajo nivel de las computadoras como son los sistemas operativos, ya que un adversario podría recargar el Kernel.

En la segunda aplicación se presenta un monedero electrónico cuya seguridad se basa en la dificultad de penetrar un dispositivo de cómputo portátil que está protegido con mecanismos de comunicación encriptados. Con ello podemos evitar que sea necesario procesar los datos en un ambiente inseguro como una computadora. En la tarjeta también se guardan las llaves o bien, se introduce el número de la tarjeta de crédito del comprador. Aunque se conocen algunos ataques a las tarjetas inteligentes, consideramos que el futuro de las transacciones electrónicas a través de la red se realizará desde la tarjeta personal del cliente o bien desde la tarjeta inteligente de su teléfono celular, utilizando algoritmos criptográficos.

Para elevar la seguridad de las aplicaciones que se basan en el uso de procesadores portátiles, hemos desarrollado el prototipo descrito antes, el cual permite la utilización de las técnicas criptográficas en el procesador de la tarjeta mediante la especificación del lenguaje Java. Además, el análisis del protocolo SET que se presentó y el diseño que se propone basado en la utilización de una tarjeta inteligente, podrá utilizarse para incorporar la tecnología de las tarjetas inteligentes en el protocolo SET. En nuestro diseño proponemos que Ks, KRc, KUb y PI se resguarden en la tarjeta. Por el momento, proponemos que PIMD sea calculado en la tarjeta y que se encripte [POMD] con KRc creando así DS en la tarjeta, con el objeto de asignar las operaciones críticas al procesador de la tarjeta. La llave Ks podrá encriptarse en la tarjeta con la llave pública del Banco guardada en su memoria.

Las pruebas realizadas nos han permitido verificar la efectividad de la librería JPCSC para la comunicación entre la computadora y la tarjeta. Fue posible codificar los mensajes enviados desde la tarjeta mediante el algoritmo DES y preservar su integridad hasta alcanzar la aplicación Servidor, mediante el algoritmo de autenticación de mensajes MAC. Por el momento hemos utilizado el algoritmo DES en el modo ECB y estamos realizando pruebas con el modo CBC. Las pruebas en modo ECB demuestran la compatibilidad de los algoritmos de encripción DES ECB de la Javacard y la Extensión Criptográfica de Java (JCE) en el Host. No hemos incluido pruebas de rendimiento, sin embargo no se aprecia una baja notable en el desempeño general del sistema que pueda atribuirse a la ejecución de las operaciones criptográficas. Los resultados obtenidos se basan en un prototipo construído completamente siguiendo la especificación del lenguaje Java incluídas las comunicaciones a través de la red. A partir de los resultados obtenidos se desarrollarán otros servicios, ya que puede utilizarse no sólo como monedero sino también en servicios de biblioteca, cafetería, impresión, servicios médicos, etc.

arriba

. . . . . .

Referencias

[1] Lizama P. L., Gómez R. (2003). *Autenticación Biométrica On Card en el Protocolo de Kerberos*. Memoria del Segundo Congreso Iberoamericano de Seguridad Informática. México, pp. 249-266
[2] Lizama P. L., León R. (2004). *Comercio Electrónico Seguro. Memoria Electrónica del Primer Congreso Nacional de Informática y Sistemas*. México.
[3] Itoi N., Honeyman P., Smartcard Integration with Kerberos V5. *CITI Technical Report 98-7*, 1998.
[4] Bella G. (1997). Formal Analysis of the Kerberos Authentication System. *Journal of Universal Computer Science*, Vol. 3, pp.1337-1381
[5] Steiner J., Neuman B. Kerberos (1988). An authentication service for open network systems. *Proceedings Winter USENIX*, Conference, Dallas, February, Usenix.
[6] BioAPI Specification (2001). *The BioAPI Consortium.* March 16, V 1.1
[7] The source for Developers – A Sun Developers Network Sites. http://wireless.java.sun/Javacard/ articles/Javacard02/. An *Introduction to Java Card Technology – Part 2, The Java Card Applet. Sun Microsystems.*
[8] Cryptoflex TM Cards Programmer's Guide. Cyberflex Access SoftwareDevelopment Kit 4.3. C300474_rev1.
[9] Cyberflex TM Access Software Development Kit Release 3C. Cyberflex Access Programmer's Guide. (2000) C300451.
[10] Java Card Applet Developer's Guide (1998). SUN Microsystems. Rev 1.10, July 17.
[11] Stallings, J. (1998). Cryptography and Network Security, Prentice Hall.U.S.A., 3 th Ed.
[12] Jerdoney R., et al (1998). *Implementation of a Provably Secure, Smartcard-based Key Distribution Protocol*. CITI Technical Report.
[13] Giampaolo B.: Modelling Security Protocols Based on Smart Cards. Computer Laboratory, University of Cambridge.

arriba

Acerca de los autores

Luis Adrián Lizama Pérez
Ingeniero en Electrónica y Comunicaciones por el ITESM, tiene Maestría en Ciencias de la Computación con especialidad en Redes de Computadoras. Se ha desempeñado como supervisor de pruebas electrónicas automotrices, para el Grupo Condumex, en Querétaro, Qro., Profesor investigador en la Universidad Juárez Autónoma de Tabasco, impartido cursos en la maestría en sistemas computacionales de la UJAT. También participó en el Verano de la Investigación Científica para Docentes y en Foros Nacionales e Internacionales como ponente. Obtuvo el mérito académico UJAT 2006 y el mérito científico UJAT 2007, cuenta con el Perfil PROMEP y es miembro del Sistema Estatal de Investigadores de Tabasco. Actualmente se desempeña como profesor de asignatura de la UVM- Querétaro.

Roberto León Oramas
Egresado de la Universidad Juárez Autónoma de Tabasco con la Licenciatura en Sistemas Computacionales. Profesionalmente se ha desempeñado como capturista, programador y jefe de proyecto en áreas como programación, desarrollo de sitios Web, en organismos públicos y privados del estado de Tabasco. Asimismo, ha participado como ponente en foros nacionales e internacionales.

Guardar Imprimir

www.uvmnet.edu ISSN :: 1665 - 9317