Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 112-126
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Desarrollo de aplicaciones informáticas
Recibido: 7/11/2013 | Aceptado: 10/12/2013 | Publicado: 21/01/2014

Modelo para la extensión de las capacidades de procesamiento y memoria de tarjetas inteligentes Java Card

Model for the extension of the processing and memory capabilities of Java Card smartcards

Susana María Ramírez Brey [1*], Adonis César Legón Campo [1], Yusnier Valle Martínez [2]

[1] Centro de Identificación y Seguridad Digital. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Torrens, Boyeros, La Habana, Cuba. CP.: 19370
[2] Dirección de Formación Postgraduada. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Torrens, Boyeros, La Habana, Cuba. CP.: 19370

[*] Autor para correspondencia: smramirez@uci.cu
{alegon, yvm}@uci.cu

Resumen

Las tarjetas inteligentes poseen características distintivas como la portabilidad, dado su tamaño reducido y el bajo costo, para poder ser usadas a gran escala. Asociadas a estas características se encuentran las limitaciones de los recursos de hardware que poseen, relacionadas fundamentalmente con la capacidad de memoria y procesamiento de estos dispositivos. Estas y otras limitaciones propias de la tecnología Java Card, constituyen limitantes significativas para los desarrolladores de aplicaciones para tarjetas inteligentes. En este trabajo se presenta un modelo de desarrollo de aplicaciones para tarjetas inteligentes con tecnología Java Card, que permite extender las capacidades de procesamiento y memoria de estas, haciendo uso de los recursos de hardware de un ordenador, y con el que además se garantiza el ambiente seguro que es característico de este tipo de dispositivo. El modelo de desarrollo que se propone

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 112-126
http://rcci.uci.cu

provee un mecanismo para el almacenamiento de datos asociados a las aplicaciones de las tarjetas inteligentes fuera de esta y para la ejecución de algoritmos de alto costo computacional, que por el tiempo de ejecución y/o complejidad es más factible realizar fuera de la tarjeta. Con este nuevo modelo de desarrollo se pretende aumentar significativamente las aplicaciones y el uso de las tarjetas inteligentes, en ambientes conectados y controlados como empresas e instituciones.
Palabras clave: Java card, limitaciones, memoria, modelo de desarrollo, procesamiento, tarjetas inteligentes.

Abstract

Smartcard have distinctive features like portability, by the reduced size, and the low cost, in order to be used on a large scale. Associated to these characteristics, find out limitations of the hardware's resources, related fundamentally with memory and processing capabilities. These and others limitations of Java Card technology constitute significant limitations for the smartcard applications developers, and in general. In this work, is presented a smartcard application development model with Java Card technology that allows to extend memory and processing capabilities, making use of the computer's hardware resources. This model guarantees the safe environment that is characteristic of this device type. The proposed development model provide a mechanism for storage data associated to smartcard applications off- card, and for the execution of high cost computational algorithms, that for runtime or complexity is more feasible to perform off- card. With this new model is intended to significantly increase the applications and use of the smartcard, in connected and controlled environments like companies and institutions.
Keywords: *Development model, java card, limitations, memory, processing, smartcards.*

Introducción

Las tarjetas inteligentes o *smartcards*, surgidas en Europa en la década de los años 70, han contribuido al desarrollo tecnológico y al aumento de la seguridad en sistemas informáticos. El auge que han tenido las tarjetas inteligentes se debe en gran medida a la seguridad tanto física como lógica que brinda esta tecnología. Los datos sensibles que pueden manejar estos dispositivos como información personal del portador del documento, llaves secretas e información de las aplicaciones que estas manejan (*applets*), son protegidos por una combinación de *hardware* y software. Además de la seguridad, otra de las características de impacto que han tenido las tarjetas inteligentes es su flexibilidad. Tecnologías como Java Card, proveen un entorno de programación que permite crear e instalar nuevas aplicaciones según sean requeridas por el propietario de la tarjeta, lo que significa que es posible hacer tarjetas "a la medida" seleccionando para la tarjeta las aplicaciones que se adapten a las circunstancias y necesidades de cada persona.

Debido fundamentalmente a estas razones, ha ido en aumento el área de aplicación de las tarjetas inteligentes y son aplicadas ya no solo en el campo de la telefonía celular o en el comercio electrónico como en sus inicios, sino en áreas muy diversas entre las que se encuentran: transacciones seguras, identificación, firma digital, control de acceso, entre otras. Como una tendencia se ha observado también que varios países han comenzado a modernizar su documento de identificación nacional (Legón, *et al.*, 2013) hasta convertirlo en un documento de identificación electrónico (eID) con el uso de esta tecnología. De esta manera se pretende aumentar la utilidad del documento de identificación en distintas esferas de la sociedad, al brindar nuevos servicios a los ciudadanos basados en su eID (servicios de transporte, almacenamiento de historia clínica, seguridad social, etc).

Sin embargo, dada las características físicas de las tarjetas inteligentes definidas en (ISO/IEC, 2003; ISO/IEC, 2007), como su tamaño reducido (para que sea una tarjeta de plástico flexible e incrementar la seguridad del hardware), y su bajo costo (para poder ser vendidas en grandes volúmenes), existen limitaciones relacionadas con la capacidad de memoria y de procesamiento de estos dispositivos. La configuración de hardware típica para las tarjetas inteligentes existentes hoy en día consta de aproximadamente de 8/16-bit CPU, ~4-Kb RAM[1] (para la pila de ejecución, variables temporales y objetos transitorios), de 48-Kb - 96-Kb de ROM[2] (para el sistema operativo de la tarjeta, la máquina virtual y aplicaciones preinstaladas), y de 8-128Kb de EEPROM[3] (para el código de las aplicaciones dentro de la tarjeta, objetos persistentes y variables de clases) (SUN MICROSYSTEMS, 2008). Estas limitaciones del *hardware* de la tecnología impactan en gran medida en el uso extensivo y aplicabilidad de las tarjetas inteligentes, fundamentalmente para algunas áreas de aplicación.

La tecnología Java Card, diseñada por la empresa SUN Mycrosytems, casi desde sus inicios se convirtió en la plataforma dominante y tecnología líder para el desarrollo de aplicaciones para tarjetas inteligentes en el mundo(Sun Microsystems, 2008), llegando a desplegar en el año 2012 más de 10 billones de tarjetas inteligentes con esta tecnología (Oracle; Java Card Forum, 2012). Java Card fue diseñada de tal forma que ciertas construcciones de Java consideradas como demasiado complejas o no aplicables para la programación de tarjetas inteligentes no son incorporadas. Dentro de estos elementos del lenguaje se encuentran algunos referentes a la carga dinámica de clases,

[1] Random-access memory

[2] Read-only memory

[3] Electrically Erasable Programmable Read-Only Memory

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 112-126
http://rcci.uci.cu

el trabajo con hilos, clonado de objetos, etc. Además no se soportan los tipos de datos *char, double, float, long* y arreglos multidimensionales; el tipo *int* es opcional. Estas y otras características de Java no soportadas por la plataforma Java Card pueden encontrarse descritas en la especificación de la misma (Sun Microsystems, 2006). En (Cap, Maibaum y Heyden, 2001) se describen y referencian un gran número de proyectos Java Card que han demostrado claramente estas limitaciones del lenguaje y sus consecuencias. Estas características del lenguaje Java Card, unidas a las limitaciones propias del *hardware* de las tarjetas inteligentes, constituyen una limitante significativa para los desarrolladores de aplicaciones Java Card para tarjetas inteligentes y en general.

En el presente trabajo se abordan las limitaciones de los recursos de *hardware* de las tarjetas inteligentes, y las soluciones, aún insuficientes, que se han estado revisando por la industria en temas de diseño de las mismas. El objetivo fundamental de este trabajo es proponer un modelo de desarrollo de aplicaciones para tarjetas inteligentes con tecnología Java Card, que extienda las capacidades de procesamiento y memoria de estas, haciendo uso de los recursos de *hardware* de un ordenador y además se garantice el ambiente seguro que es característico en este tipo de dispositivo.

Materiales y métodos

Para el desarrollo de la presente investigación, se utilizaron los siguientes métodos teóricos y empíricos:

Analítico – sintético: Se seleccionó para el análisis del objeto de estudio y sus componentes, de manera que se pudieran describir sus características generales, así como las relaciones entre sus componentes.

Análisis histórico – lógico: Se seleccionó para analizar el proceso de desarrollo de aplicaciones para tarjetas inteligentes y la extensión de sus capacidades de procesamiento y memoria, desde su perspectiva histórica, evidenciando cómo ha evolucionado a través del tiempo.

Modelación: Se tuvo en cuenta para modelar los componentes del modelo de desarrollo de aplicaciones para la extensión de las capacidades de procesamiento y memoria de las tarjetas inteligentes, que se presenta como aporte fundamental de este trabajo.

Alternativas a las limitaciones de procesamiento y memoria de las tarjetas inteligentes.

Como posibles soluciones a las limitaciones mencionadas, la industria de software ha estado revisando temas de diseño que permiten incorporar múltiples chips de memoria en una tarjeta inteligente. Empresas como Gemalto han producido tarjetas gemelas (*twin cards*), con la incorporación de dos chips no conectados en una sola tarjeta, que no

pueden por tanto compartir datos entre ellos. Por otra parte la industria también ha ido desarrollado algunos dispositivos con una configuración de *hardware* superior (32-bit de CPU, 24-Kb de RAM, > 256-Kb de ROM, >128Kb de EEPROM, etc). Estos dispositivos demandan un soporte más completo del lenguaje, para lo cual Sun ha creado una nueva arquitectura Java Card (3.0.1 *Connected Edition*)(Sun Microsystems, 2009). Esta edición presenta un ambiente de ejecución significativamente enriquecido y una máquina virtual nueva como se detalla en (Sun Microsystems, 2008). A pesar de los avances que significa esta nueva generación de tarjetas inteligentes, el mercado aún es dominado por las generaciones anteriores debido fundamentalmente a los costos de producción que implican y en muchos casos la memoria EEPROM disponible sigue siendo insuficiente.

Otro de los enfoques en los que se ha estado trabajando incluye el uso de ordenadores de conjunto con las tarjetas inteligentes, para utilizar de esta manera los recursos de *hardware* que tienen estos. Como ejemplo temprano se encuentra Blaze (Blaze, 1995) que propone el uso de un ordenador potente con una tarjeta inteligente para un cifrado de clave simétrica, debido a que el ordenador proporciona mayor velocidad para el cifrado. Luego de esta primera publicación, se realizan otras (Blaze, Feigenbaum y Naor, 1999; Lucks, 1997) donde, tomando como base el *Remotely Keyed Encryption Protocol* propuesto por BLAZE (se usa *Remotely Keyed Encryption Scheme* como término equivalente) y el principio que plantea su autor, se analizan algunas debilidades de seguridad que posee y se formalizan requerimientos de seguridad que debe cumplir. Luego, Lucks (Lucks, 1999) describe un nuevo esquema : *Accelerated Remotely Keyed encryption scheme* (ARK) que plantea ser más eficiente y cumplir los mismos requerimientos de seguridad que (Blaze, Feigenbaum y Naor, 1999). A pesar de las diferencias y formalismos en cuanto a seguridad, todos estos algoritmos poseen el mismo principio de utilizar la capacidad de procesamiento de un host para el cifrado de grandes bloques de información, usando la seguridad de una tarjeta inteligente para almacenar la llave secreta con que se realiza el cifrado. Por otra parte, en (Cap, Maibaum y Heyden, 2001) se presenta un método para aumentar la memoria de datos de las *smartcards* utilizando memoria virtual. Esta solución posibilita el almacenamiento de datos asociados a las aplicaciones de las tarjetas inteligentes fuera de la misma, en un servidor, de manera que se extienda la capacidad de memoria de estos dispositivos y sea recuperable su información ante una pérdida o deterioro. Sin embargo esta solución provee una extensión solo de la capacidad de almacenamiento de las tarjetas Java Card, pero no de su capacidad de procesamiento. Además la documentación que se presenta es insuficiente y no existe acceso al código fuente de esta solución.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 112-126
http://rcci.uci.cu

En la revisión de la literatura de manera general, se observa que es recurrente el tema de las limitaciones de procesamiento y memoria de las tarjetas inteligentes, enmarcadas fundamentalmente en algunas de las áreas de aplicación más críticas por los recursos que necesitan. No obstante estas limitaciones, no se considera que ha sido explotada por la industria o la academia, la idea de proveer soluciones de software. Por tanto, surge la necesidad de crear un modelo de desarrollo de referencia para la extensión de las capacidades no solo de memoria, sino también de procesamiento, de las aplicaciones Java Card para tarjetas inteligentes.

Modelo de desarrollo tradicional de aplicaciones para tarjetas inteligentes Java Card

Existen varios elementos que intervienen en el desarrollo tradicional de aplicaciones para tarjetas inteligentes Java Card (Figura 1). Entre estos elementos se encuentran, primeramente, los *applets* Java Card que corren dentro del ambiente de la tarjeta gracias al intérprete Java Card de la máquina virtual. Además de los *applets* Java Card intervienen en el conjunto de una aplicación, los elementos correspondientes al terminal o *host*. Estos elementos a su vez podrían interactuar con una aplicación remota o *back- end* conformando un sistema completo que proporciona un servicio final a un usuario.

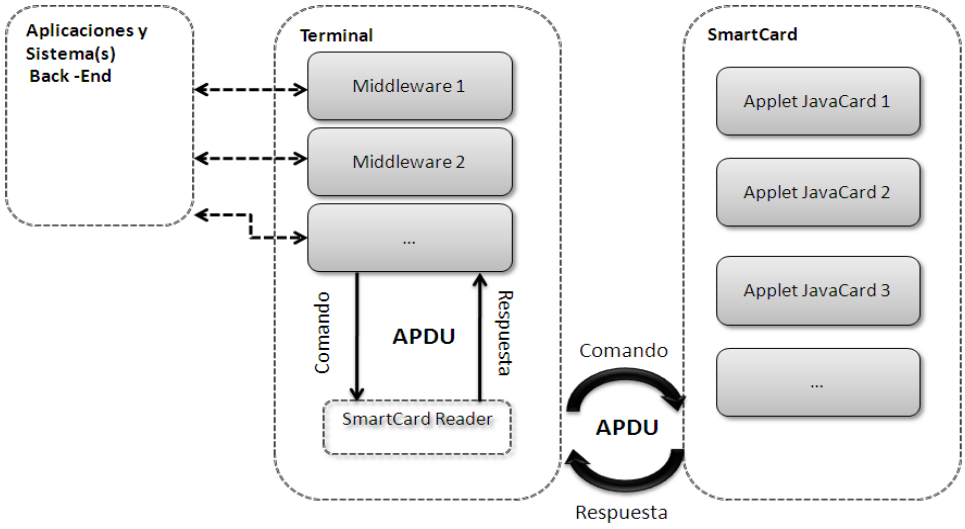

Figura 1. Modelo tradicional de desarrollo de aplicaciones para tarjetas inteligentes.

El terminal contiene las aplicaciones que comúnmente residen en un ordenador. Aquí se encuentran los *middlewares*, que no son más que librerías que exponen métodos de acceso a un *applet* determinado en una tarjeta inteligente. La

Grupo Editorial "Ediciones Futuro"
Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

comunicación entre un *middleware* y un *applet* Java Card determinado, se basa en el modelo de comunicación "Paso de Mensajes", usando comandos APDU[4]. Los *middlewares* envían comandos APDU con las peticiones a la tarjeta inteligente, mediante la comunicación con un lector de tarjeta (*Smartcard Reader*) que utiliza el estándar PC/SC[5] para la conexión con el ordenador. Dichos comandos APDU son recibidos y procesados por un *applet* específico que devuelve una respuesta en el mismo formato APDU. El modelo de desarrollo tradicional está centrado alrededor del procesamiento de peticiones entrantes, donde un *applet* dentro de la tarjeta obtiene una petición y luego la procesa. De esta manera una tarjeta inteligente puede verse como un servidor, nunca toma la iniciativa. A continuación se propone un modelo de desarrollo extendido, que permite obtener una mayor capacidad de procesamiento y memoria en aplicaciones para tarjetas inteligentes, usando componentes de software que posibilitan la utilización de los recursos de *hardware* de un ordenador.

Resultados y discusión

Las principales características generales del modelo de desarrollo de aplicaciones para la extensión de las capacidades de procesamiento y memoria de las tarjetas inteligentes, que fueron definidas son las siguientes:

- El modelo de desarrollo que se propone provee un mecanismo para el almacenamiento de los datos asociados a las aplicaciones de las tarjetas inteligentes, fuera de esta, en un repositorio de datos externo.
- El modelo de desarrollo que se propone provee un mecanismo para la ejecución de algoritmos de alto costo computacional, que por el tiempo de ejecución y/o complejidad es más factible realizar fuera de la tarjeta.
- El modelo de desarrollo necesita contar con un ambiente conectado y controlado como empresas e instituciones.
- El modelo de desarrollo provee mecanismos de seguridad que proporcionan un ambiente seguro fuera de la tarjeta y protegen la integridad y confidencialidad de los datos que se almacenan en el servidor externo.

El modelo de extensión que se propone integra nuevos componentes de software al modelo de desarrollo tradicional de aplicaciones para tarjetas inteligentes Java Card, que pueden ser utilizados para applets Java Card específicos, que por sus características requieran operaciones con mayor capacidad de memoria o procesamiento de la que la tarjeta es capaz de proveer. En la Figura 2 se muestra una representación del mismo.

[4] Abreviatura para Application protocol data unit

[5] Abreviatura para Personal Computer/Smart Card

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 112-126
http://rcci.uci.cu

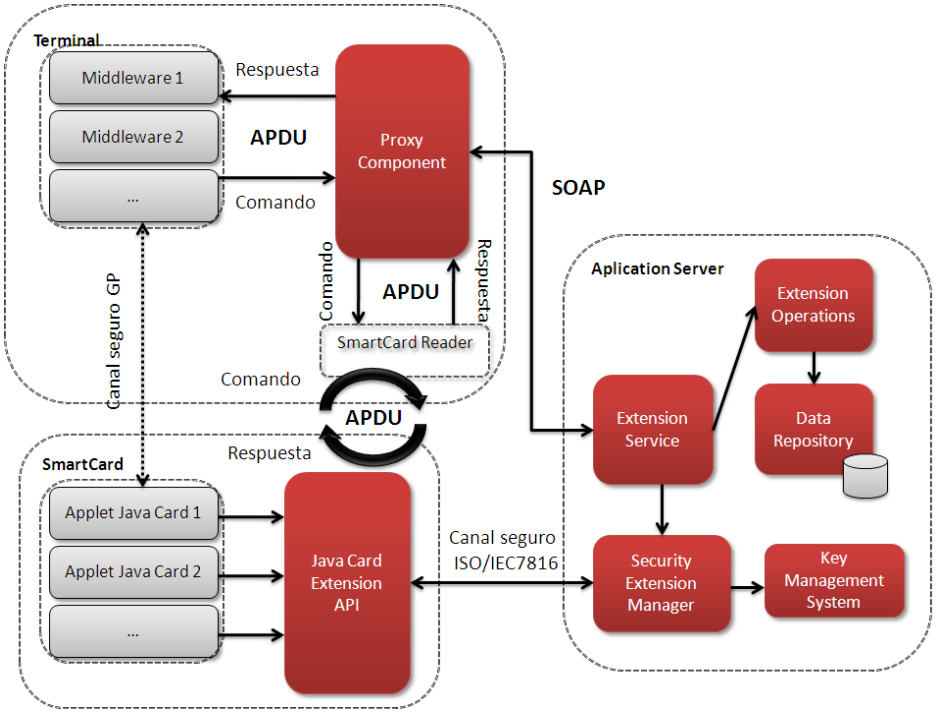

Figura 2. Modelo de extensión de capacidades de procesamiento y memoria de tarjetas inteligentes Java Card.

Los nuevos componentes de software que se proponen como parte de este modelo y sus responsabilidades se detallan a continuación:

Java Card Extension API: Este componente va instalado en la tarjeta inteligente, de conjunto con los *applets* Java Card que se desarrollen. Esta API provee un conjunto de métodos a utilizar por el desarrollador Java Card, como conformar una petición de extensión con los datos necesarios para realizarla fuera de la tarjeta, o procesar el resultado final de una operación de extensión.

Proxy Component: Este componente reside en el terminal y sirve de intermediario de la comunicación entre todos los componentes del modelo. Este componente además debe funcionar como un *wrapper* de la API PC/SC, con la idea de que cualquier *middleware* lo pueda usar para la comunicación con una tarjeta inteligente, abstrayendo al *middleware* a solo realizar operaciones necesarias para su comunicación con el *applet* o los *applets* que necesita. A su vez el Proxy Component implementa toda la lógica de un *Reader* (lector), interceptando los comandos APDU que por él pasen y con un comportamiento en cierta medida "inteligente" reenvía al *applet* en la tarjeta, al componente

Extension service en el servidor externo o a un *middleware* estos comandos APDU. Entre las principales responsabilidades que debe tener este componente se encuentran:

- Obtener los lectores disponibles para la conexión.
- Realizar la conexión y desconexión con un lector de tarjetas determinado.
- Recibir, enviar e interpretar un comando APDU.
- Enviar una petición de extensión al servicio de extensión o enviar respuesta de una operación de extensión a la tarjeta.

Extension Service: Servicio de extensión que reside en un servidor de aplicación y es el encargado de recibir las peticiones de ejecución o de almacenamiento fuera de la tarjeta inteligente. Se recomienda que sea un servicio genérico, que invoque operaciones de extensión específicas, para minimizar el desarrollo para los programadores. Entre las principales responsabilidades que debe tener este componente se encuentran:

- Ejecutar una función o algoritmo en el servidor
- Almacenar o leer información asociada a una tarjeta en un repositorio de datos.
- Enviar la respuesta de una operación realizada (procesamiento o almacenamiento).

Extension Operations (OE): Constituyen las operaciones específicas a ser ejecutadas en el servidor de aplicaciones. Estas operaciones deben ser desarrolladas y publicadas en el servidor, y pueden ser de almacenamiento o consulta de información, o la ejecución de operaciones de costo computacional o complejidad, que por el tiempo de ejecución es factible que se realicen fuera de la tarjeta.

Data Repository: Es un repositorio de datos para almacenar información asociada a un *applet* fuera de la tarjeta inteligente. En este componente se almacena información asociada a una operación de extensión de almacenamiento, tales como: el identificador de la operación y el identificador del *chip* de la tarjeta a la que se le va a asociar una información almacenada en el servidor para su posterior recuperación. También se almacenan las variables con sus respectivos valores, que se van a gestionar fuera de la tarjeta. Por lo básico de los tipos de datos que se almacenan en las tarjetas y su conversión en arreglos de *bytes* para su salida o entrada de esta, no se definen tipos de datos complejos, sino una estructura simple de clave – valor que permita almacenar variables y sus valores.

Security Extension Manager: Es el componente que se encarga de implementar la seguridad entre los nuevos componentes que introduce el modelo de desarrollo. Específicamente se encarga de establecer un canal seguro de comunicación entre el servidor de aplicaciones y la Java Card Extension API en la tarjeta, para cuando se realicen peticiones que requieran la ejecución de operaciones de extensión.

Key Management System (KMS): Componente que se encarga de la administración de manera segura de las llaves que se utilizarán para el establecimiento de los canales seguros entre los componentes del modelo.

Además de estos nuevos componentes que forman parte del modelo de extensión que se propone, se mantienen los componentes de software fundamentales de una solución para tarjetas inteligentes Java Card: los *middlewares* y sus *applets* correspondientes, que son implementados por el desarrollador de igual manera que en el modelo tradicional.

Los componentes del modelo de extensión son desplegados en tres ambientes diferentes: el terminal, la tarjeta inteligente y un servidor de aplicaciones. Al igual que en el modelo tradicional de desarrollo también, la comunicación entre un *middleware* determinado, el componente Proxy y un applet Java Card es a través de un comando APDU. Por su parte, la comunicación entre el componente Proxy y el Extensión Service ocurre a través del protocolo SOAP[6].

Flujo de comunicación entre los componentes del modelo de extensión

En la Figura 3 se presenta un flujo de comunicación básico entre cada uno de los componentes del modelo de extensión. El flujo de comunicación comienza desde el terminal, cuando un *middleware* específico envía un comando APDU a la tarjeta inteligente, que es interceptado ahora por el Proxy Component, que lo reenvía a la tarjeta. En la tarjeta un *applet* específico recibe el comando APDU y lo procesa para determinar si se está solicitando una operación que requiere ser ejecutada fuera de la tarjeta (operación de extensión) o una operación estándar. En cualquiera de los casos se responde al Proxy Component con un comando APDU respuesta.

Si el comando APDU enviado por el Proxy Component es un comando APDU de una operación estándar, se le envía un APDU respuesta con la respuesta del comando enviado. En caso contrario, si se está solicitando una operación de extensión, usando el componente Java Card Extension API, se conforma un TLV[7] con los datos necesarios para ejecutar la operación de extensión en el servidor de aplicaciones fuera de la tarjeta. En el caso de que los datos no

[6] Simple Object Access Protocol

[7] Se refiere al formato Etiqueta-longitud-valorusado para representar información.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 112-126
http://rcci.uci.cu

quepan en un comando APDU respuesta, la Java Card Extension API puede enviar cadenas APDU respuesta con todos los datos que se necesitan para ejecutar una operación de extensión (pueden ser varios parámetros para ejecutar un algoritmo costoso, por ejemplo).

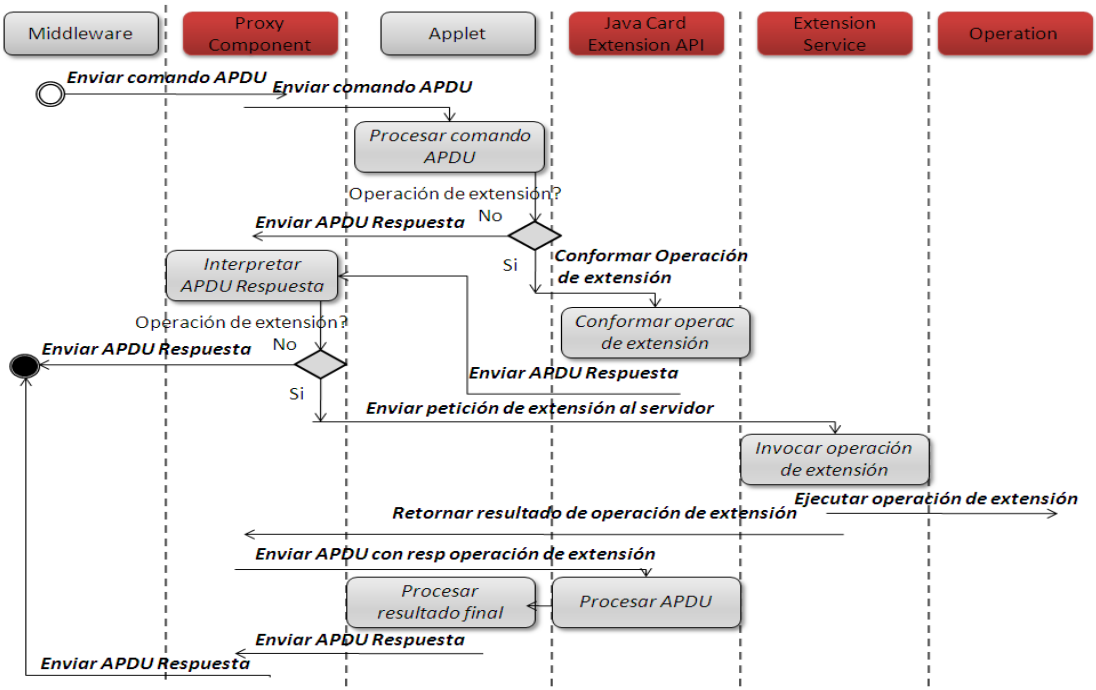

Figura 3. Flujo de comunicación entre componentes del modelo de extensión.

El comando APDU respuesta enviado por el *applet* en cualquiera de los dos casos anteriormente explicados, es interceptado por el *Proxy Componente* interpretado para saber cómo proceder. En caso de ser un comando APDU respuesta estándar se le reenvía esta respuesta al *middleware* y termina el flujo de comunicación. En el otro caso, que el APDU respuesta esté solicitando realizar una operación de extensión, se envía esta petición al *Extension Service*. Las operaciones de extensión en el servidor pueden ser de operaciones de almacenamiento o consulta de información, o la ejecución de algoritmos de alto costo computacional que por el tiempo de ejecución y/o complejidad es más factible realizar fuera de la tarjeta Como resultado del servicio web se devuelve el resultado de la ejecución de la operación en el servidor al Proxy Component. Este componente envía a la tarjeta un APDU con la respuesta de la operación de extensión recibida por el servidor, el cual se procesa por la Java Card Extension API y el *applet* al que corresponde realiza con ese resultado cualquier operación que necesite o simplemente no hace nada. Luego de este

Grupo Editorial "Ediciones Futuro"
Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

flujo de procesamiento en la tarjeta, se devuelve un APDU respuesta al Proxy Componet para que sea reenviado finalmente al *middleware* con el resultado final de la operación de extensión.

Mecanismos de seguridad propuestos

Como mecanismos de seguridad se proponen los modelos de seguridad de ISO/IEC 7816 (ISO/IEC, 2005) y Global Platform (Global Platform, 2003). Estos modelos están compuestos principalmente por un conjunto de elementos que se encuentra en el chip de la tarjeta, aunque deben ser considerados por el terminal, para poder establecer una configuración de seguridad en determinado momento, e iniciar y mantener un canal seguro de comunicación con la tarjeta inteligente. Estos elementos son: contextos y objetos de seguridad; y protocolo de autenticación mutua y de mensajería segura.

Los contextos de seguridad definen ambientes para las aplicaciones en el chip, a los cuales pertenecen varios objetos de seguridad como: conjuntos de llaves simétricas o asimétricas, configuraciones para el canal seguro de comunicación que se quiera establecer, objetos para autenticación de usuario, entre otros. Los protocolos de autenticación mutua, conforman la primera fase en la comunicación segura entre el terminal y la tarjeta, y describen cuáles son los pasos necesarios para realizar una autenticación entre ambas partes. Estos protocolos están soportados por un conjunto de comandos APDU (ISO/IEC, 2005). La mensajería segura es la segunda fase en una comunicación entre el terminal y la tarjeta, y solo puede establecerse luego de haberse realizado algún proceso de autenticación mutua satisfactoriamente.

Para la comunicación *middleware – applet*, se propone un canal seguro *Global Platform* con el protocolo SCP[8] 01, donde la autenticación mutua se realiza de forma simétrica. Una descripción detallada de este protocolo y los comandos que se intercambian en el mismo se puede encontrar en el estándar (Global Platform, 2003) y en (Jardines, 2013). Entre los componentes que se introducen como parte del modelo de desarrollo, para peticiones que requieren la ejecución de operaciones de extensión fuera de la tarjeta, se propone un canal seguro ISO/IEC 7816 entre la API *Java Card Extension* y el componente *Security Extension Management* (ambientes seguros ambos). Esta decisión está basada en el concepto de que las llaves estén almacenadas de manera segura en la tarjeta y el *Key Management System*, que se encuentra en el servidor de aplicaciones. Para enviar las llaves a la tarjeta se debe hacer durante la etapa de personalización a través de un comando *Put Data*. A diferencia de *Global Platform*, ISO no define la

[8] Abreviatura de Secure Channel Protocol

Revista Cubana de Ciencias Informáticas
Vol. 8, No. 1, Mes enero-marzo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 112-126
http://rcci.uci.cu

estructura y codificación de los comandos a utilizar para establecer un canal seguro en el estándar, solo especifica cuáles son los comandos para crear, establecer y verificar los datos que se van a necesitar en el proceso de autenticación mutua (simétrica o asimétrica). Los principales comandos que se proveen en (ISO/IEC, 2005) son: *Put Data*, *Get Challenge*, *Manage Security Environment*, *Perform Security Operation*, *Mutual Authenticate*, *Internal Authenticate y External Authenticate.*

Teniendo lo anterior como base se diseñó un protocolo de autenticación mutua simétrica basado en el modelo de seguridad ISO/IEC 7816, que se muestra a continuación en la Figura 4.

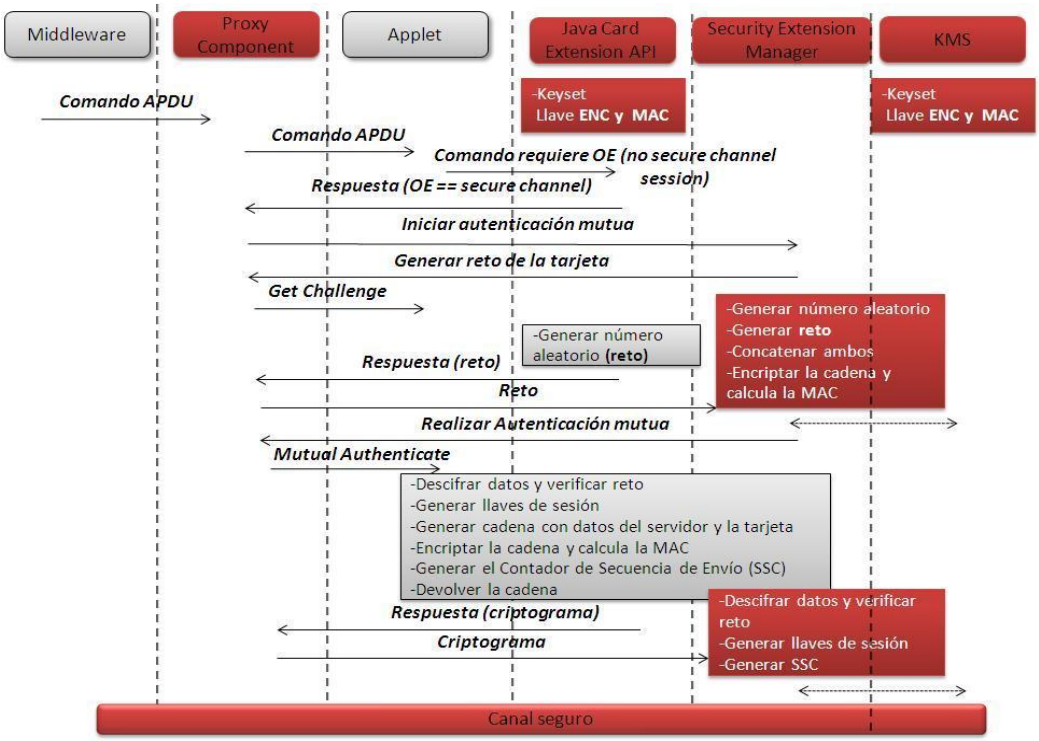

Figura 4. Protocolo de autenticación mutua ISO/IEC 7816 definido para componentes del modelo.

De igual manera que para la ejecución de las operaciones de extensión, descritas en el flujo de la Figura 3, todos la comunicación es dirigida por el *Proxy Component*. Sin embargo, en este protocolo de autenticación mutua, todo el procesamiento de información ocurre entre la *Java Card Extension* API y el componente *Security Extension Management*, además del KMS que almacena las llaves a utilizar por el servidor de aplicaciones. Luego, durante la

etapa de mensajería segura la data del comando se empaqueta en un TLV donde la etiqueta corresponde con el nivel de seguridad que se vaya a implementar según el canal de comunicación establecido entre la tarjeta inteligente y el terminal y la decisión del desarrollador.

Conclusiones

En este trabajo ha sido presentado un modelo de desarrollo de aplicaciones para la extensión de las capacidades de procesamiento y memoria de las tarjetas inteligentes, propuesto a partir de las restricciones de hardware que tienen estos dispositivos aún, y la insuficiencia detectada de proveer soluciones de software a esta problemática. El modelo de desarrollo que se propone provee un mecanismo para el almacenamiento de datos asociados a las aplicaciones de las tarjetas inteligentes fuera de esta y para la ejecución de algoritmos de alto costo computacional, que por el tiempo de ejecución y/o complejidad es más factible realizar fuera de la tarjeta. El modelo de desarrollo además provee mecanismos de seguridad que proporcionan un ambiente seguro fuera de la tarjeta y protegen la integridad y confidencialidad de los datos que se almacenan en el servidor externo. Este nuevo modelo de desarrollo puede aumentar significativamente las aplicaciones y el uso de las tarjetas inteligentes, en un ambiente conectado, principalmente para entornos controlados como empresas e instituciones.

Como aporte práctico de este trabajo se ha desarrollado una plataforma de software que automatiza el modelo propuesto y permite ejecutar operaciones de extensión (procesamiento y almacenamiento) fuera de la tarjeta inteligente, aprovechándose los recursos de hardware de los ordenadores, con tarjetas de menores prestaciones. Como trabajo futuro de la presente investigación queda la presentación de los resultados obtenidos en la aplicación de la plataforma de software desarrollada, para algunas aplicaciones de tarjetas inteligentes.

Referencias

– BLAZE, MATT. High-Bandwidth Encryption with Low-Bandwidth. In *Proceedings of the Fast Software Encryption Workshop*. Springer, 1995.
– BLAZE, MATT; FEIGENBAUM, JOAN; NAOR, Mori. A formal treatment of remotely keyed encryption. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1999. ISBN 0898714346.
– CAP, CLEMENS H; MAIBAUM, NICO y HEYDEN, LARS. Extending the data storage capabilities of a Java-based smartcard. In *Proceedings. Sixth IEEE Symposium on Computers and Communications*. IEEE, 2001. ISBN 0769511775.

– GLOBAL PLATFORM. GlobalPlatform Card Specification Version 2.1.1. 2003, no. March.

– ISO/IEC. 7816-1: Physical characteristics. 2003.

– ISO/IEC. 7816-2: Cards with contacts — Dimensions and location of the contacts. 2007.

– ISO/IEC. 7816-4: Organization, security and commands for interchange. 2005.

– JARDINES, Ander. Guía para establecer una autenticación mutua y abrir un canal seguro GlobalPlatform 2.1.1 en Tarjetas Inteligentes. In *XI Seminario Iberoamericano de Seguridad en las Tecnologías de la Información. Convención Informática*. Havana, Cuba, 2013.

– LEGÓN, ADONIS CÉSAR; RAMIREZ, SUSANA MARÍA; LANDRIÁN, JORGE; *et al.* Sistema de gestión de servicios para documentos de identificación nacional electrónicos. In *XI Seminario Iberoamericano de Seguridad en las Tecnologías de la Información. Convención Informática*. Havana, Cuba, 2013.

– LUCKS, STEFAN. Accelerated remotely keyed encryption. In *Proceedings of the Fast Software Encryption Workshop*. Berlin Heidelberg : Springer, 1999. ISBN 354066226X.

– LUCKS, STEFAN. On the security of remotely keyed encryption. In *Proceedings of the Fast Software Encryption Workshop*. Berlin Heidelberg: Springer, 1997. ISBN 3540632476.

– ORACLE; JAVA CARD FORUM. Java Card Forum. 2012. Dostupné také z. Disponible en: [http://www.javacardforum.org/openday/].

– SUN MICROSYSTEMS. Java Card Platform Version 3.0.1- Connected Edition Specification. 2009. Dostupné také z. Disponible en: [http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javame-419430.html#java_card_kit-3.0.1-rr-spec-oth-JPR].

– SUN MICROSYSTEMS. THE JAVA CARD ™ 3 PLATFORM. 2008. Dostupné také z: Disponible en: [http://www.oracle.com/technetwork/articles/javase/javacard3-whitepaper-149761.pdf].

– SUN MICROSYSTEMS. Virtual Machine Specification- Java Card™ Platform, Version 2.2.2. 2006.