UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

MODELO PARA LA EXTENSIÓN DE LAS CAPACIDADES DE PROCESAMIENTO Y MEMORIA DE APLICACIONES JAVA CARD EN TARJETAS INTELIGENTES

Tesis presentada en opción al título de Máster en Informática Aplicada

Autor: Ing. Susana María Ramírez Brey

Tutores: Dr. C. Yusnier Valle Martínez

Ms. C. Adonis Cesar Legón Campos

Ciudad de La Habana, Julio de 2014

A MIS PADRES Y ESPOSO, QUE SON LA RAZÓN DE MI VIDA...

DECLARACIÓN JURADA DE AUTORÍA Y AGRADECIMIENTOS

Yo Susana María Ramirez Brey, con carné de identidad 86100808279, declaro que soy el autor principal del resultado que expongo en el presente trabajo titulado “Modelo para la extensión de las capacidades de procesamiento y memoria de aplicaciones Java Card en tarjetas inteligentes”, para optar por el título de Máster en Informática Aplicada.

El trabajo fue desarrollado durante el período comprendido entre el 2013-2014 en colaboración del Ms.C. Adonis Cesar Legón Campos, el Dr.C. Yusnier Valle Martínez, el Ing. Amed Alfonso Ríos y la Ing. Julié Arianné Pérez Vive, quienes me reconocen la autoría principal del resultado expuesto en este trabajo.

En especial deseo agradecer a mis tutores Adonis Cesar Legón Campos y Yusnier Valle Martínez, por su ayuda infinita en mi formación como máster. A todos los profesores del claustro de la maestría y en especial al profesor Alcides Cabrera Campos por sus recomendaciones. A todos los compañeros del Centro de Identificación y Seguridad Digital que colaboraron de alguna manera en el desarrollo de esta investigación, en especial a Amed Alfonso Ríos, Julié Arianné Pérez Vive, Rafael David Portilla Santiáguez, Ander Sanchez Jardines, María Lourdes Morilla Faurés y Ramón Santana Fernández. También quiero agradecer a mis padres y esposo que siempre apoyaron mi crecimiento profesional y son el motor impulsor de todo lo que hago. A todos ellos, así como a todos los colegas y amigos que no menciono por razones de espacio, mi más sincero agradecimiento.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de la Habana a los ____ días del mes de ____ del año _____.

_____

Firma del Maestrante

Resumen

Las tarjetas inteligentes poseen características distintivas como el tamaño reducido para ser documentos portables, requerimientos de estrés y flexibilidad y la producción a gran escala que las hace sensible al coste. Asociadas a estas características se encuentran limitaciones de los recursos de *hardware* que poseen, fundamentalmente con las capacidades de memoria y procesamiento de estos dispositivos. Estas limitaciones propias y otras relacionadas con la tecnología *Java Card*, tienen un impacto negativo en el desarrollo de aplicaciones para tarjetas inteligentes, fundamentalmente en algunas áreas de aplicación críticas por los recursos que requieren. Aunque en la literatura es recurrente el tema de las limitaciones del *hardware* de estos dispositivos, no ha sido suficientemente explotada por la industria o la academia la idea de proveer soluciones de software a estas limitantes.

En este trabajo se presenta un modelo para extender las capacidades de procesamiento y memoria de las aplicaciones *Java Card* en tarjetas inteligentes, haciendo uso de forma segura de los recursos de *hardware* de un ordenador. El modelo que se propone provee un mecanismo para el almacenamiento de datos asociados a las aplicaciones fuera de la tarjeta inteligente y para la ejecución de algoritmos de alto costo computacional, que por el tiempo de ejecución y/o complejidad sean más factible realizar fuera de la tarjeta.

Como validación práctica del modelo de extensión se desarrolló una plataforma que permite la implementación y ejecución, de manera simple y con poco esfuerzo, de operaciones de extensión para aplicaciones que requieren recursos que la tarjeta no es capaz de proveer. Se realizaron implementaciones de algunas operaciones de extensión específicas en diferentes escenarios, que permitieron demostrar la factibilidad del modelo de extensión propuesto y además sirven de guía para el desarrollo de otras aplicaciones de este tipo con el uso de la plataforma.

Palabras clave: extensión, *hardware*, *Java Card*, limitaciones, memoria, operaciones de extensión, plataforma de desarrollo, procesamiento, tarjetas inteligentes.

Índice general

INTRODUCCIÓN **1**

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS -METODOLÓGICOS **7**

1.1. Introducción a las tarjetas inteligentes 7
1.2. Características físicas de las tarjetas inteligentes 9
1.3. Requerimientos de procesamiento y memoria para diferentes áreas de aplicación de las tarjetas inteligentes 12
1.4. Especificaciones y estándares para tarjetas inteligentes 15
1.5. Seguridad en tarjetas inteligentes 22
1.6. Soluciones de software que combinan las tarjetas inteligentes con ordenadores para lograr mayor capacidad de cómputo y almacenamiento 30
Conclusiones del Capítulo 34

CAPÍTULO 2. MODELO PARA LA EXTENSIÓN DE LAS CAPACIDADES DE PROCESAMIENTO Y MEMORIA DE APLICACIONES *JAVA CARD* EN TARJETAS INTELIGENTES **36**

2.1. Modelo tradicional de desarrollo de aplicaciones *Java Card* 36
2.2. Principios del modelo para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes 38
2.3. Componentes del modelo de extensión 39
2.4. Flujo de comunicación entre los componentes del modelo de extensión 45
2.5. Mecanismos de seguridad del modelo de extensión 48
Conclusiones del capítulo 54

CAPÍTULO 3. PLATAFORMA DE DESARROLLO Y EJECUCIÓN DE OPERACIONES DE EXTENSIÓN PARA APLICACIONES *JAVA CARD* EN TARJETAS INTELIGENTES **56**

3.1. Características generales de la Plataforma 56
3.2. Ambiente de desarrollo 57
3.3. Arquitectura de software de la Plataforma 58
3.4. Diagrama de Componentes de la Plataforma 60
3.5. Despliegue de la Plataforma 62
3.6. Pruebas de software realizadas a la Plataforma 64
3.7. Implementación de operaciones de extensión sobre la Plataforma 65
Conclusiones del capítulo 69

CONCLUSIONES **71**

RECOMENDACIONES **72**

REFERENCIAS BIBLIOGRÁFICAS **73**

Índice de figuras

Figura 1. Clasificación de las tarjetas inteligentes acorde al tipo de chip y el método de transmisión (Traducida de (Rankl et al., 2010)) 8
Figura 2. Capacidades de almacenamiento y procesamiento aritmético requerido para algunas áreas de aplicación de las tarjetas inteligentes. (Traducida de (Rankl et al., 2010)) 15
Figura 3. Arquitectura de Java Card (Traducida de(Ortiz, 2003)) 20
Figura 4. Dominios de seguridad y aplicaciones en una tarjeta inteligente GlobalPlatform. 24
Figura 5. Flujo de autenticación mutua GlobalPlatform del SCP01 (Terminal – Dominio de seguridad) 25
Figura 6. Arquitectura de entornos de seguridad por ficheros de ISO/IEC 7816 28
Figura 7. Arquitectura de la Smart Card Extension (Cap et al., 2001) 34
Figura 8. Modelo tradicional de desarrollo de aplicaciones Java Card para tarjetas inteligentes ..37
Figura 9. Flujo de comunicación del modelo tradicional de desarrollo de aplicaciones Java Card para tarjetas inteligentes.....37
Figura 10. Vista general del modelo de extensión. 38
Figura 11. Componentes del modelo de extensión. 40
Figura 12. Flujo de comunicación entre los componentes del modelo de extensión. 46
Figura 13. Flujo de autenticación mutua GlobalPlatform del SCP01 (middleware – applet) 49
Figura 14. Flujo de autenticación mutua simétrica ISO/IEC 7816 (API Java Card de Extensión – Administrador de seguridad) 52
Figura 15. Arquitectura de la Plataforma de desarrollo para la extensión de las capacidades de procesamiento y memoria de aplicaciones Java Card en Tarjetas Inteligentes 59
Figura 16. Diagrama de componentes del Java Card Applet Extension Client 60
Figura 17. Diagrama de componentes del Java Card Applet Extension Proxy 61
Figura 18. Diagrama de componentes del Java Card Applet Extension Server 61
Figura 19. Diagrama de despliegue de la Plataforma 63

Índice de tablas

Tabla 1. Configuración típica de hardware de las tarjetas inteligentes(Sun Microsystems, 2008).11
Tabla 2. Volumen de información que requieren diferentes perfiles de aplicaciones para tarjetas inteligentes (Bobineau et al., 2001). 14
Tabla 3. Estructura de un comando APDU.....17
Tabla 4. Campos del comando APDU 17
Tabla 5. Formato de respuesta a los comandos APDU 17
Tabla 6. Campos de la respuesta APDU.....18
Tabla 7. Características soportadas y no soportadas de Java. 21
Tabla 8. Estructura TLV para petición de una operación de extensión 42
Tabla 9. Detalles de la estructura TLV para petición de una operación de extensión 42
Tabla 10. Descripción de la autenticación mutua simétrica GlobalPlatform middleware-applet 50
Tabla 11.Descripción de la autenticación mutua simétrica ISO/IEC 7816 tarjeta-servidor.....53
Tabla 12. Resultados obtenidos en el cálculo del factorial.....66
Tabla 13. Resultados obtenidos en la generación de llaves ECDH y RSA.....68
Tabla 14. Extensión de las capacidades de procesamiento y memoria de las tarjetas Gemplus GemXpresso Pro R3.2 E32 PK 69

Introducción

Las tarjetas inteligentes o *smart cards*, surgidas en Europa en la década de 1970, han contribuido al desarrollo tecnológico y al aumento de la seguridad en sistemas informáticos. La tecnología de las tarjetas inteligentes es un estándar de la industria definido y controlado por el Comité Técnico de la Organización Internacional de Normalización[1] (ISO) y la Comisión Electrotécnica Internacional[2] (IEC). La colección de estándares ISO/IEC 7816 define varios aspectos de las tarjetas inteligentes, entre los que se encuentran: características físicas, protocolos de transmisión, comandos, arquitectura de seguridad, etc.

El auge que han tenido las tarjetas inteligentes se debe en gran medida a la seguridad tanto física como lógica que brindan estas tecnologías. 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, son protegidos por una combinación de *hardware* y software. Debido fundamentalmente a esta razón 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 muchas. Como una tendencia se ha observado en los últimos años que varios países han comenzado a modernizar su documento de identificación nacional hasta convertirlo en un eID[3] 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 (Legón et al., 2013).

Asociadas a algunas características físicas de las tarjetas inteligentes definidas en (ISO/IEC, 2007, 2011a) como el tamaño reducido para ser documentos portables e incrementar la seguridad del *hardware* o los requerimientos en cuanto a estrés y flexibilidad, 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 CPUs[4] de 8/16 bits (arquitectura del

[1] Traducido de las siglas en inglés ISO: *International Standards Organization.*

[2] Traducido de las siglas en inglés IEC: *International Electronic Committee.*

[3] Documento de identificación electrónico (traducido de las siglas en inglés eID: *Electronic identification*).

[4] Unidad central de procesamiento (traducido de las siglas en inglés CPU: C*entral processing unit*).

procesador), ~2 kB de RAM[5] (para la pila de ejecución, variables temporales y objetos transitorios), de 48 kB – 64 kB de ROM[6] (para el sistema operativo de la tarjeta, la máquina virtual y las aplicaciones preinstaladas), y de 8 kB – 32 kB aproximadamente de EEPROM[7] (para el código de las aplicaciones dentro de la tarjeta, objetos persistentes y variables de clases). 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 (Ramírez, Legón, & Valle, 2014).

A finales de 1996 la empresa Sun Microsystems acordó diseñar una versión de su plataforma *Java* para tarjetas inteligentes. La tecnología *Java Card*, 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, según datos publicados en (Oracle & Java Card Forum, 2012). *Java Card* fue diseñado 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, el trabajo con hilos, el clonado de objetos, etc. Además no se soportan los tipos de datos *char, double, float, long* y los arreglos multidimensionales, el tipo de dato *int* incluso es opcional para algunas tarjetas. Existen un gran número de proyectos *Java* Card que a lo largo de los años han demostrado las limitaciones del lenguaje y sus consecuencias, en (Cap, Maibaum, & Heyden, 2001) se describen y referencian algunos de estos proyectos. Las 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.

Como posibles soluciones a las limitaciones antes mencionadas, la industria ha estado revisando temas de diseño que permiten incorporar múltiples *chips* de memoria en una tarjeta inteligente. Empresas como Gemalto (Gemalto, 2013) han producido tarjetas gemelas con la incorporación de dos *chips* no conectados en una sola tarjeta. De igual manera se han ido desarrollando algunos dispositivos con una configuración de *hardware* superior. A pesar de los avances que significa esta nueva generación de

---

[5] Memoria de acceso aleatorio (traducido de las siglas en inglés RAM: *Random-access memory*).

[6] Memoria de solo lectura (traducido de las siglas en inglés ROM: *Read-only memory).*

[7] Memoria de solo lectura programable y borrable eléctricamente (traducido de las siglas en inglés EEPROM: *Electrically Erasable Programmable Read-Only Memory*).

tarjetas inteligentes, el mercado aún es dominado por las generaciones anteriores, debido fundamentalmente a los costos de producción que implican los dispositivos con estas características.

Otro de los enfoques que se podría utilizar para contrarrestar las limitantes del *hardware* de las tarjetas inteligentes, es la utilización de soluciones de software que permitan explotar la capacidad computacional y de almacenamiento de los ordenadores, de conjunto con el de las tarjetas. En este enfoque se debe prestar especial atención a la seguridad, debido a que los datos de las aplicaciones que maneja la tarjeta inteligente pueden verse comprometidos al salir fuera de esta, de manera insegura. En la revisión de la literatura existen pocas soluciones de software que explotan este enfoque totalmente. Esto se considera que se debe en gran medida a que esta tecnología es controlada y desarrollada fundamentalmente por la industria y empresas productoras de tarjetas inteligentes (Ramírez et al., 2014).

En el Centro de Identificación y Seguridad Digital (CISED) se desarrollan aplicaciones para tarjetas inteligentes, fundamentalmente en las áreas de documentos de identificación electrónicos, documentos electrónicos de viaje y licencias de conducción electrónicas. Asociadas a estas soluciones también se desarrollan aplicaciones para la creación y almacenamiento de llaves y certificados digitales, verificación biométrica en la tarjeta, aplicaciones PKI[8], entre otras. Los especialistas que han formado parte de estas investigaciones y desarrollos a lo largo de los años han experimentado las limitaciones descritas con anterioridad, debido a que han contado con tarjetas inteligentes de escasos recursos de *hardware*. Estas tarjetas dificultan el desarrollo de diversas aplicaciones, como por ejemplo las de PKI que requieren tamaños de llaves RSA[9] mayores a 2048 bits, o contar con capacidades criptográficas para algoritmos como Curvas Elípticas. Debido a estos elementos se considera que existen aplicaciones para tarjetas inteligentes que pueden mejorar significativamente su desempeño y efectividad con el uso de la memoria y la capacidad de procesamiento externa de un ordenador.

De todo lo anteriormente expuesto se deriva la necesidad e importancia que reviste una solución que permita a los desarrolladores usar los recursos de *hardware* de un

[8] Infraestructura de clave pública (traducido de las siglas en inglés PKI: *Public Key Infraestructure*)

[9] De las siglas Rivest - Shamir – Adleman, creadores del algoritmo (Ron Rivest, Adi Shamir y Leonard Adleman).

ordenador, de conjunto con una tarjeta inteligente, para extender las capacidades de procesamiento y almacenamiento de estas.

Por tanto se define el siguiente **problema de investigación:** Las restricciones de procesamiento y memoria de las tarjetas inteligentes limitan el desarrollo de aplicaciones *Java Card* y, por tanto, su uso extensivo en diversas áreas de aplicación.

A partir del problema de investigación planteado, se identifica como **objeto de estudio** el proceso de desarrollo de aplicaciones *Java Card* para tarjetas inteligentes.

Para dar solución al problema de investigación, se precisó como **objetivo general** definir un modelo para extender las capacidades de procesamiento y memoria de las aplicaciones *Java Card* en tarjetas inteligentes, mediante el uso de forma segura de los recursos de *hardware* de un ordenador.

La investigación se conduce teniendo como **hipótesis**: La definición de un modelo que haga uso de forma segura de los recursos de *hardware* de un ordenador para extender las capacidades de procesamiento y memoria de las tarjetas inteligentes, solventará las limitaciones existentes en el desarrollo de aplicaciones *Java Card* para estos dispositivos.

El objetivo general de la investigación se deriva en los siguientes **objetivos específicos**:

1. Establecer los fundamentos teóricos-metodológicos relacionados con el desarrollo de aplicaciones *Java Card* para tarjetas inteligentes, con énfasis en las limitantes que posee.
2. Caracterizar los estándares, tecnologías y mecanismos de seguridad que formarán parte del modelo.
3. Definir el esquema general del modelo para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes
4. Implementar los componentes de una plataforma de software que permita el desarrollo y ejecución de operaciones fuera de la tarjeta inteligente, como aplicación del modelo de extensión propuesto.
5. Realizar una valoración de la efectividad del modelo de extensión propuesto, a partir de la implementación de operaciones sobre la plataforma, para diferentes escenarios.

Métodos de investigación:

El método 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.

El análisis histórico – lógico se tuvo en cuenta en la elaboración del estado del arte de la investigación, analizando el proceso de desarrollo de aplicaciones para tarjetas inteligentes y las limitaciones del *hardware* de las mismas, desde su perspectiva histórica.

La modelación fue utilizada para crear abstracciones con vista a explicar la realidad. Este método se tuvo en cuenta para modelar tanto la solución de referencia que se propone como parte de esta investigación, como la plataforma de software que implementa y permite validar la misma.

El experimento se utilizó en el diseño y ejecución de las pruebas que se realizaron a la plataforma de software, para constatar la hipótesis de la investigación.

Aporte y novedad de la investigación:

La contribución principal del presente trabajo consiste en la propuesta de un modelo para extender las capacidades de procesamiento y memoria de las tarjetas inteligentes con tecnología *Java Card*. Este modelo de extensión puede aumentar significativamente las aplicaciones y el uso de las tarjetas inteligentes, en ambientes conectados y controlados, donde sea posible aprovechar los recursos de *hardware* de los ordenadores, con tarjetas inteligentes *Java Card* de bajas prestaciones.

Como resultado complementario se proporciona una plataforma de software que permite el desarrollo y ejecución de operaciones fuera de la tarjeta inteligente, según el modelo de extensión propuesto.

Lo novedoso de la investigación está reflejado en la utilización de soluciones de software que permitan explotar la capacidad computacional y de almacenamiento de los ordenadores, de conjunto con el de las tarjetas inteligentes. Para Cuba representa un avance en el desarrollo de soluciones para esta tecnología, que aunque es conocida y ampliamente utilizada a nivel mundial, en el país las investigaciones en este campo son incipientes.

Estructura del documento: el documento se encuentra dividido en tres capítulos como se describe a continuación:

En el **Capítulo 1** se introducen conceptos relacionados con las tarjetas inteligentes y sus características físicas. Se describen algunas de las especificaciones y estándares de mayor importancia en el desarrollo de aplicaciones para tarjetas inteligentes, además de los principales modelos de seguridad para estas. También se exponen trabajos donde se evidencian las limitaciones del *hardware* de esta tecnología para algunas áreas de aplicación, y se realiza un análisis de soluciones de software que permiten extender las capacidades de procesamiento y memoria de las tarjetas inteligentes, con el uso de los ordenadores.

En el **Capítulo 2** se hace la propuesta de un modelo para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes. Se realiza la conceptualización del modelo a partir de tecnologías y estándares existentes para el desarrollo de aplicaciones *Java Card* para tarjetas inteligentes. Se describen las premisas del modelo, sus componentes principales, el flujo de comunicación entre ellos y los mecanismos de seguridad que se proponen para garantizar un ambiente de ejecución seguro.

En el **Capítulo 3** se presenta una plataforma de software que permite el desarrollo y ejecución de operaciones fuera de la tarjeta inteligente, según el modelo de extensión propuesto. Se describen las tecnologías, *frameworks* y herramientas utilizadas para la construcción de la plataforma en sus diferentes ambientes. Se exponen también los aspectos fundamentales de la implementación y funcionamiento de la plataforma. Finalmente se presentan los resultados obtenidos en la implementación de operaciones sobre la plataforma, para constatar la hipótesis de la investigación.

Además se presentan las Conclusiones, Recomendaciones y Referencias Bibliográficas de este trabajo.

Capítulo 1. Fundamentos teóricos -metodológicos

En este capítulo se introducen los principales conceptos relacionados con las tarjetas inteligentes. Además se describen sus características físicas, prestando especial atención a las capacidades de procesamiento y memoria que estas poseen, con limitaciones para algunas de sus áreas de aplicación. Se realiza un análisis de algunas soluciones de software que combinan el uso de las tarjetas inteligentes con ordenadores, para lograr mayor capacidad de cómputo y almacenamiento. Asimismo se caracterizan los principales estándares, especificaciones y modelos de seguridad que sirven de base al desarrollo de aplicaciones para tarjetas inteligentes.

1.1. Introducción a las tarjetas inteligentes

Desde la década de 1950 comenzó a registrarse una tendencia al uso de tarjetas plásticas para la identificación y el pago de algunos servicios. Estas tarjetas plásticas fueron evolucionando hasta la incorporación de la banda magnética y el PIN[10]. La tecnología de banda magnética poseía una debilidad crucial que residía en la posibilidad de que los datos almacenados en la banda magnética podían ser leídos, eliminados o reescritos por cualquier persona con el equipamiento necesario.

Las primeras llamadas tarjetas inteligentes usadas a gran escala fueron las tarjetas de memorias(también llamadas *memory cards*) que contaban con un *chip* para el almacenamiento de información. El área de aplicación fundamental de estas tarjetas en sus inicios, fue en las aplicaciones telefónicas, ya que se producían como tarjetas pre-pagadas con el valor almacenado electrónicamente en el *chip*, que iba siendo reducido por la cantidad del cargo de la llamada cada vez que la tarjeta era usada. De esta manera se podía prevenir el incremento posterior del valor almacenado, que podía ser fácilmente realizado con las tarjetas de banda magnética. Este tipo de manipulación conocida como *buffering*, era prevenida en las tarjetas de teléfono inteligentes por la lógica de seguridad en el *chip* que hacía imposible borrar la celda de memoria una vez que esta había sido escrita (Rankl, Effing, & Cox, 2010) .

No fue hasta la década de 1970 que el enorme progreso en la microelectrónica posibilitó integrar el almacenamiento de datos y el procesamiento lógico en un solo *chip* de silicio con una medida de algunos milímetros cuadrados. A partir de la idea de incorporar este

[10] Número de identificación personal (traducido de las siglas en inglés PIN: *Personal identification number*).

circuito integrado en una tarjeta de identificación surgen las tarjetas inteligentes o *smart cards* (Rankl et al., 2010) que se conocen en la actualidad. Este *chip* integrado con un microprocesador permite a la tarjeta inteligente almacenar, acceder a datos y aplicaciones, además de intercambiar datos de manera segura con lectores y otros sistemas. Las tarjetas inteligentes más modernas pueden tener incluido un co-procesador criptográfico el cual permite una mayor potencia de cómputo, necesario para operaciones de cifrado. Estas tarjetas por supuesto son mucho más funcionales y por tanto costosas. La posibilidad de almacenar llaves secretas de manera segura y ejecutar algoritmos criptográficos en este tipo de tarjetas, hace que sea posible implementar, por ejemplo, sistemas de pago offline altamente seguros.

Además de la clasificación según el tipo de *chip* (tarjetas con *chip* de memoria y con microprocesador), las tarjetas inteligentes se clasifican teniendo en cuenta otros criterios, como la interfaz que utilizan para la transmisión de datos, que puede ser por contacto, sin contacto y las tarjetas duales (combinan las dos interfaces de comunicación). En la Figura 1 se muestra un esquema que resume las principales clasificaciones de las tarjetas inteligentes antes descritas.

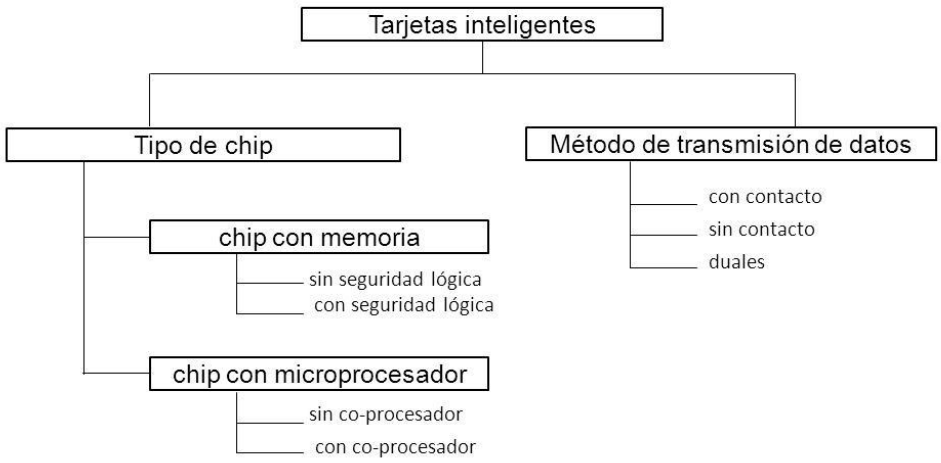

Figura 1. Clasificación de las tarjetas inteligentes acorde al tipo de *chip* y el método de transmisión (Traducida de (Rankl et al., 2010))

Las tarjetas inteligentes por contacto necesitan ser introducidas en un lector de tarjetas para establecer la comunicación con el *chip* usando los contactos eléctricos. Por el contrario las tarjetas inteligentes sin contacto emplean la radiofrecuencia (RFID[11]) para

[11] Identificación por radio frecuencia (traducido de las siglas en inglés RFID: *Radio Frequency Identification*).

la comunicación con el lector y no necesitan la inserción física de la tarjeta en el mismo (CardLogix Corporation, 2009).

De manera general la tecnología de las tarjetas inteligentes puede proveer altos niveles de seguridad y protección de la información, y esto es debido a las características que se resumen a continuación:

- Soportan una autenticación multi-factor al combinar tres factores: algo que se posee (la propia tarjeta inteligente), con algo que se es (los datos biométricos) y algo que se conoce (el PIN).

- Soportan el uso de firma digital que puede ser utilizada, por ejemplo, para determinar que la tarjeta fue expedida por una organización válida.

- Usan tecnologías de seguridad en el *chip*, y son diseñadas y personalizadas con características físicas de seguridad que impiden su falsificación.

- Pueden poseer co-procesadores criptográficos que realicen complejas operaciones aritméticas.

- Pueden controlar quién accede a la información almacenada en las aplicaciones.

Todos estos factores han propiciado que esta tecnología sea usada mundialmente cuando la seguridad y la privacidad de la información son requerimientos críticos. También el uso de las tarjetas inteligentes con teléfonos móviles ha sido de especial importancia para su proliferación internacional. Otras potenciales aplicaciones incluyen las tarjetas de identificación, sistemas de control de acceso para áreas restringidas y ordenadores, almacenamiento seguro de información, firma digital, monederos electrónicos y tarjetas multifuncionales que incorporan varias aplicaciones en una tarjeta (Rankl et al., 2010).

En resumen, las ventajas de las tarjetas inteligentes están dadas por la capacidad de almacenar información, datos confidenciales de manera segura y la habilidad de ejecutar algoritmos criptográficos. Estas ventajas hacen que sea posible desarrollar un amplio rango de nuevas aplicaciones, siendo limitadas únicamente por la memoria disponible en la tarjeta inteligente y el poder de cómputo del procesador que posee.

1.2. Características físicas de las tarjetas inteligentes

Las tarjetas inteligentes se ajustan a normas internacionales como la colección de estándares ISO/IEC 7816 e ISO/IEC 14443 (ISO/IEC, 2008a, 2008b, 2010, 2011b) para tarjetas sin contacto; y además están disponible en una variedad de formas incluidas las tarjetas de plástico, tarjetas SIM[12] utilizados en teléfonos móviles GSM[13] y tokens USB (Smart Card Alliance, 2013a).

Las tarjetas inteligentes con microprocesador son tarjetas que tienen una estructura análoga a la de un ordenador. Sin embargo, su procesador tiene poco parecido al que se puede encontrar en un ordenador moderno, aunque el núcleo del *chip* si guarda cierta relación con el de las generaciones anteriores de ordenadores. Los *chips* de las tarjetas inteligentes tienden a ser muy pequeños y por esto siempre existe un límite en las funcionalidades y los recursos que estos pueden manejar. Existen varias razones para las limitaciones del tamaño del *chip*, entre ellas se encuentran: el costo del *chip* que es proporcional al área de silicio usado y aunque de forma individual puede no ser mucho la producción a gran escala las hace sensible al coste; los requerimientos de las tarjetas inteligentes en cuanto a estrés y flexibilidad, que no son posibles alcanzar con *chips* muy grandes; y por último, que cuando el *chip* se hace mayor y más complejo, los requisitos de energía para su alimentación también crecen (Mayes & Markantonakis, 2008).

El procesador, que es el corazón del *chip* en la tarjeta inteligente, usualmente está rodeado de cuatro bloques funcionales adicionales: la máscara ROM, la EEPROM, la RAM y al menos un puerto I/O[14]. La memoria de solo lectura (ROM) contiene el sistema operativo y se conoce como la máscara de la tarjeta; su contenido solo puede ser grabado durante el proceso de fabricación. La memoria de acceso aleatorio (RAM) que se utiliza para el almacenamiento dinámico de variables de programas en tiempo de ejecución y la pila de ejecución. En este sentido es similar a la RAM de un ordenador aunque en mucha menor cantidad. De igual manera cuando se desconecta la alimentación de la memoria RAM se pierde su contenido. La memoria EEPROM que es muy útil ya que puede programarse después de la fabricación, es una memoria de lectura/escritura para el almacenamiento de datos y es no volátil[15]. La EEPROM puede ser utilizada para almacenar datos de usuarios y aplicaciones, así como los datos y

---

[12] Módulo de identificación del abonado (traducido de las siglas en inglés SIM: *Subscriber identity module*).

[13] Sistema global para las comunicaciones móviles (traducido de las siglas en inglés GSM: *Global System for Mobile communications*).

[14] Entrada/salida (traducido de las siglas en inglés: *input/output*).

[15] Se refiere a la característica de que al quedar sin energía, la memoria pierda su contenido.

funciones que requieren modificación durante la vida operativa de una aplicación. También se conoce un tipo de EEPROM llamada *Flash memory* o *Flash* EEPROM que comparte la característica de ser no volátil, pero permite la programación en bloques más grandes que la EEPROM normal, por lo que los tiempos de acceso para la lectura y escritura son también menores (Rankl et al., 2010).

Existen limitaciones en los recursos de *hardware* de las tarjetas inteligentes y son varios los trabajos que hacen referencias a ellas, como (Anciaux, Bobineau, Bouganim, Pucheral, & Valduriez, 2001; Blaze, 1996; Cap et al., 2001; Hunt & Holcombe, 2004; Márquez, 2006; Mayes & Markantonakis, 2008; Mohammed, Rahman, Prakash, & Daud, 2004; Rankl et al., 2010; Smart Card Alliance, 2012). Esto da una idea tanto del interés en el tema, como de la necesidad que existe de soluciones que brinden una alternativa a estas limitaciones. En la Tabla 1 se muestra un resumen de las principales características que posee el *hardware* de las tarjetas inteligentes tradicionales y el de algunas tarjetas inteligentes con características superiores.

Tabla 1. Configuración típica de *hardware* de las tarjetas inteligentes(Sun Microsystems, 2008)

***Hardware* tradicional de tarjetas inteligentes**	**Gama alta de tarjetas inteligentes**
CPU de 8/16 bits	CPU de 32 bits
~2 kB RAM	24 kB RAM
48 kB - 64 kB ROM	>256 kB ROM
8 kB - 32 kB EEPROM	>128 kB EEPROM
Interfaces I/O en Serie	Interfaces de altas velocidades
9,6 kB /s – 30 kB /s	1,5 MB/s – 12 MB/s
Full duplex	Half Duplex

Como es posible apreciar, la industria ha estado desarrollando tarjetas inteligentes con mayores prestaciones que las tarjetas tradicionales, que poseen mayor RAM, ROM, EEPROM e incluso procesadores de 32 bits. A pesar de esto, la RAM y EEPROM se tratan de mantener tan pequeñas como sea posible, debido a que requieren el mayor espacio por bit (Rankl et al., 2010). La adquisición de estos dispositivos de gama alta en la actualidad es costosa y la tecnología se sigue perfeccionando de manera que se puedan reducir los costos de fabricación. De momento todas las aplicaciones que requieren grandes cantidades de memoria (sobre todo memoria EEPROM) y de procesamiento de las tarjetas inteligentes son poco prácticas. En general, también son

inapropiadas para el procesamiento complejo de datos numéricos o procesamiento de datos en bloque, con la excepción de funciones criptográficas, donde el *chip* puede tener un co-procesador especializado para la ejecución rápida de este tipo de funciones.

1.3. Requerimientos de procesamiento y memoria para diferentes áreas de aplicación de las tarjetas inteligentes

Las tarjetas inteligentes son usadas hoy en día en una gran variedad de industrias en todo el mundo, para soportar aplicaciones de control de acceso, programas de lealtad, identidad y firma digital, sistemas prepago, asistencia médica, transacciones seguras y otras muchas (CardLogix Corporation, 2009; Smart Card Alliance, 2013b). Desde el mismo comienzo de su desarrollo las tarjetas inteligentes proveían un medio ideal para la criptografía. Podían almacenar de forma segura material criptográfico, claves de cifrado, certificados digitales, contraseñas, e incluso patrones biométricos (Noore, 2000; Sanchez, Mengibar, & Sanchez, 2003); al mismo tiempo estaban capacitadas para ejecutar algoritmos criptográficos cada vez más potentes. Por otro lado destacaban por su forma, tamaño y facilidad de manejo, de tal manera que podían ser empleadas en una variedad de contextos. Estas fueron las principales motivaciones que llevaron a los bancos a desarrollar soluciones de seguridad para su empleo en transacciones monetarias, impulsando así el desarrollo de esta tecnología en la década de 1980.

Dentro de las aplicaciones que hacen un mayor uso de las tarjetas inteligentes se encuentran las telecomunicaciones con la telefonía celular, donde se integra al teléfono móvil el llamado módulo de identificación del abonado (SIM en GSM/GPRS o USIM[16] en UMTS[17]) o también conocido como *SIM Cards* (CardLogix Corporation, 2009). Dichas tarjetas (U)SIM almacenan la información personal del abonado, credenciales de seguridad y las preferencias que pueden ser protegidas por una clave. En este contexto, las *SIM Cards* permiten: el acceso autorizado y autenticado del abonado, políticas particulares de facturación, resolución de políticas de *roaming*[18] entre redes de distintos operadores, así como el acceso seguro a una variedad de servicios móviles emergentes

[16] Módulo de identificación del abonado universal (traducido de las siglas en inglés USIM: *Universal subscriber identity module*).

[17] Sistema universal de telecomunicaciones móviles (traducido de las siglas en inglés UMTS*: Universal Mobile Telecommunications System*).

[18] Concepto de comunicaciones inalámbricas relacionado con la capacidad de un dispositivo para moverse de una zona de cobertura a otra.

(comercio electrónico móvil, navegación web, servicios de mensajes cortos, etc.) (Márquez, 2006).

Además, en el plano institucional la tarjeta inteligente como elemento de identificación segura está cada vez más presente. Los negocios y universidades utilizan tarjetas de este tipo para la identificación de empleados y estudiantes, permitiéndoles el acceso a ciertos datos, equipamientos y departamentos según su estatus. En la pasada década también se puso en marcha el DNI[19] y el pasaporte electrónico en varios países de Europa como Irlanda (Rouine & Murphy, 2005), Alemania (Friedrich & Seidel, 2006) y Holanda (Kleef, 2006). Estos eID usualmente combinan varias aplicaciones como licencia de conducción electrónica, aplicaciones de salud y otras no gubernamentales como monedero electrónico, entre otras (Legón et al., 2013).

No todas las áreas de aplicaciones de las tarjetas inteligentes hoy en día poseen los mismos requerimientos de procesamiento y memoria. Desde hace algunos años se vienen discriminando las aplicaciones por sus requerimientos. Tal es el caso temprano de (Bobineau, Bouganim, Pucheral, & Valduriez, 2001) donde se agrupan un conjunto de aplicaciones con fines similares en perfiles, clasificando el volumen de datos que manejan cada una de ellas:

- Dinero e Identificación: Agrupa aplicaciones tales como tarjetas de créditos, monederos electrónicos, tarjetas SIM para GSM, tarjetas de transporte, entre otras. Estas son aplicaciones que de manera general manejan muy pocos datos, principalmente el identificador del portador y alguna información específica.

- Bases de datos descargables: Son paquetes de datos predefinidos (listas de restaurantes, hoteles, sitios turísticos, catálogos, etc.) que pueden ser descargados a la tarjeta antes de un viaje y pueden ser accedidos luego desde cualquier terminal. El volumen de datos que manejan este tipo de aplicaciones puede llegar a ser importante.

- Ambientes de usuario: El objetivo de estas aplicaciones es almacenar en la tarjeta un perfil extendido del portador de la misma, incluyendo entre otros, los datos relativos a su ambiente informático (configuración del ordenador, contraseñas, cookies, marcadores, licencias de software, etc.), o una libreta de

[19] Documento nacional de identificación.

direcciones como una agenda. El volumen de información puede llegar a ser tan extenso como se desee.

- Información personal: La información personal que se almacene puede ser de diferentes naturalezas: académica, de atención médica, un historial del mantenimiento del carro, etc. Estas aplicaciones tienen fuertes requerimientos de almacenamiento de información.

En la Tabla 2 se resume el volumen de información necesario a manejar por estos perfiles de aplicaciones de tarjetas inteligentes, usando un criterio cualitativo. Como es evidente, las aplicaciones que se encuentran en los perfiles de información personal y bases de datos descargables son las que requieren de un volumen mayor de información.

Tabla 2. Volumen de información que requieren diferentes perfiles de aplicaciones para tarjetas inteligentes (Bobineau et al., 2001).

Aplicaciones	**Volumen**
Dinero & Identificación	Pequeño
Bases de datos descargables	Grande
Ambiente de usuario	Medio
Información personal	Grande

Además de los diferentes requerimientos de almacenamiento y manejo de información, también existen diferentes necesidades en cuanto al procesamiento aritmético en las principales áreas de aplicación de las tarjetas inteligentes existentes hoy en día. En la Figura 2 se muestran estas áreas de aplicación prestando particular atención a la necesidad de memoria y de procesamiento que requieren cada una de ellas. Dentro de las aplicaciones con mayores requerimientos de procesamiento se encuentran las de cifrado de datos (sobre todo en cifrado de archivos, tráfico en tiempo real, multimedia y video), donde la velocidad del cifrado está atada a la latencia, la velocidad de la interfaz de la tarjeta y la capacidad computacional de esta.

En la Figura 2 se observa también como una de las aplicaciones que más requerimiento de memoria posee, la relacionada con el almacenamiento de datos médicos asociados a pacientes o tarjetas de salud (también llamadas *smart healthcare card*). Este tipo de aplicaciones puede almacenar una gran variedad de información asociada a pacientes y

datos médicos de estos. Para este tipo de aplicaciones, debido a la capacidad de memoria de las tarjetas inteligentes existentes en el mercado en la actualidad, no pueden ser almacenados ficheros de datos como reportes de laboratorio e imágenes de diagnóstico. Como alternativa este tipo de información es almacenada en un servidor central y accedida por la tarjeta. Además, existen algunas implementaciones de este tipo de aplicaciones que proveen un acceso seguro a sistemas basados en la nube. (Smart Card Alliance, 2012).

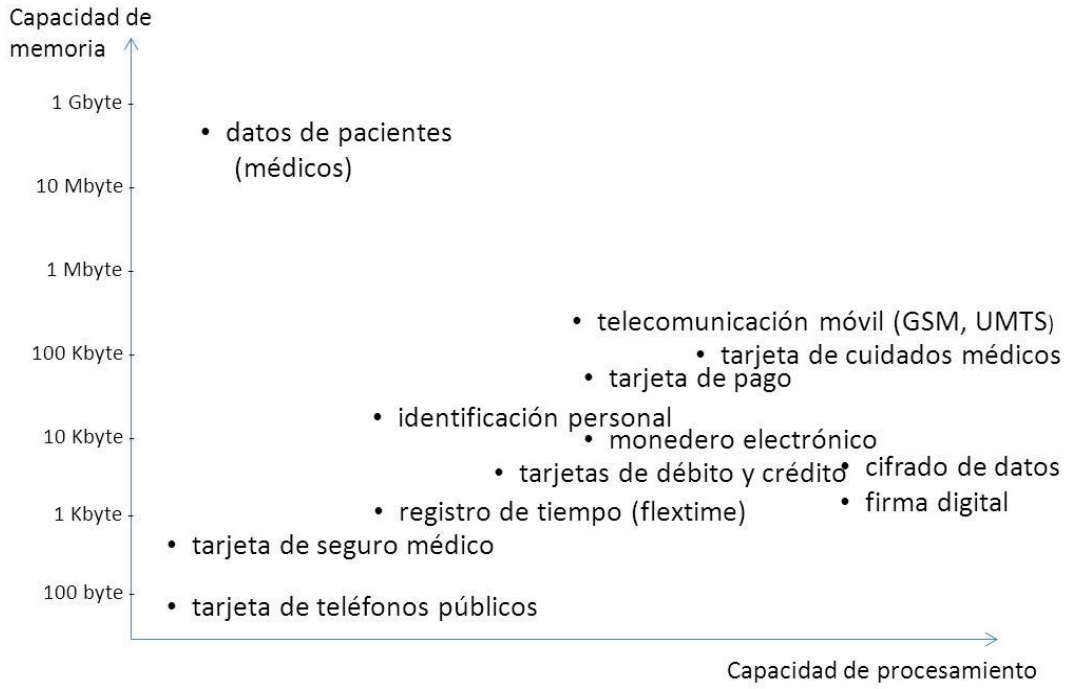

Figura 2. Capacidades de almacenamiento y procesamiento aritmético requerido para algunas áreas de aplicación de las tarjetas inteligentes. (Traducida de (Rankl et al., 2010))

A pesar de la cantidad de aplicaciones disponibles hoy en el mercado para tarjetas inteligentes y que son considerados dispositivos avanzados, con interesantes recursos criptográficos y computacionales comparado con su reducido tamaño, las limitaciones del *hardware* de la tecnología ha provocado que a menudo no se haya ido más allá de considerarlas como elementos de soporte en un esquema global (Márquez, 2006). Los trabajos referenciados en este epígrafe evidencian que en la literatura 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.

1.4. Especificaciones y estándares para tarjetas inteligentes

Con el desarrollo de la tecnología de las tarjetas inteligentes se comenzaron a estandarizar 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. Estos y otros elementos se encuentran recogidos en la colección de estándares ISO/IEC 7816. Además de los estándares físicos y de fabricación se han definido estándares específicos, entidades de tarjetas de crédito (EMV[20]) y fabricantes de teléfonos móviles (GSM), son ejemplos de organizaciones que han adaptado las tarjetas inteligentes a los servicios que ofrecen.

Además se encuentran los estándares y tecnologías que cubren las capas más altas de programación y están relacionados con la comunicación de un ordenador con los terminales de tarjetas; y de manera general con el desarrollo de aplicaciones para tarjetas inteligentes. A continuación se describen los principales estándares y tecnologías para tarjetas inteligentes que son de interés para la investigación.

1.4.1. Colección de estándares ISO/IEC 7816

El principal objetivo de esta colección de estándares es lograr la interoperabilidad entre los distintos fabricantes de tarjetas inteligentes y lectores, en lo que respecta a varios aspectos entre los que se encuentran: características físicas, protocolos de transmisión, comandos y seguridad. Entre los estándares que define la serie se encuentran catorce partes y las principales para el proceso de desarrollo de aplicaciones para tarjetas inteligentes son: ISO/IEC 7816-4: Organización, seguridad y comandos para intercambio (ISO/IEC, 2005a); ISO/IEC 7816-5: Sistema de numeración y procedimientos de registro de proveedores de aplicaciones (ISO/IEC, 2004a); ISO/IEC 7816-6: Elementos de datos inter-industriales para el intercambio (ISO/IEC, 2004b); ISO/IEC 7816-7: Comandos para lenguaje de consulta estructurado para una tarjeta (ISO/IEC, 1999); ISO/IEC 7816-8: Comandos para operaciones de seguridad (ISO/IEC, 2004c); ISO/IEC 7816-9: Comandos para administración de tarjetas (ISO/IEC, 2004d); ISO/IEC 7816-11: Verificación de la identidad personal a través de métodos biométricos (ISO/IEC, 2004e); ISO/IEC 7816-15: Aplicación de información criptográfica (ISO/IEC, 2008c).

Dentro de esta colección de estándares, uno de los fundamentales para el desarrollo de aplicaciones para tarjetas inteligentes es el ISO/IEC 7816-4. De acuerdo a este estándar el intercambio de información entre la tarjeta inteligente y un lector se realiza a través de

[20] Europay, MasterCard y Visa (estándar para la autenticación de pagos mediante tarjetas de crédito y débito).

un APDU[21], el cual no es más que un paquete de información con un formato específico. Las tarjetas inteligentes nunca inician la comunicación con el lector, sino que responden a los comandos que este les envía. Se define dos tipos de APDU, los llamados comandos APDU que son los que envía el lector a la tarjeta y la respuesta APDU que son los que envía la tarjeta al lector como respuesta a un comando APDU. En la Tabla 3 se muestra la estructura definida para un comando APDU según el estándar.

Tabla 3. Estructura de un comando APDU

Encabezado (Obligatorio)				**Cuerpo (Opcional)**		
CLA	INS	P1	P2	Lc	Data	Le

La descripción de los campos del comando APDU se describe en la Tabla 4.

Tabla 4. Campos del comando APDU

Campo	**Tamaño**	**Descripción**
CLA	1 byte	Clase de la instrucción
INS	1 byte	Código que indica la instrucción, es especificado en cada una de las descripciones de los comandos
P1	1 byte	Parámetros de la instrucción. Proveen más información sobre la instrucción
P2	1 byte	
Lc	1 byte	Número de bytes en la *Data* de comando
Data	Lc bytes	Secuencia de bytes con información (datos)
Le	1 byte	Cantidad máxima de bytes esperados como respuesta

La respuesta a los comandos (respuesta APDU) utiliza el formato que se muestra en la Tabla 5.

Tabla 5. Formato de respuesta a los comandos APDU

Cuerpo (Opcional)	**Cola (Obligatoria)**
Data	SW1, SW2

[21] Unidad de datos del protocolo de aplicación (traducido de las siglas en inglés APDU: *Application protocol data unit*).

La descripción de los campos de la respuesta APDU se muestra en la Tabla 6.

Tabla 6. Campos de la respuesta APDU

Campo	Tamaño	Descripción
Data	Hasta Le bytes	Secuencia de bytes con información (Datos)
SW1	1 byte	Palabra de estado. Denotan el estado del procesamiento del comando en la tarjeta
SW2	1 byte	

1.4.2. Estándar PC/SC

Ni la colección de estándares ISO/IEC 7816 ni otras especificaciones para tarjetas inteligentes establecen elementos de interoperabilidad entre aplicaciones como APIs[22] independientes de los dispositivos, herramientas de desarrollo o de compartición de recursos. Debido a esto en mayo de 1996 se creó el PC/SC Workgroup, una alianza formada por importantes fabricantes de ordenadores y tarjetas inteligentes, cuyo objetivo principal ha sido el de desarrollar especificaciones que resolviesen los problemas de interoperabilidad entre las tarjetas y los terminales, así como la cooperación entre el terminal y el sistema operativo.

PC/SC[23] es un conjunto de especificaciones para la estandarización de la integración de las tarjetas inteligentes con los ordenadores personales, desarrolladas por el PC/SC Workgroup. Este conjunto de especificaciones fueron desarrolladas por Microsoft, Hewlett-Packard, Siemens-Nixdorf y algunos proveedores de tarjetas inteligentes, para facilitar la introducción de esta tecnología en el mundo de los ordenadores. Define un API que permite a los desarrolladores trabajar de forma uniforme con lectores de tarjetas de distintos fabricantes que cumplan con esta especificación. El API de PC/SC está incorporada desde sistemas operativos como Microsoft Windows 200x/XP y Microsoft Windows NT/9x. También existe una implementación de código abierto llamada PC/SC Lite para sistemas operativos GNU/Linux. La especificación se divide en diez partes que contienen los requisitos detallados de interoperabilidad de dispositivos compatibles, información de diseño, interfaces de programación y otras (PC/SC Workgroup, 2013).

[22] Interfaz de programación de aplicaciones (traducido de las siglas en inglés API: *Application programming interface*).

[23] Computadora personal/ tarjeta inteligente (traducido de las siglas en inglés PC/SC: *Personal Computer/Smart Card*).

1.4.3. *Java Card*

Desde los inicios del desarrollo de la tecnología de las tarjetas inteligentes, la industria comenzó a buscar una plataforma de software fiable para el desarrollo de estructuras de memoria que pudieran ser desplegadas en estos dispositivos como se describe en (Chen, 2000). En este sentido las investigaciones más importantes estuvieron conducidas por varios grupos en Francia, Holanda y los Estados Unidos. Los desarrollos realizados por estos grupos sirvieron de base y guiaron el camino para que a finales de 1996 la empresa Sun Microsystems acordara diseñar una versión de la plataforma *Java* para tarjetas inteligentes. La industria de manera inmediata colaboró con Sun para ayudar a crear el estándar *Java Card*, cuya primera versión fue lanzada en 1997 (Castel, 2013).

La tecnología *Java Card* permite a las tarjetas inteligentes y otros dispositivos con memoria muy limitada, correr aplicaciones pequeñas (llamadas *applets*) que utilizan la tecnología *Java*. Provee a los vendedores de tarjetas inteligentes una plataforma de ejecución segura e interoperable que puede almacenar y actualizar múltiples aplicaciones en un solo dispositivo (Sun Microsystems, 2008). Esta tecnología cuenta de tres elementos integrados dentro de la tarjeta:

- La máquina virtual *Java Card* (JCVM por sus siglas en inglés: *Java Card Virtual Machine*): provee un subconjunto del lenguaje de programación *Java* y una máquina virtual que permite ejecutar múltiples aplicaciones de manera segura.

- El entorno de ejecución *Java Card* (JCRE por sus siglas en inglés: *Java Card Runtime Enviroment*): es responsable de la administración de los recursos de la tarjeta, de las comunicaciones, la seguridad y la ejecución de los *applets.*

- La interfaz de programación de aplicaciones *Java Card* (API *Java Card*): define el núcleo del *framework* y los paquetes y clases extensión de *Java* para el desarrollo de aplicaciones para tarjetas inteligentes. A través de este API se provee soporte para manejar comandos APDU, códigos PIN, excepciones y rutinas específicas de *Java Card.*

La característica más significativa del entorno de ejecución de *Java Card* es que proporciona una separación clara entre el sistema de la tarjeta y las aplicaciones. El JCRE encapsula la complejidad y los detalles del sistema de la tarjeta, debido a que las aplicaciones solicitan servicios de este sistema y recursos, a través de una interfaz de

programación de alto nivel bien definida. El JCRE es el responsable de gestionar los recursos de la tarjeta, las comunicaciones, la ejecución de los *applets* y la seguridad de los procesos. Se compone de la máquina virtual de *Java Card*, la estructura de clases de aplicación (API), extensiones específicas de la industria y las clases nativas del sistema.

La diferencia principal entre la máquina virtual de *Java Card* y la máquina virtual de *Java* es que la implementación de la JCVM está dividida en dos partes: la de la máquina virtual situada en el interior de la tarjeta que incluye el intérprete *Java Card* y la parte que se encuentra fuera de esta, llamada convertidor. Ambas partes componen toda la funcionalidad de una máquina virtual, es decir, la carga de ficheros de clases *Java* y la ejecución de los mismos. El convertidor carga y pre-procesa los ficheros de clases que forman un paquete *Java* y obtiene como resultado un fichero en formato CAP[24]. El fichero CAP se carga en la tarjeta *Java Card* y se ejecuta en el intérprete. El convertidor tiene también la misión de detectar las partes del lenguaje *Java* no soportadas por *Java Card*. Por su parte, el intérprete de *Java Card* proporciona soporte en tiempo de ejecución para el lenguaje *Java* y por lo tanto permite que el código de las aplicaciones sea independiente del *hardware*. (Márquez, 2006). En la Figura 3 se muestra el esquema completo de la arquitectura de *Java Card.*

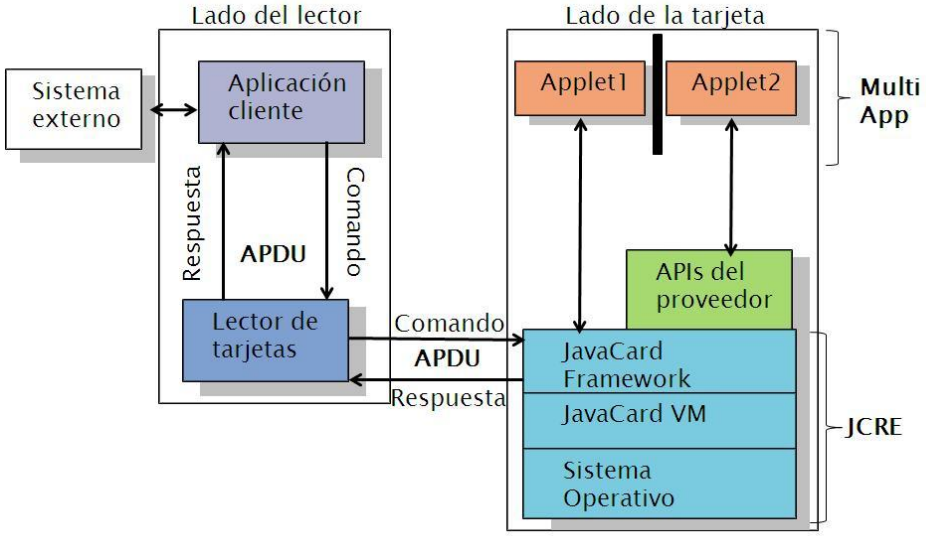

Figura 3. Arquitectura de *Java Card* (Traducida de(Ortiz, 2003))

Como se observa en la figura, los *applets Java Card* no son los únicos elementos que intervienen en el conjunto de una aplicación para tarjetas inteligentes, sino que además

[24] *Applet* convertido (traducido de las siglas en inglés CAP: *Converted applet*).

de la parte de la tarjeta, se incluye la aplicación correspondiente a la parte del terminal o *host* (lado del lector). Esta a su vez podría interactuar con una aplicación remota (sistema externo) que conformaría el sistema completo.

Java Card fue diseñado de tal forma que ciertas construcciones de *Java* consideradas como demasiado complejas o no aplicables para la programación de tarjetas inteligentes no son incorporadas. En la Tabla 7 se muestra un resumen de las principales características soportadas y no soportadas del lenguaje *Java*, por la máquina virtual de *Java Card*.

Tabla 7. Características soportadas y no soportadas de *Java*.

Características de *Java* soportadas	**Características de *Java* no soportadas**
Tipos de datos primitivos pequeños: *boolean, byte, short*	Tipos de datos primitivos grandes: *long, double, float*. Caracteres y *string*. Arreglos multidimensionales
Arreglos de una sola dimensión	Carga dinámica de clases
Paquetes, clases, interfaces y excepciones	Administrador de seguridad
Características de la orientación a objeto: herencia, métodos virtuales, sobrecarga y creación dinámica de objetos.	Recolector de basura
El soporte de la palabra reservada *int* y el tipo de dato *int* de 32 bits es opcional.	Trabajo con hilos
	Serialización y clonado de objetos

La versión 2.2.2 de la plataforma fue liberada en el 2006 y es la sexta actualización basada en la arquitectura original de *Java Card*. En la versión 3 de la plataforma SUN Microsystems introduce una nueva versión de la arquitectura para permitir el soporte a las tarjetas inteligentes con mayores recursos de *hardware* y los últimos estándares criptográficos. Esta nueva arquitectura está separada en dos ediciones: *Classic* y *Connected*. La edición *Classic* (SUN Microsystems, 2011) está basada en una evolución de la plataforma 2.2.2 y soporta el desarrollo tradicional de aplicaciones basadas en *applets*. Por su parte, la edición *Connected* (SUN Microsystems, 2009) presenta un ambiente de ejecución significativamente mejorado y una nueva máquina virtual que incorpora algunos de los elementos del lenguaje *Java* no soportados en versiones anteriores.

1.5. Seguridad en tarjetas inteligentes

ISO/IEC 7816 y *GlobalPlatform* son los principales modelos de seguridad existentes para tarjetas inteligentes. Ambos están compuestos por un conjunto de elementos que se encuentran en el *chip* de la tarjeta y que 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.
- Protocolos de autenticación mutua y 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, objetos para la autenticación de usuario, propiedades específicas que definen el uso que se le puede dar al contexto, entre otros.

Los protocolos de autenticación mutua conforman la primera fase en la comunicación segura entre el terminal y la tarjeta inteligente, y describen cuáles son los pasos necesarios para realizar una autenticación entre ambas partes. Estos protocolos están soportados por un conjunto de comandos APDU. La mensajería segura es la segunda fase en una comunicación segura entre el terminal y la tarjeta inteligente, y solo puede establecerse luego de haberse realizado algún proceso de autenticación mutua satisfactoriamente.

1.5.1. Modelo de seguridad *GlobalPlatform*

GlobalPlatform (GlobalPlatform, 2014) es una organización independiente, enfocada a gestionar una infraestructura estandarizada para el desarrollo y despliegue de tarjetas inteligentes. Proporciona un conjunto de especificaciones universalmente reconocidas e implementadas, junto con configuraciones de mercado y documentos de apoyo. Estos documentos técnicos ofrecen una plataforma completa para el desarrollo de aplicaciones para tarjetas inteligentes, para poder establecer una conexión segura con la misma y administrar sus aplicaciones.

La arquitectura de una tarjeta inteligente con un sistema operativo que cumpla con las especificaciones de *GlobalPlatform* incluye un contexto de seguridad por defecto,

conocido como dominio de seguridad del Proveedor (ISD por las siglas en inglés: *Issuer Security Domain*), al cual pertenece la aplicación *CardManager*[25]. Opcionalmente la tarjeta inteligente podrá tener otros dominios de seguridad denominados dominios de seguridad suplementarios (SSD por las siglas en inglés: *Supplementary Security Domain*) o dominios de seguridad del proveedor de aplicaciones. Todas las aplicaciones que se instalen en una tarjeta con esta arquitectura deben pertenecer a un dominio de seguridad (Global Platform, 2003).

Un dominio de seguridad puede contener varios conjuntos de llaves simétricas o asimétricas. Estos conjuntos de llaves son utilizados en los protocolos de autenticación mutua y mensajería segura, permitiendo garantizar la autenticidad, integridad y confidencialidad en cualquier sesión de comunicación entre el terminal y la tarjeta inteligente (Global Platform, 2003). El conjunto de llaves simétricas está compuesto por las llaves: S-MAC[26], S-ENC[27] y S-DEK[28].

Cada dominio de seguridad es visto como una aplicación más dentro de la tarjeta y por lo tanto, al igual que las aplicaciones que se instalan en el *chip*, tiene un conjunto de privilegios y parámetros específicos que describen para lo que se va a utilizar (Global Platform, 2003). La Figura 4 muestra la arquitectura de los dominios de seguridad y las aplicaciones en una tarjeta inteligente con *GlobalPlatform*.

Las especificaciones de *GlobalPlatform* incluyen un API que permite hacer uso de las funcionalidades, objetos de seguridad y protocolos de canal seguro correspondientes al dominio de seguridad al que pertenece una determinada aplicación (Global Platform, 2011). Este API facilita en gran medida el desarrollo de aplicaciones sobre plataformas como *Java Card* porque permite abstraer al desarrollador de implementar la mayor parte de los elementos de seguridad antes mencionados. Debido fundamentalmente a esto se ha convertido en una de las propuestas más utilizadas para la comunicación entre un *middleware*[29] y su *applet* correspondiente.

---

[25] Es la aplicación representante del proveedor de la tarjeta.

[26] Llave para el cálculo del código de autenticación de mensaje del canal seguro (traducido de las siglas en inglés MAC: *Message authentication code*).

[27] Llave para el cifrado del canal seguro (traducido de las siglas en inglés ENC: *Encryption key*).

[28] Llave para el cifrado de información sensible (traducido de las siglas en inglés DEK: *Data encryption key*).

[29] Se refiere a bibliotecas de clases que exponen métodos de acceso a un *applet* determinado.

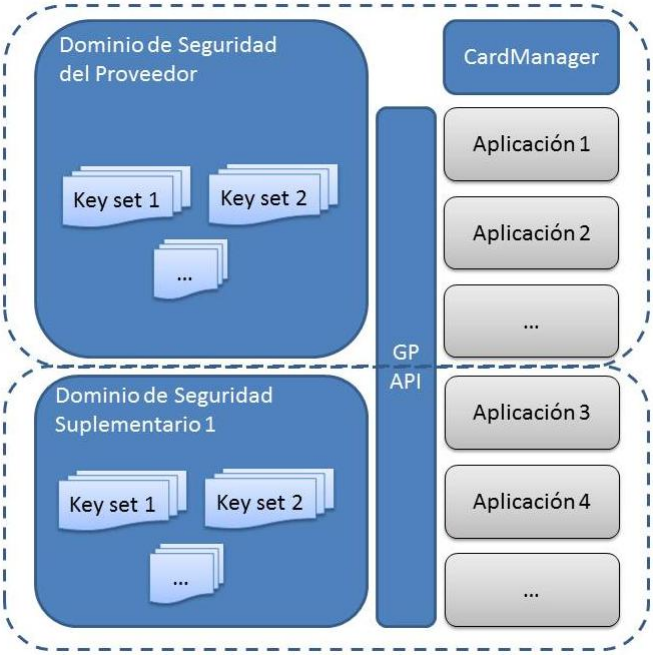

Figura 4. Dominios de seguridad y aplicaciones en una tarjeta inteligente *GlobalPlatform.*

La documentación *GlobalPlatform* define ampliamente la comunicación segura por encima de lo que se define en las especificaciones ISO/IEC 7816 (Jardines, 2013). A continuación, en aras de esclarecer el proceso de envío de comandos e interpretación de las respuestas recibidas, se describirá el protocolo de autenticación mutua simétrica de *GlobalPlatform*.

1.5.1.1. Protocolo de Autenticación Mutua

El proceso de autenticación mutua se basa en el principio de que ambas entidades, el terminal y la tarjeta, deben demostrar que tienen conocimiento sobre un secreto común. Uno de los mecanismos más utilizados es el Reto-Respuesta (también conocido como *Challenge-Response*), mediante el cual las partes intercambian información generada de forma aleatoria y son conformados criptogramas, utilizando las llaves o secretos que ambos conocen, que luego son intercambiados para comprobar que ambos coinciden y se autentican ambas partes.

En el protocolo de canal seguro '01' (SCP por las siglas en inglés*: Secure Channel Protocol*) o SCP01, la autenticación mutua se realiza de forma simétrica, utilizando los comandos *Initialize Update* y *External Authenticate.* El flujo de la autenticación mutua *GlobalPlatform* entre el terminal y un dominio de seguridad en la tarjeta, puede verse en la Figura 5.

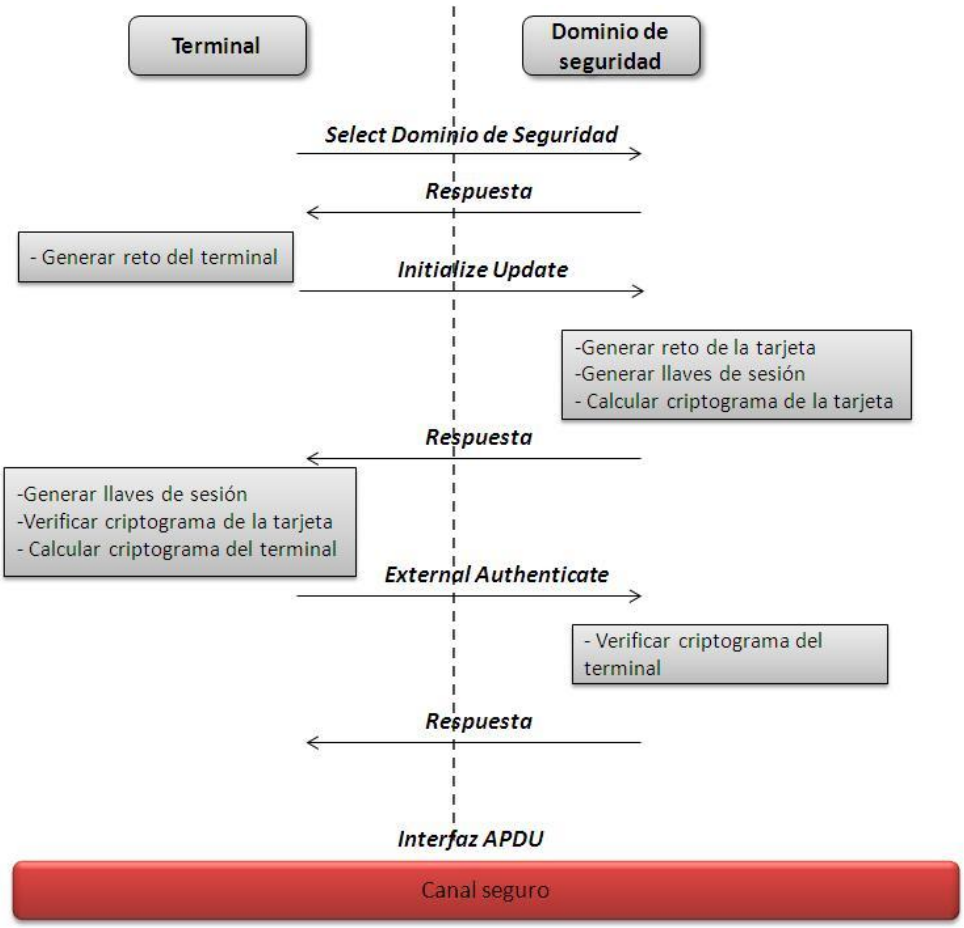

Figura 5. Flujo de autenticación mutua *GlobalPlatform* del SCP01 (Terminal – Dominio de seguridad)

El proceso de autenticación mutua del SCP01 entre el terminal y un dominio de seguridad, comienza cuando el terminal selecciona el dominio de seguridad con el cual quiere establecer un canal seguro, utilizando el comando *Select.* Luego el terminal genera un reto y lo envía en un comando *Initialize Update* hacia la tarjeta. En el momento en que la tarjeta recibe el reto del terminal genera su propio reto. Usando el reto del terminal, el reto generado por ella y las llaves estáticas S-MAC y S-ENC que posee, la tarjeta genera las llaves de sesión para el canal seguro y un criptograma (criptograma de la tarjeta). El criptograma de la tarjeta, de conjunto con su reto, el identificador del protocolo de canal seguro utilizado y otros datos son transmitidos al terminal desde la tarjeta. Como el terminal tiene ahora la misma información que usó la tarjeta para generar su criptograma, es capaz de generar las mismas llaves de sesión del canal seguro y el mismo criptograma de la tarjeta, por tanto, realizando una comparación es posible autenticar a la tarjeta. Luego el terminal usa un proceso similar para crear un criptograma propio (criptograma del terminal) que es enviado a la tarjeta

en un comando *External Authenticate*. Como la tarjeta tiene la misma información que el terminal usó para generar el criptograma del *host*, es capaz de generar el mismo criptograma y autenticar al terminal (Global Platform, 2003).

También es posible establecer un canal seguro de comunicación con alguna aplicación en la tarjeta inteligente. Para esto se requiere que la aplicación pertenezca a un dominio de seguridad y que este tenga definido entre sus parámetros a soportar el protocolo específico de canal seguro a utilizar. Los comandos antes mencionados se le deben enviar a la aplicación y no al dominio de seguridad al que esta pertenece. La aplicación de la tarjeta puede hacer uso del API de *GlobalPlatform* para realizar las operaciones criptográficas necesarias en el proceso de autenticación mutua y en el uso del canal seguro establecido.

Las versiones más actuales de las especificaciones *GlobalPlatform* soportan protocolos de autenticación mutua asimétricos, tal es el caso del protocolo SCP10. En esta versión los dominios de seguridad soportan conjuntos de llaves asimétricas (públicas y privadas) utilizando el algoritmo RSA (RSA Laboratories, 2002) y certificados digitales (Global Platform, 2011).

1.5.1.2. Mensajería segura

La mensajería segura es la segunda fase dentro de un canal seguro de comunicación y tiene como objetivo que todos los comandos sean intercambiados utilizando el nivel de seguridad (integridad y/o confidencialidad) establecido con anterioridad, durante la negociación del canal de comunicación (la autenticación mutua). El nivel de seguridad establecido se aplica a todos los comandos que son intercambiados entre la tarjeta y el terminal hasta que termina la sesión del canal seguro. Los niveles de seguridad que establece *GlobalPlatform* (especificados en el comando *External Authenticate*) son: No mensajería segura (0x00), MAC (0x01) y Cifrado y MAC (0x03).

La integridad de los comandos está dada por la generación de una MAC (C_MAC) que se obtiene mediante la aplicación de múltiples operaciones DES[30] en cadena, usando las llaves de sesión generadas durante el proceso de autenticación mutua. Se utilizan los algoritmos *Full Triple DES MAC y Retail MAC* para el cálculo de la MAC de todo el comando. La tarjeta al recibir un comando que contiene una C_MAC, usando las mismas llaves de sesión del canal seguro, realiza la misma operación y compara la C_MAC

[30] *Data Encryption Standard.*

generada internamente con la C_MAC recibida por el terminal, asegurando la integridad de todos los comandos. La integridad de una secuencia de comandos transmitida a la tarjeta es lograda usando la C_MAC del comando actual, así como un ICV[31] para todos los comandos subsecuentes (Global Platform, 2003).

Para la confidencialidad de los datos transmitidos en los mensajes se puede realizar el cifrado del campo de datos de los comandos APDU mediante la aplicación de múltiples operaciones DES en cadena, usando también las llaves de sesión generadas durante el proceso de auntenticación mutua. Para ello se pueden utilizar algoritmos simétricos como el 3DES[32] en los modos ECB[33]/CBC[34]. Es posible aplicar también un nivel de seguridad adicional para los datos sensibles (se utiliza la llave S-DEK) que siempre serán transmitidos como datos secretos (ej. llaves privadas y secretas).

1.5.2. Modelo de seguridad ISO/IEC 7816

ISO/IEC 7816 es otro de los estándares que define la comunicación segura con una tarjeta inteligente. Para ello brinda el concepto de entorno de seguridad (SE por las siglas en inglés: *Security Enviroment*). Un entorno de seguridad es una estructura (análoga a un dominio de seguridad de *GlobalPlatform*) donde se almacenan datos concernientes a algoritmos criptográficos, modos de operación, llaves simétricas, asimétricas y códigos PIN, necesarios para la mensajería segura y operaciones de seguridad sobre ficheros que se encuentran dentro de una tarjeta inteligente (ISO/IEC, 2005a).

Según este estándar la información es almacenada en contenedores denominados Ficheros Elementales (EF por las siglas en inglés: *Elementary Files*), los cuales pertenecen a un Fichero Dedicado (DF por las siglas en inglés: *Dedicated File*), conformando así un sistema de archivos cuya raíz es el Fichero Máster (MF por las siglas en inglés: *Master File*). Cada EF presenta determinadas condiciones de seguridad que deben ser cumplidas para permitir su acceso. Los datos necesarios para solventar estos mecanismos de seguridad se encuentran guardados en los componentes que forman parte del entorno de seguridad (Figura 6).

---

[31] Vector inicial de cadena (traducido de las siglas en inglés *Initial chaining vector*).

[32] *Triple Data Encryption Standard*.

[33] Libro de código electrónico (traducido de las siglas en inglés ECB: *Electronic Code Book*)

[34] Bloques de cifrado en cadena (traducido de las siglas en ingles CBC: *Cipher-Block Chaining*)

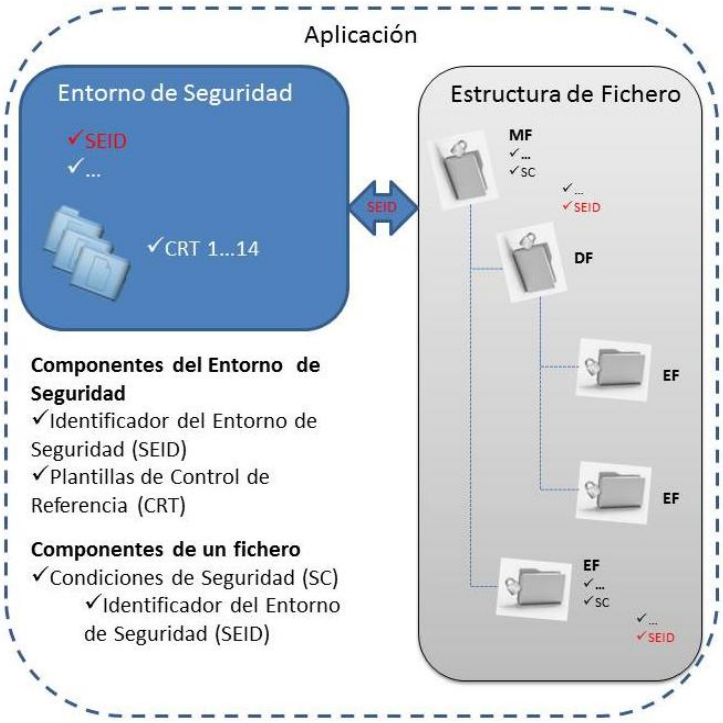

Figura 6. Arquitectura de entornos de seguridad por ficheros de ISO/IEC 7816

Un entorno de seguridad está formado por tres componentes que son almacenados dentro de la tarjeta en formato TLV[35] (ISO/IEC, 2005a). Los componentes son los siguientes:

- El identificador del entorno de seguridad (SEID por las siglas en inglés: *Security Environment Identifier*).
- El estado del ciclo de vida (LCS por las siglas en inglés: *Life Cycle State*).
- La plantilla de control de referencia (CRT por las siglas en inglés: *Control Reference Template*).

El SEID es un byte que identifica a cada uno de los entornos de seguridad que estén presentes en la tarjeta. El valor del ciclo de vida indica el estado del ciclo de vida del entorno de seguridad. La plantilla de control de referencia es un conjunto de objetos en formato TLV que brindan una especificación completa del algoritmo criptográfico a utilizar y contienen un conjunto de objetos de datos de control de referencia (CRDO por las siglas en inglés *Control Reference Data Object*), que contienen la referencia a la

[35] Etiqueta, longitud, valor (traducido de las siglas en inglés TLV: *Tag, length, value*).

información a utilizar y los datos adicionales que se necesitan dentro de la tarjeta inteligente, en los mecanismos de seguridad para acceder a los ficheros (ISO/IEC, 2005a).

Existen seis tipos de CRT, ellos son: la Plantilla para la Autenticación (AT por las siglas en inglés: *Template for authentication*); la Plantilla de Acuerdo de Clave (KAT por las siglas en inglés: *Template for key agreement*); la Plantilla de Checksum Criptográfico (CCT por las siglas en inglés: *Template for cryptographic checksum*); la Plantilla de Firma Digital (DST por las siglas en inglés: *Template for digital signature*); La Plantilla Confidencial (CT por las siglas en inglés: *Template for confidentiality*); y por último, la Plantilla de *Hash* (HT por las siglas en inglés: *Template for hash-code*).

1.5.2.1. Protocolo de Autenticación Mutua

La información que se almacena en un entorno de seguridad no solo se utiliza para garantizar la seguridad en el acceso a los datos contenidos en los ficheros, sino que puede ser aprovechada por el proceso de autenticación mutua simétrica o asimétrica entre el terminal y la tarjeta. Para el caso de la autenticación simétrica, los CRT de autenticación y confidencialidad (*AT* y *CT*) contienen la llave madre o secreto que comparten ambas partes, de la cual se generan las llaves estáticas S-ENC y S-MAC y luego las llaves de sesión del canal seguro, al igual que en *GlobalPlatform*. Las llaves de sesión del canal seguro, unido a los criptogramas y a los datos aleatorios que se generan por ambas partes, se utilizan para validar el reto enviado por las dos partes de forma recíproca.

El proceso de autenticación mutua asimétrica también se puede llevar a cabo utilizando las facilidades que brindan los entornos de seguridad para almacenar datos de un certificado digital (firma digital) y la clave pública para verificarlo. En este caso la información para la autenticación se puede guardar en otros CRT, como los de firma digital, acuerdo de llave y autenticación (DST, KAT y AT) presentes en los entornos de seguridad.

Para crear, establecer y verificar los datos que se van a necesitar en el proceso de autenticación mutua (simétrica o asimétrica) en los entornos de seguridad, se utilizan un grupo de comandos APDU definidos en el estándar ISO/IEC 7816-4. Estos son los comandos principales: *Put Data*, *Get Challenge*, *Manage Security Enviroment (MSE)*, *Perform Security Operation*, *Mutual Authenticate*, *Internal Authenticate* y *External Authenticate*. Es importante aclarar que el estándar ISO/IEC 7816 no describe el

protocolo de autenticación mutua en detalle como *GlobalPlatform*, solo provee los comandos para crear, establecer y verificar los datos que se van a necesitar en el proceso de autenticación mutua.

1.5.2.2. Mensajería segura

En la mensajería segura que se define en el estándar ISO/IEC 7816, al igual que en las especificaciones de *GlobalPlatform,* la obtención de las llaves de sesión para S-MAC y *S*-ENC*,* el procedimiento para calcular la C-MAC y el cifrado de un comando APDU, se realizan de forma similar. 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. Los niveles de seguridad que define ISO/IEC 7816 son: No mensajería segura (0x00), Comando con mensajería segura (0x01), Respuesta con mensajería segura (0x02) y Comando y respuesta con mensajería segura (0x03).

1.6. Soluciones de software que combinan las tarjetas inteligentes con ordenadores para lograr mayor capacidad de cómputo y almacenamiento

Como uno de los enfoques para contrarrestar las limitaciones del *hardware* de las tarjetas inteligentes se encuentra el uso de soluciones de software que permitan explotar la capacidad computacional y de memoria de los ordenadores, de conjunto con el de las tarjetas inteligentes. En los epígrafes subsiguientes se describen algunas de estas soluciones, fundamentalmente en las áreas de verificación biométrica y cifrado de datos.

1.6.1. Verificación biométrica mediante el uso de tarjetas inteligentes

La verificación biométrica tiene la ventaja de garantizar que solo el usuario autorizado puede tener acceso a determinada información o áreas. Como mecanismo para obtener el máximo de seguridad se usan las tarjetas inteligentes, que permiten almacenar de manera segura en un ambiente cerrado, características biométricas tales como la huella dactilar, la retina, el iris, etc. Dentro de estas características la huella dactilar es la más utilizada para realizar una identificación biométrica. Además de almacenar las plantillas biométricas[36] en la tarjeta, existen aplicaciones que permiten realizar el *matching* dentro

[36] Representación digital de una característica distintiva de un individuo extraída de una muestra biométrica, que contiene la información necesaria para realizar el reconocimiento biométrico (ISO/IEC, 2005b).

de la misma, sin que la plantilla sea expuesta. Estas aplicaciones son conocidas como *Match on Card* (MoC).

Debido a las características del *hardware* de las tarjetas inteligentes antes descritas, existen limitaciones en este tipo de aplicaciones, relacionadas fundamentalmente con el tamaño de almacenamiento de las plantillas biométricas y la complejidad de los algoritmos de verificación que se utilizan. En este sentido se han realizado trabajos que optimizan estos algoritmos MoC como (Bergman, 2008; Bourlai, Kittler, & Messer, 2010; Bourlai, Messer, & Kittler, 2005; Bringer, Chabanne, Kevenaar, & Kindarji, 2009). Se coincide con (Bourlai et al., 2010) en que de manera general, el reconocimiento biométrico usando tarjetas inteligentes ha recibido relativamente poca atención en comparación con el uso de un ordenador o un servidor como sistema anfitrión. Esto es debido mayormente a que el uso de plataformas pequeñas como las tarjetas inteligentes, con restricciones de *hardware* para realizar estas tareas, es complejo.

Como alternativa a las aplicaciones que usan algoritmos de verificación biométrica en la tarjeta, existen las llamadas aplicaciones *Match off Card.* En este tipo de aplicaciones la plantilla biométrica es recuperada de la tarjeta y el *matching* es realizado afuera de la misma, usando los recursos de *hardware* de un ordenador. Existen algunos trabajos que demuestran que las tasas de errores de las implementaciones de los algoritmos *Match off Card* son menores que las de los algoritmos *Match on Card.* Por ejemplo, en (Grother, Salamon, Watson, Indovina, & Flanagan, 2009) se comparan las implementaciones de tres proveedores diferentes y para todas ellas, los algoritmos *Match on Card* producen aproximadamente de un 20 a un 80% más falsos positivos que las implementaciones *Match off Card*. Evidentemente la selección del lugar donde va a ocurrir la verificación biométrica (dentro o fuera de la tarjeta) depende en gran medida de la arquitectura del sistema a implementar y sus requerimientos, pero también es importante considerar estos resultados.

1.6.2. Protocolo o Esquema de Encriptación Remota usando tarjetas inteligentes

Como ejemplo temprano del uso de ordenadores de conjunto con las tarjetas inteligentes para aplicaciones de cifrado se encuentra el trabajo publicado en (Blaze, 1996). En este trabajo Matt Blaze expone el buen rendimiento de las tarjetas inteligentes criptográficas en aplicaciones tales como protocolos de autenticación Reto – Respuesta y en la firma digital de mensajes. Sin embargo se plantea que está limitada la utilidad de estas tarjetas en aplicaciones que requieren el cifrado de archivos, tráfico en tiempo real,

multimedia y video, donde la velocidad del sistema está limitada por la latencia, la velocidad de la interfaz de la tarjeta inteligente y la capacidad computacional de esta.

En este artículo se introduce el protocolo *Remotely Keyed Encryption Protocol* (RKEP). La propuesta que se presenta consiste en que el procesador del *host* sea el que asuma la mayor parte de las operaciones criptográficas que requieran una alta capacidad computacional y velocidades. Este enfoque implica un intercambio entre parámetros tales como seguridad, rendimiento y costo. Finalmente se escoge una variante en que no se incrementan considerablemente los requerimientos de seguridad. Consiste básicamente en un sistema de cifrado de clave secreta en el que una tarjeta inteligente, segura, pero limitada computacionalmente, desvía la mayor parte del procesamiento a un *host* inseguro, pero rápido. De esta manera se plantea que pueden ser procesados por un *host* bloques de tamaño arbitrario, con el intercambio de un simple mensaje con la tarjeta inteligente donde se encuentra almacenada la llave secreta. Con esta propuesta se obtiene un protocolo menos seguro que el uso de una tarjeta criptográfica, pero con mejores resultados en cuanto a velocidad y a capacidad computacional se refiere.

Luego de esta primera publicación se realizan otras (Blaze, Feigenbaum, & Naor, 1999; Lucks, 1997) donde tomando como base el *Remotely Keyed Encryption Protocol (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 para mejores resultados. Posteriormente Lucks (Lucks, 1999) describe un nuevo esquema, el *Accelerated Remotely Keyed Encryption Scheme* (ARK) que plantea ser más eficiente y cumplir los mismos requerimientos de seguridad que (Blaze et al., 1999).

A pesar de las diferencias y formalismos en cuanto a la seguridad, todos estos protocolos poseen el mismo principio, que consiste en 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.

1.6.3. Extensión de las capacidades de almacenamiento de tarjetas *Java Card*

Como resultado de la investigación y desarrollo de Clemens H. Cap, Nico Maibaum y Lars Heyden de la Universidad de Rostock en Alemania, en (Cap et al., 2001) se presenta una extensión para una *smart card* (SCE por las siglas del inglés: *smart card*

extension), como un nuevo concepto para aumentar la memoria de una tarjeta *Java Card*. La idea básica del concepto que se presenta es la de almacenar los datos de las aplicaciones no en la propia tarjeta inteligente, sino en un servidor externo, para lo cual se necesita un ambiente conectado.

La solución que brindan cuenta de dos componentes de software: un servicio de comunicación que provee una conexión segura al servidor de memoria y un administrador de memoria que es el responsable de decidir qué datos serán mapeados en la tarjeta y cuáles en el servidor. Para la comunicación con el lector de tarjetas y la administración de la tarjeta proponen la utilización de OCF[37]. OCF surgió como iniciativa de estandarizar el acceso a las tarjetas inteligentes en un entorno *Java*. Es una colección extensible de clases para el terminal que soporta el desarrollo de aplicaciones *Java Card*. En sus inicios fue una iniciativa del Consorcio Open Card, pero luego las implementaciones fueron dirigidas por IBM y Gemalto. En la actualidad su desarrollo se encuentra inactivo y se considera un estándar obsoleto (CardLogix Corporation, 2009).

Como principio de esta solución se define que la SCE debe ser transparente para la aplicación del terminal donde se encuentre la tarjeta inteligente. La decisión de delegar el acceso a datos al servidor de memoria externa puede hacerse desde el propio terminal o desde la tarjeta. Luego de un análisis de las características que cada diseño implica, concluyen que un diseño más seguro es en el que la propia tarjeta inteligente tome la decisión, aunque implica la modificación del código residente en la tarjeta.

Aunque la decisión de extensión sea realizada desde la tarjeta, la solicitud enviada al servidor de memoria por el *Card Service* puede estar comprometida. Para prevenir esta situación no deseada, la tarjeta puede enviar las solicitudes con una firma digital o cifrada (Figura 7). Como otros elementos de la arquitectura se selecciona RMI[38] en *Java*, además de SSL[39] para el transporte de los datos, completando así la seguridad de la arquitectura propuesta.

---

[37] Framework Open Card (traducido de las siglas en inglés OCF: *Open Card Framework*: http://www.opencard.org/).

[38] Invocación remota de métodos (traducido de las siglas en inglés RMI: *Remote method invocation*).

[39] Capa de conexión segura (traducido de las siglas en inglés SSL: *Secure Sockets Layer*).

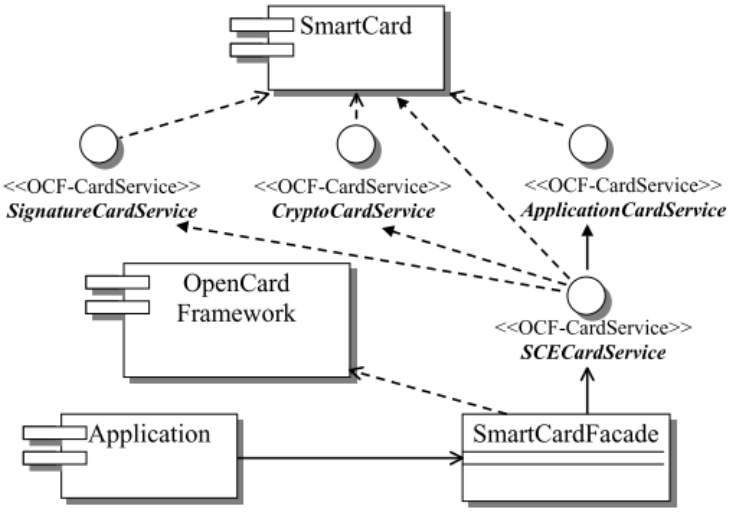

Figura 7. Arquitectura de la *Smart Card Extension* (Cap et al., 2001)

Como experiencia en su aplicación se describen brevemente los resultados obtenidos en el proyecto FASME (por las siglas en inglés *Facilitating Administrative Services for Mobile Europeans*) (Auerbach & Maibaum, 2002).

Conclusiones del Capítulo

En la revisión de la literatura 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, se considera que no ha sido lo suficientemente explotada por la industria o la academia la idea de proveer soluciones de software a estas limitaciones.

En este capítulo se referencian algunas soluciones de software que tienen como propósito ampliar las capacidades de memoria o procesamiento de las tarjetas inteligentes mediante el uso de ordenadores. Se considera que son insuficientes las soluciones de este tipo que explotan este enfoque para contrarrestar las limitaciones propias del *hardware* de las tarjetas inteligentes, y esto se debe en gran medida a que esta tecnología es controlada y desarrollada fundamentalmente por empresas productoras de tarjetas inteligentes. Ninguna de las soluciones encontradas se encuentra estandarizada ni es de propósito general, sino para áreas de aplicación específicas, por lo que no cumplen los requerimientos de la solución que se pretende obtener.

Después de analizada la solución para la extensión de las capacidades de almacenamiento de tarjetas *Java Card* (Cap et al., 2001), se considera que esta solución es la que cumple con un mayor número de requisitos deseados para la solución que se pretende lograr como parte de esta investigación. Entre estos requisitos identificados se encuentran:

- La posibilidad de almacenar datos asociados a 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.

- La implementación de mecanismos de seguridad que proporcionan un ambiente seguro fuera de la tarjeta.

- El diseño de la extensión *smart card* (SCE) de manera que sea transparente para las aplicaciones del terminal.

- La decisión de almacenar los datos en el servidor de memoria se puede realizar desde la tarjeta, siendo este un diseño más seguro que el de delegar esta responsabilidad al terminal.

Sin embargo también se observan varias limitaciones importantes a considerar:

- Se realiza la extensión de la capacidad de almacenamiento de las tarjetas *Java Card*, pero no de su capacidad de procesamiento.

- Documentación incompleta de la solución y no acceso al código fuente.

- Condicionamiento de la arquitectura al uso del Open Card Framework que se considera un estándar obsoleto.

Con este análisis queda demostrada la necesidad de crear un modelo de referencia para la extensión de las capacidades no solo de memoria, sino también de procesamiento, de las tarjetas inteligentes *Java Card*. La caracterización realizada de estándares, especificaciones y modelos de seguridad para tarjetas inteligentes en este capítulo, sirvió de base para el desarrollo del modelo de extensión que se propone como parte de esta investigación.

Capítulo 2. Modelo para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes

En este capítulo se presenta la propuesta de un modelo para extender las capacidades de procesamiento y memoria de las aplicaciones *Java Card* en tarjetas inteligentes, mediante el uso de forma segura de los recursos de *hardware* de un ordenador. El modelo de extensión que se propone provee un mecanismo para el almacenamiento de datos asociados a las aplicaciones de las tarjetas inteligentes y para la ejecución de algoritmos de alto costo computacional, que por el tiempo de ejecución y/o complejidad, sean más factibles realizar fuera de la tarjeta. Se describen las premisas del modelo, sus componentes principales, el flujo de comunicación entre ellos y los mecanismos de seguridad que se proponen para garantizar un ambiente de ejecución seguro.

2.1. Modelo tradicional de desarrollo de aplicaciones *Java Card*

Existen varios elementos que intervienen en el desarrollo tradicional de aplicaciones *Java Card* para tarjetas inteligentes. Entre estos elementos se encuentran los *applets Java Card* que corren dentro del ambiente de la tarjeta mediante el 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. El terminal o *host* contiene las aplicaciones que comúnmente residen en un ordenador. Aquí se encuentran los *middlewares*, que no son más que bibliotecas de clases que exponen métodos de acceso a un *applet* determinado en una tarjeta inteligente (Ramírez et al., 2014).

La comunicación entre el terminal y un *applet Java Card* determinado se basa en el modelo de comunicación de Paso de Mensajes, usando los comandos APDU definidos en (ISO/IEC, 2005a). Los *middlewares* envían comandos APDU con las peticiones a la tarjeta inteligente, mediante la comunicación con un lector de tarjeta que utiliza el estándar PC/SC para la conexión con el ordenador. Dichos comandos APDU son recibidos y procesados por un *applet* específico que devuelve una respuesta en el formato de una respuesta APDU. En las Figuras 8 y 9 se muestra el esquema completo de estos componentes y el flujo de comunicación entre los mismos.

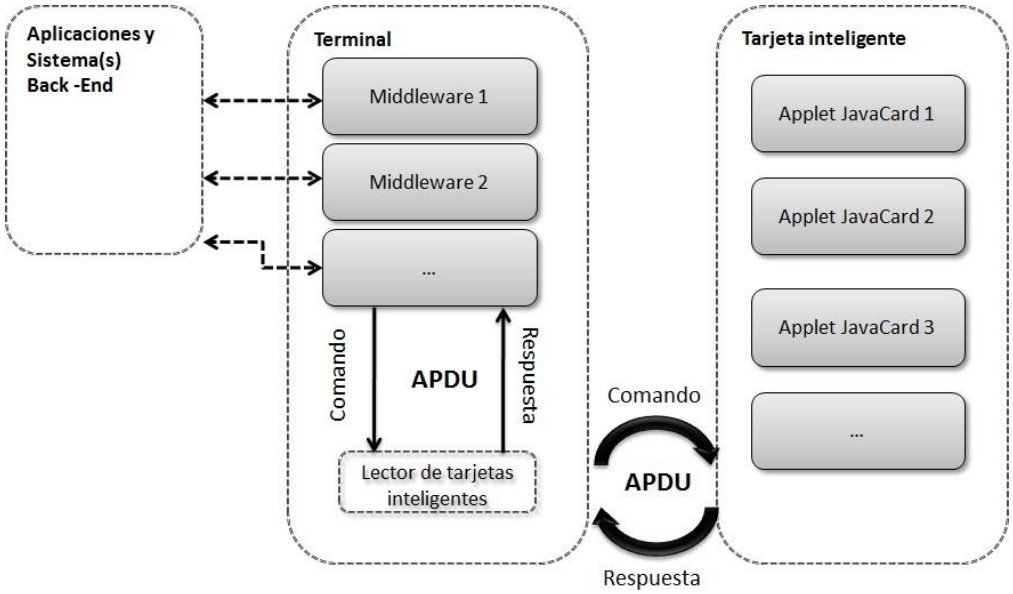

Figura 8. Modelo tradicional de desarrollo de aplicaciones *Java Card* para tarjetas inteligentes

En la Figura 9 puede verse como el modelo tradicional de desarrollo 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 que nunca toma la iniciativa.

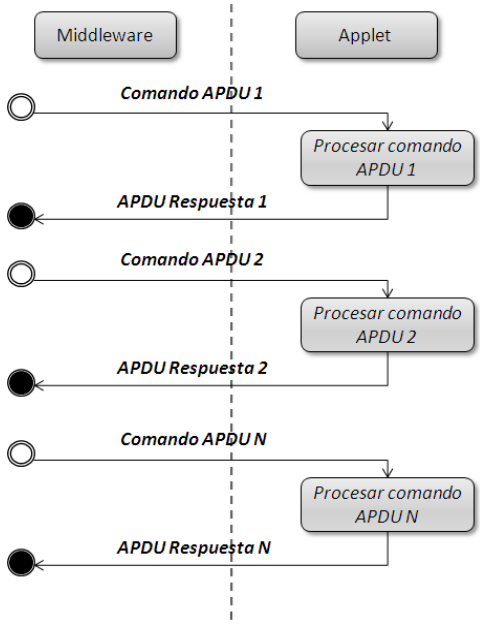

Figura 9. Flujo de comunicación del modelo tradicional de desarrollo de aplicaciones *Java Card* para tarjetas inteligentes

Ante la necesidad de tratar con las limitaciones de *hardware* de las tarjetas inteligentes, fundamentadas con anterioridad, se propone un modelo de extensión que permite obtener una mayor capacidad de procesamiento y memoria para aplicaciones *Java Card* en tarjetas inteligentes, con la utilización de los recursos de *hardware* de un ordenador de conjunto con la tarjeta. En epígrafes siguientes se presenta este modelo extensión, sus principios, representación general y la propuesta específica para algunas tecnologías.

2.2. Principios del modelo para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes

Tomando como base el modelo tradicional de desarrollo para tarjetas inteligentes *Java Card* explicado en el epígrafe anterior, así como los estándares de comunicación ISO/IEC 7816, PC/SC, y partiendo de la problemática de las limitantes de los recursos de *hardware* de las tarjetas inteligentes con bajas prestaciones, se presenta el siguiente modelo general para la extensión de las capacidades de procesamiento y memoria de las tarjetas inteligentes *Java Card* (Figura 10).

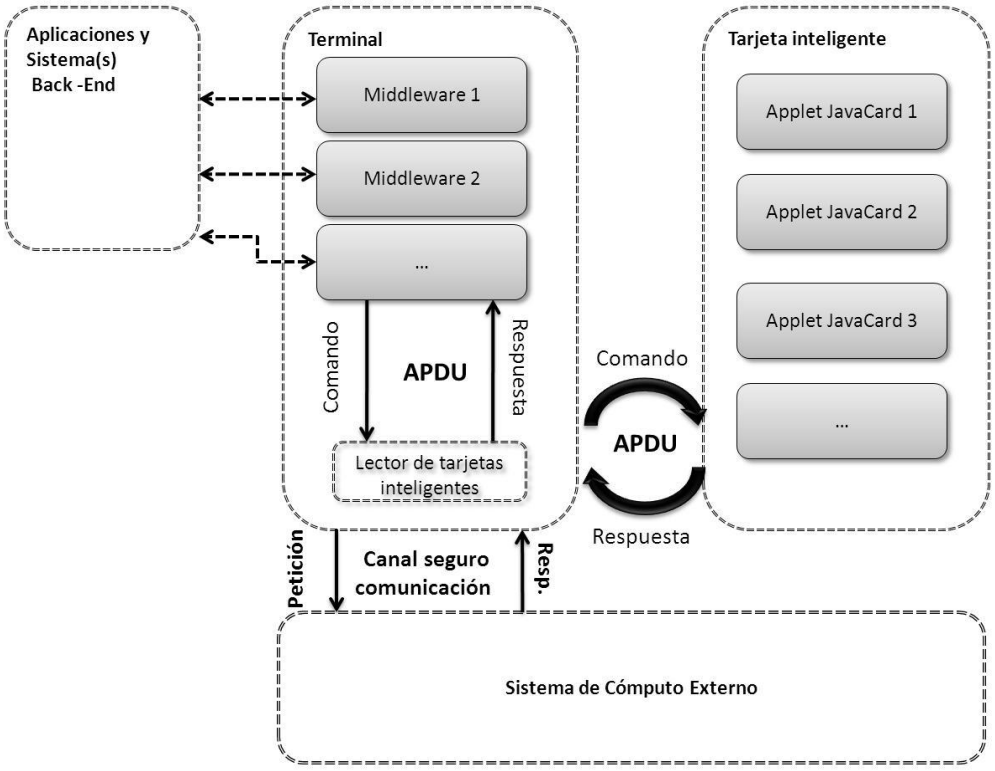

Figura 10. Vista general del modelo de extensión.

La propuesta se basa en la utilización de los recursos de *hardware* de un ordenador con mayores prestaciones que las que posee una tarjeta inteligente estándar, que se propone como un Sistema de Cómputo Externo. A este Sistema de Cómputo Externo se le envían peticiones de forma segura desde un terminal para la ejecución de operaciones de extensión[40](OE), ya sean de almacenamiento o acceso a determinada información, y/o ejecución de algoritmos de alto coste computacional.

Los principios que se tuvieron en cuenta para la definición del modelo de extensión, son los siguientes:

- La comunicación entre los componentes del modelo debe basarse en un modelo de comunicación síncrono (comando - respuesta) como se define en el estándar ISO/IEC 7816.

- Para generar menor impacto en el desarrollo de nuevas aplicaciones, el terminal y la tarjeta inteligente deben abstraerse de la ejecución de operaciones de extensión por el Sistema de Cómputo Externo.

- En la comunicación con el Sistema de Cómputo Externo se deben implementar estándares de seguridad, que garanticen un ambiente de ejecución seguro fuera de la tarjeta inteligente.

- Se debe contar con un ambiente conectado, que posea un buen enlace de red, de manera que la comunicación con el Sistema de Cómputo Externo no constituya un costo adicional significativo.

Con estos principios se pretende garantizar una solución que cumpla con los principales estándares asociados a la tecnología de las tarjetas inteligentes y provea simplicidad, productividad y seguridad de cara a los desarrolladores de aplicaciones para tarjetas inteligentes. Teniendo en cuenta los principios definidos y la propuesta general del modelo de extensión, se presentan a continuación los componentes y tecnologías específicas que se proponen implementar como parte de tres ambientes diferentes: el terminal, la tarjeta inteligente y el Sistema de Cómputo Externo.

2.3. Componentes del modelo de extensión

[40] Se denominan así, en el ámbito de esta investigación, a las operaciones que se realizan fuera de la tarjeta, ya sea por la capacidad de procesamiento o memoria que requieren.

El modelo de extensión que se propone integra nuevos componentes de software al modelo tradicional de desarrollo de aplicaciones *Java Card* para tarjetas inteligentes, en tres ambientes diferentes: la tarjeta inteligente, el terminal y un servidor web (que representa al Sistema de Cómputo Externo de la vista general). De esta manera se pretende desacoplar las responsabilidades de todos los componentes y lograr simplicidad en la ejecución de las operaciones de extensión para *applets Java Card*, que por sus características requieran mayor capacidad de memoria o procesamiento de la que la tarjeta es capaz de proveer (Figura 11).

Para el terminal se propone un componente que funcione como intermediario de toda la comunicación, llamado Componente Proxy. Este componente es el responsable de abstraer a un *middleware* de la ejecución de operaciones de extensión fuera de la tarjeta inteligente.

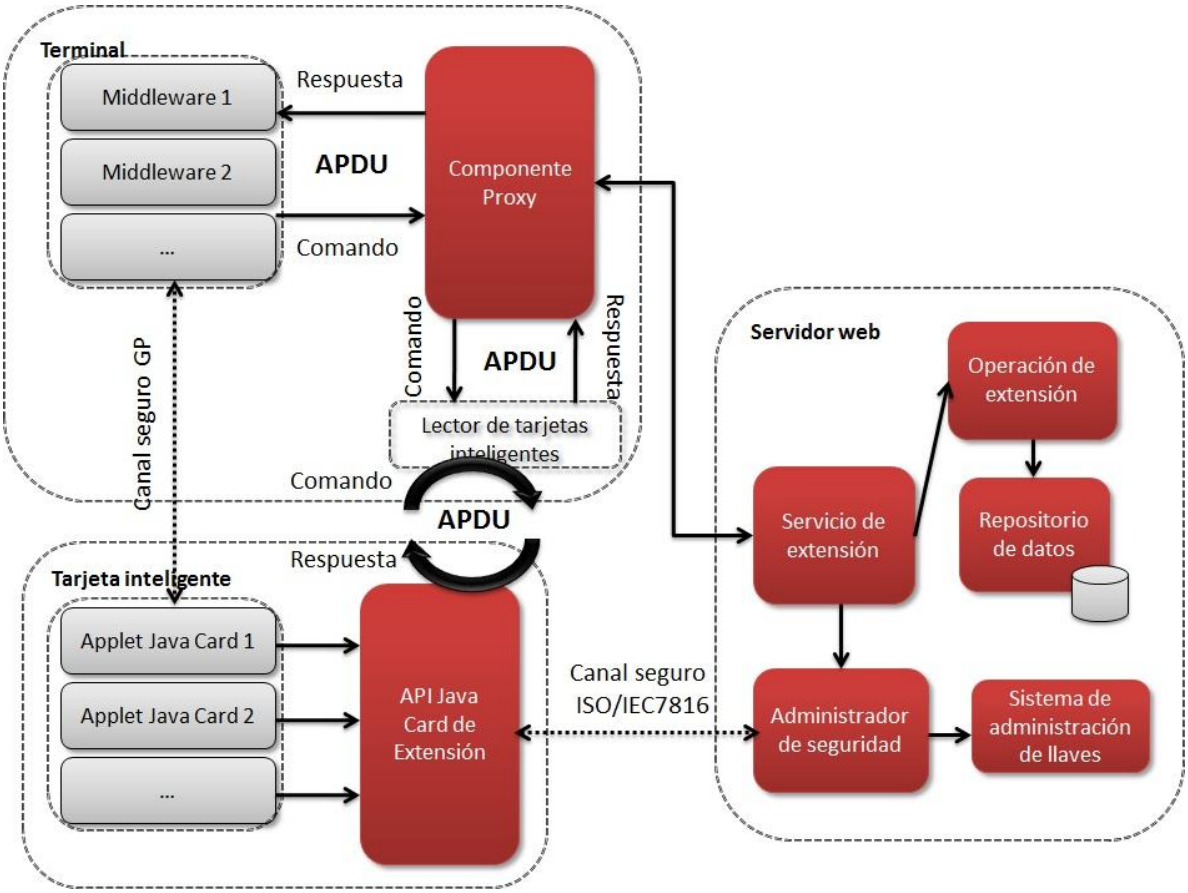

Figura 11. Componentes del modelo de extensión.

En la tarjeta inteligente se propone un API denominado API *Java Card* de Extensión, el cual es responsable de abstraer a un *applet* en la tarjeta de la ejecución de las

operaciones de extensión fuera de esta. Con este API se brindan funcionalidades que permiten a un *applet* conformar una solicitud de operación de extensión, o procesar el resultado de la misma. Además tiene la responsabilidad de manejar los mecanismos de seguridad del modelo para la tarjeta inteligente.

Como implementación concreta del Sistema de Cómputo Externo se propone un servidor web con un conjunto de componentes desplegados. Entre estos componentes se encuentra un servicio llamado Servicio de extensión, que tiene la responsabilidad de ejecutar operaciones de extensión previamente definidas y publicadas. Se provee también un Repositorio de datos para el almacenamiento de información fuera de la tarjeta, de manera que sea posible su recuperación con posterioridad. Para el manejo de la seguridad del servidor web se propone un Administrador de seguridad, que hace uso de un Sistema de administración de llaves (KMS[41]) para el almacenamiento seguro de las llaves a utilizar en la comunicación segura con la tarjeta inteligente.

Además de estos nuevos componentes, se mantienen los componentes de software fundamentales de una solución para tarjetas inteligentes *Java Card*: los *middlewares* y sus *applets* correspondientes, que son implementados de igual manera que en el modelo tradicional. De igual manera también, la comunicación entre un *middleware* determinado, el componente Proxy y un *applet Java Card* es a través de un comando APDU y su respuesta correspondiente. Los componentes de software que se proponen como parte de este modelo de extensión y sus principales responsabilidades se describen en los epígrafes siguientes.

2.3.1. API Java Card de Extensión

Este componente va instalado en la tarjeta inteligente de conjunto con los *applets Java Card*. Este API provee un conjunto de funcionalidades 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.

Entre las principales responsabilidades que debe tener este API se encuentran:

- Conformar la petición de una operación de extensión.
- Enviar una respuesta APDU con la petición de la operación de extensión.

[41] Sistema de administración de llaves (traducido de las siglas en inglés KMS: *Key Management System*).

- Enviar una cadena de respuestas APDU con datos de la petición de la operación de extensión.

- Recibir un comando APDU con la respuesta de la ejecución de una operación de extensión.

- Procesar el resultado final de la operación de extensión.

Para la petición de una operación de extensión se define una estructura TLV. Esta estructura (Tabla 8) contiene los datos necesarios para realizar la solicitud de una operación de extensión al Servicio de extensión. Los datos que se envían desde la tarjeta son: el identificador del *chip*, el identificador del *applet*, el identificador de la operación de extensión que se está solicitando y como parámetros todos los necesarios para la ejecución de la operación.

Tabla 8. Estructura TLV para petición de una operación de extensión

Tag	***Length***	***Value***															
		T	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**V**	**…**
		Identificador del *chip* de la tarjeta. *(IdCard)*			Identificador del *applet.* *(IdApplet)*			Identificador de la operación de extensión. *(IdOE)*			Parámetros de la operación de extensión. *(Param)*						

Las etiquetas, longitudes y descripción de cada uno de estos campos se detallan en la Tabla 9.

Tabla 9. Detalles de la estructura TLV para petición de una operación de extensión

Valores	**Longitud (bytes)**	**Descripción**
AEh	1	Etiqueta que se define para una operación de extensión
L.OE	Variable	Longitud de los datos de la operación de extensión
AEh	1	Etiqueta que se define para una operación de extensión
L.IdCard	1	Longitud del identificador del *chip* de la tarjeta
IdCard	4	Identificador del *chip* de la tarjeta
AEh	1	Etiqueta que se define para una operación de extensión

L.IdApplet	1	Longitud del identificador del *applet*
IdApplet	8	Identificador del *applet*
AEh	1	Etiqueta que se define para una operación de extensión
L.IdOE	1	Longitud del identificador de la operación de extensión
IdOE	1…2	Identificador de la operación de extensión en el servidor
AEh	1	Etiqueta que se define para una operación de extensión
L.Param	1	Longitud del parámetro de la operación de extensión
Param	Variable	Parámetro que requiere la operación de extensión

2.3.2. Componente Proxy

Este componente reside en el terminal de conjunto con los *middlewares* y sirve de intermediario de la comunicación entre todos los componentes del modelo de extensión. El Componente Proxy implementa la lógica de un lector de tarjetas, interceptando las transmisiones de los comandos y respuestas APDU que se envían entre un *middleware* y un *applet* determinado, así como las peticiones entre un *applet* y el Servicio de extensión, decidiendo en cada caso a quién enviar las respuestas recibidas. Este componente funciona como un *wrapper*[42] del 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.

Entre las principales responsabilidades que debe tener este componente se encuentran:

- Obtener los lectores disponibles para la conexión con la tarjeta inteligente.
- Realizar la conexión con un lector de tarjetas inteligentes.
- Realizar la desconexión de un lector de tarjetas inteligentes.
- Recibir un comando APDU de un *middleware*.
- Enviar un comando APDU a un *applet* en la tarjeta inteligente.
- Interpretar un comando APDU o una respuesta APDU recibida.

[42] Es referido a un programa que controla el acceso a un segundo programa.

- Enviar un comando APDU solicitando más datos de la petición de la operación de extensión.
- Enviar datos de la petición de extensión al Servicio de extensión.
- Enviar uno o varios comandos APDU con la respuesta de una operación de extensión a un *applet* en la tarjeta inteligente.
- Enviar una respuesta APDU a un *middleware*.

2.3.3. Servicio de extensión

Servicio que reside en un servidor web y es el encargado de recibir las peticiones de ejecución o almacenamiento fuera de la tarjeta inteligente. Se propone que sea un servicio genérico que invoque operaciones de extensión específicas, previamente definidas e implementadas, para minimizar el desarrollo para los programadores de aplicaciones para tarjetas inteligentes.

Entre las principales responsabilidades que debe tener este servicio se encuentran:

- Ejecutar una función o algoritmo determinado.
- Almacenar información asociada a una aplicación de una tarjeta inteligente determinada.
- Consultar información asociada a una aplicación de una tarjeta inteligente determinada.
- Enviar la respuesta de la ejecución de una operación de extensión.

2.3.4. Operación de extensión

Se define como una operación específica a ser ejecutada en el servidor por el Servicio de extensión. Estas operaciones de extensión deben ser implementadas y publicadas a priori; pueden ser de almacenamiento y/o consulta de información, o la ejecución de operaciones de alto costo computacional que sean más factible de realizar fuera de la tarjeta inteligente.

2.3.5. Repositorio de datos

En este componente se almacena la información asociada a una aplicación determinada de una tarjeta inteligente, para su posterior recuperación. Por lo básico de los tipos de datos que se manejan en las tarjetas inteligentes y su conversión en arreglos de bytes para su salida o entrada de esta, no es necesaria la definición de tipos de datos complejos, sino una estructura simple compuesta por propiedades (clave – valor) que permita almacenar variables y sus valores. Esta estructura debe estar agrupada por cada tarjeta, para lo cual se puede utilizar el identificador del *chip* de la tarjeta. También se recomienda que se agrupe por aplicaciones, utilizando para esto el identificador del *applet;* de esta manera no existe conflicto dado el caso de que dos aplicaciones definan variables con el mismo nombre. Se recomienda que la información se almacene cifrada en el repositorio de datos.

2.3.6. Administrador de seguridad

Al hacer uso de un KMS para todas las operaciones sensibles como el manejo de llaves, se considera este componente como un ambiente seguro. Debido a esto es el componente que se encarga de establecer un canal seguro de comunicación entre el servidor web y el API *Java Card* de Extensión en la tarjeta inteligente, para cuando se realicen peticiones que requieran la ejecución de operaciones en el servidor. El protocolo de autenticación mutua definido como parte de esta comunicación segura se presenta más adelante en este capítulo. Además tiene como responsabilidad el cifrado de la información que se almacene en el repositorio de datos, para lo cual se pueden utilizar las llaves que se encuentran en el KMS.

2.3.7. Sistema de administración de llaves

Componente que se encarga de la administración de manera segura de los objetos criptográficos y la generación de llaves. Se utiliza por su característica de proporcionar un canal seguro en la comunicación con dispositivos criptográficos u otros componentes.

2.4. Flujo de comunicación entre los componentes del modelo de extensión

En la Figura 12 se muestra de una manera simplificada el flujo de comunicación entre los componentes del modelo de extensión (sin tener en cuenta los mecanismos de seguridad). En este flujo de comunicación se representan a través de mensajes direccionales y actividades las responsabilidades fundamentales de los componentes en el proceso de ejecución de una operación de extensión fuera de la tarjeta inteligente.

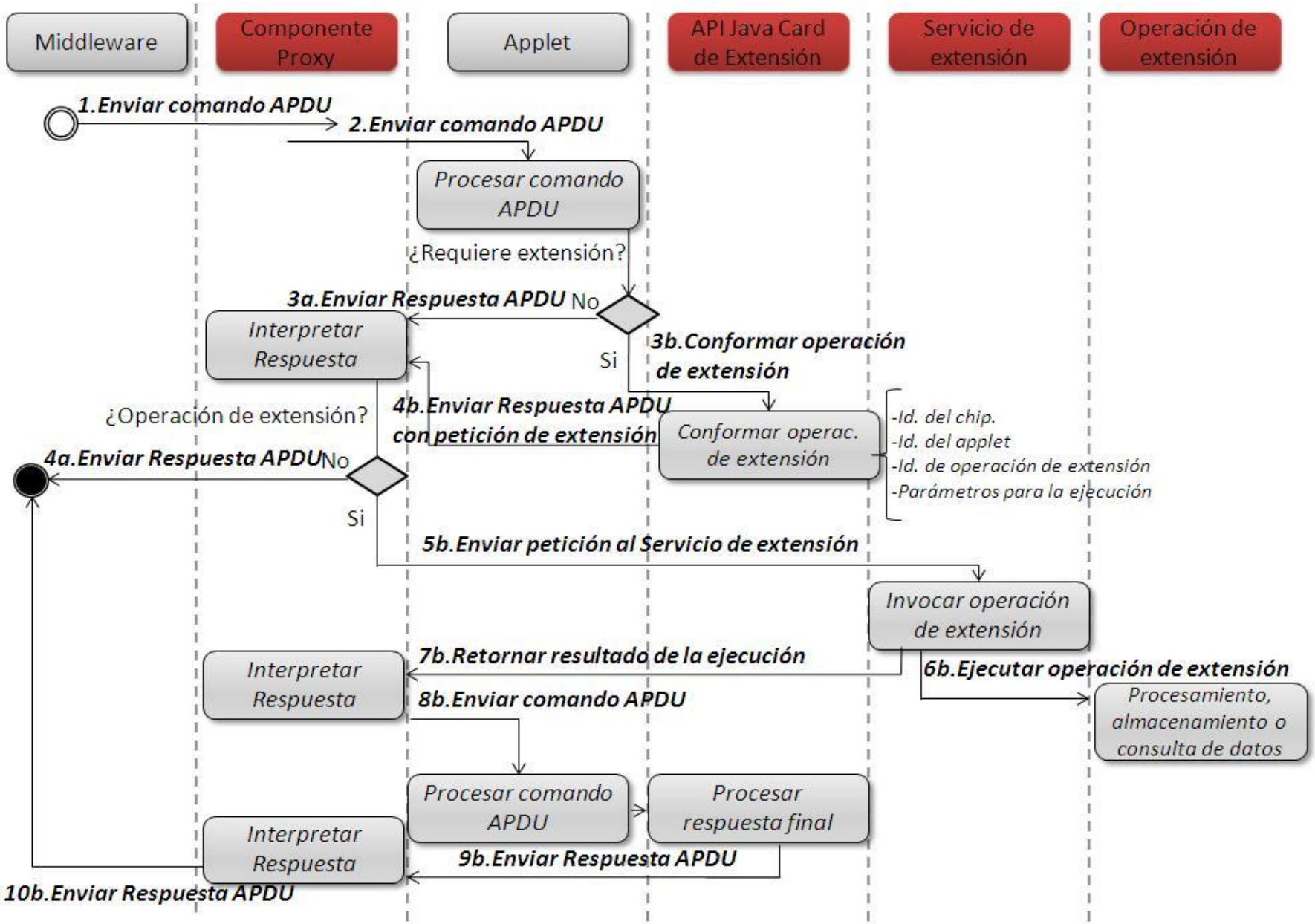

Figura 12. Flujo de comunicación entre los componentes del modelo de extensión.

Como elemento importante de este flujo de comunicación se resalta el hecho de que es desde la tarjeta inteligente que se toma la decisión de ejecutar determinada operación fuera de la misma, teniendo en cuenta que el modelo de extensión que se propone es orientado al desarrollador *Java Card* y que la tarjeta posee toda la información que se requiere para conformar una petición de extensión. Por ende, la decisión de almacenar datos o ejecutar un algoritmo determinado en el servidor web, es del *applet* en la tarjeta inteligente, siendo este diseño más seguro que el de delegar esta decisión al *middleware* o el Componente Proxy en el terminal.

El flujo de comunicación comienza desde el terminal, cuando un *middleware* inicia la comunicación enviando un comando APDU a la tarjeta inteligente, que es interceptado ahora por el Componente Proxy quien luego lo reenvía a la tarjeta. En la tarjeta inteligente 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 o una operación estándar. En cualquiera de los casos se responde al Componente Proxy con una respuesta APDU. Esta respuesta APDU es interceptada por el Componente Proxy e interpretada para saber cómo proceder. En caso de ser una respuesta APDU estándar se le reenvía esta respuesta al *middleware* y termina el flujo de comunicación. En caso de que la operación que se está solicitando se haya determinado que requiere extensión, el *applet,* apoyándose en las funcionalidades que brinda el API *Java Card* de Extensión, conforma una petición de extensión con todos los datos que se requieren para su ejecución (detallados en las Tablas 8 y 9). Cuando el Componente Proxy recibe e interpreta esta respuesta APDU, determina que se está solicitando realizar una operación de extensión y comprueba si recibió todos los datos de la petición; en caso de no ser así se solicitan los datos restantes. Luego de recibidos todos los datos se envían al Servicio de extensión que invoca una operación de extensión (según el identificador definido para esta operación), que debe haber sido implementada y publicada a priori.

Luego de ejecutada la operación de extensión se devuelve el resultado obtenido al Componente Proxy. Este componente envía a la tarjeta un comando APDU con la respuesta recibida de la operación de extensión, para su procesamiento final. El *applet* con el que se está realizando la comunicación recibe el comando APDU con la respuesta de la ejecución de la operación de extensión, y esta se procesa realizando cualquier otra operación que se necesite para dar una respuesta final al *middleware*. Luego de este procesamiento en la tarjeta inteligente, se devuelve una respuesta APDU al Componente Proxy para que sea reenviada al *middleware* con el resultado final de la operación de extensión, concluyendo el flujo de comunicación.

2.5. Mecanismos de seguridad del modelo de extensión

Como uno de los principios del modelo se definió que debía implementar estándares de seguridad que garantizaran un ambiente de ejecución seguro, fuera de la tarjeta inteligente. Como se presentaron con anterioridad, *GlobalPlatform* e ISO/IEC 7816 son los principales modelos de seguridad para tarjetas inteligentes existentes hoy en día. Para la comunicación segura entre un *middleware* y un *applet* determinado se propone utilizar (aunque de manera opcional), el canal seguro de comunicación de *GlobalPlatform*. Este modelo es una de las propuestas más utilizadas debido al API que provee *GlobalPlatform* que permite hacer uso de las funcionalidades, objetos de seguridad y protocolos de canal seguro, correspondientes al dominio de seguridad al que pertenece una aplicación.

Para los componentes que se introducen como parte del modelo de extensión es necesario de igual manera proveer una comunicación segura que proteja la información que a través de ellos sale fuera de la tarjeta, para su procesamiento o almacenamiento. Como variante se propone utilizar el canal seguro que propone el modelo de seguridad ISO/IEC 7816. En cuanto a la decisión de entre qué componentes del modelo implementar este canal seguro, se seleccionaron el API *Java Card* de Extensión (en la tarjeta inteligente) y el Administrador de seguridad (en el servidor web).

Los canales seguros de comunicación de *GlobalPlatform* e ISO/IEC 7816 aseguran tres niveles de seguridad fundamentales (Global Platform, 2003) (ISO/IEC, 2005a):

- Autenticación mutua: en que la tarjeta y el terminal prueban su autenticidad demostrando que tienen conocimientos sobre un mismo secreto.
- Integridad y autenticación del origen de datos: en la que la tarjeta se asegura de que los datos recibidos desde el terminal realmente vienen de una entidad autenticada, en la secuencia correcta y no han sido alterados.
- Confidencialidad: en la que los datos que se transmiten desde el terminal a la tarjeta, no son visibles por una entidad no autorizada.

En los epígrafes siguientes se profundiza en la primera fase de la comunicación segura del modelo de extensión, describiendo las propuestas de los protocolos de autenticación mutua.

2.5.1. Protocolo de autenticación mutua para la comunicación *middleware –applet*

Debido a que el modelo *GlobalPlatform* se encuentra entre los más utilizados para una comunicación segura *middleware – applet,* el SCP01 es la propuesta seleccionada para la comunicación segura entre estos componentes del modelo. La decisión de implementar este canal seguro es opcional, el desarrollador puede decidir no utilizar un canal de seguro de comunicación entre el *middleware* y su *applet correspondiente*, aunque se recomienda su uso. Si por los requerimientos de seguridad de la aplicación se decide implementar un canal seguro entre el *middleware* y el *applet*, la autenticación mutua entre estos componentes (Figura 13) constituye la primera fase de una comunicación segura.

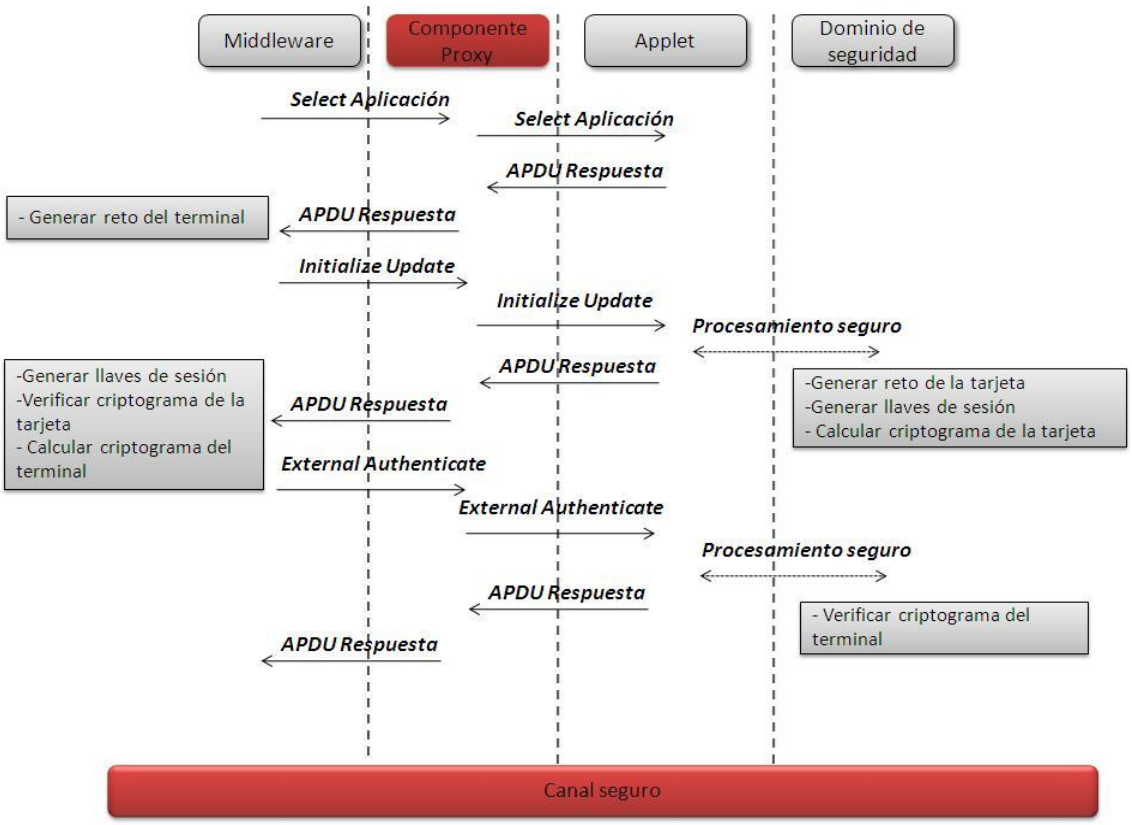

Figura 13. Flujo de autenticación mutua *GlobalPlatform* del SCP01 (*middleware – applet*)

Como se explicó con anterioridad, dentro de los mecanismos que se utilizan se encuentra el de Reto-Respuesta, mediante el cual las partes intercambian información generada de forma aleatoria y mediante la conformación de criptogramas, utilizando las llaves o secretos que ambos conocen, son intercambiados para comprobar que el reto enviando por el terminal coincide con el respondido por la tarjeta y viceversa. En el SCP01 donde se define la autenticación mutua simétrica *GlobalPlatform*, se utilizan los comandos *Initialize Update* y *External Authenticate*. El flujo de comunicación ocurre de

igual manera a la que se describe en (*epígrafe 1.5.1.1- Protocolo de autenticación mutua GlobalPlatform*), con la variación de que todos los comandos y respuestas APDU son interceptados ahora por el Componente Proxy. En la Tabla 10 se describen de forma sintética los pasos de la autenticación mutua entre el *middleware* que se encuentra en el terminal y un *applet* en la tarjeta. No se incluye en el flujo los pasos en los que interviene el Componente Proxy debido a que su responsabilidad es solo de intermediario, redirigiendo los mensajes que le llegan.

Tabla 10. Descripción de la autenticación mutua simétrica *GlobalPlatform middleware-applet*

Middleware	***Applet***
1. Envía un comando *Select* para seleccionar el *applet* con el que se quiere establecer el canal seguro de comunicación. Para esto es necesario conocer el identificador del *applet*.	2. Se selecciona el *applet* con el que se quiere establecer el canal seguro y se envía la respuesta del comando.
3. Genera un reto para enviar a la tarjeta (arreglo de 8 bytes de longitud y en su interior números aleatorios). 4. Envía el comando *Initialize Update* donde se envía el reto generado por el terminal.	5. Genera un reto para enviar al terminal (arreglo de 8 bytes de longitud y en su interior números aleatorios). 6. Genera las llaves de sesión para el canal seguro (usando el reto del terminal, el reto de la tarjeta y las llaves estáticas S-MAC y S-ENC). 7. Genera el criptograma de la tarjeta. 8. Envía la respuesta del comando con la siguiente información: Datos de derivación de llave (se utilizan para obtener las llaves estáticas de la tarjeta) (10 bytes). Información de la llave (número de versión de la llave, identificador del protocolo de canal seguro (en este caso el 01)) (2 bytes). El reto generado por la tarjeta (8 bytes). El criptograma de la tarjeta que es un criptograma de autenticación (8 bytes).
9. Genera las llaves estáticas (se utiliza la llave madre y los datos de derivación de llave).	14. Verifica el criptograma del terminal.

10. Genera las llaves de sesión. 11. Verifica el criptograma de la tarjeta. 12. Genera el criptograma del terminal. 13. Envía comando *External Authenticate* que es utilizado por la tarjeta para autenticar el terminal y para determinar el nivel de seguridad requerido para todos los comandos posteriores.	Comienza la comunicación segura, dada por el nivel de seguridad que se especificó en el P1 del comando *External Authenticate.*
Si el flujo de comunicación es exitoso, se ha podido establecer el canal seguro.	

Luego de la autenticación mutua, para el resto de los comandos que se intercambien entre el *middleware* y su *applet* dentro de la sesión del canal seguro, se utilizará el nivel de seguridad que se haya establecido durante la negociación del canal de comunicación (en el comando *External Authenticate*).

2.5.2. Protocolo de autenticación mutua API *Java Card* de Extensión – Administrador de seguridad

Además de la comunicación segura entre un *middleware* y su *applet*, es fundamental la comunicación segura entre los componentes que se introducen como parte del modelo de extensión. Para esta comunicación segura no es conveniente la utilización, de igual manera que entre el *middleware* y el *applet*, del canal seguro de *GlobalPlatform*. Entre algunas de las razones se encuentra que no se tiene acceso por los componentes del modelo a las llaves para el canal seguro que se encuentran en el *middleware,* además de que es deseada la utilización de otras llaves y que no se interfiera en el canal seguro entre el *middleware* y el *applet*.

Como variante se encuentra el canal seguro que propone el modelo ISO/IEC 7816. Para la implementación del canal seguro del modelo de extensión se seleccionaron el API *Java Card* de Extensión (en la tarjeta) y el Administrador de seguridad (en el servidor web). Esta decisión está basada en el hecho de que ambos componentes se consideran ambientes seguros: el API por encontrarse dentro de la tarjeta y el Administrador de seguridad por utilizar un KMS para todas las operaciones sensibles como el manejo de las llaves. En la Figura 14 se representa el flujo autenticación mutua simétrica ISO/IEC 7816 entre el API *Java Card* de Extensión y el Administrador de seguridad.

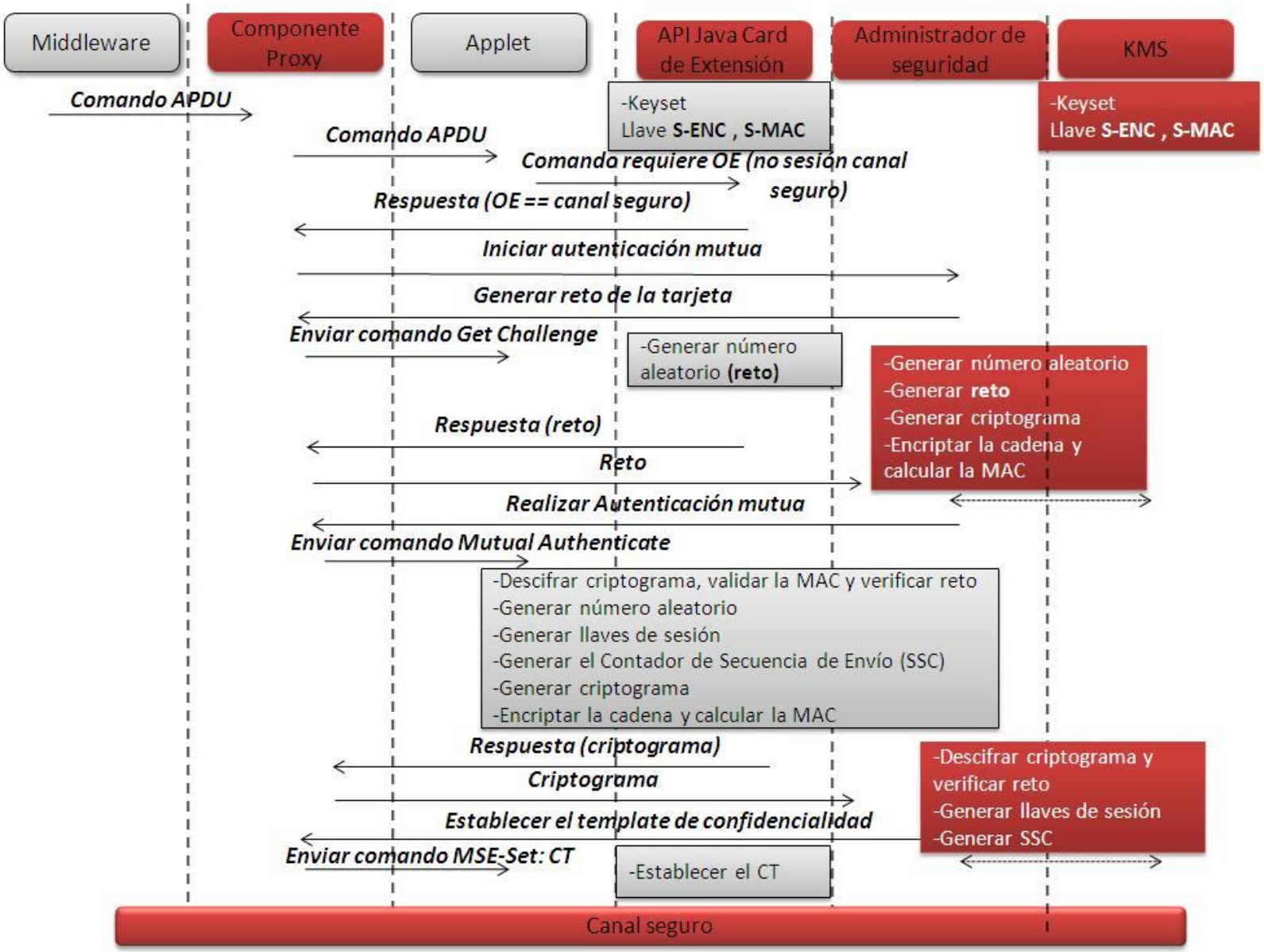

Figura 14. Flujo de autenticación mutua simétrica ISO/IEC 7816 (API *Java Card* de Extensión – Administrador de seguridad)

A diferencia del canal seguro entre el *middleware y* el *applet*, este canal seguro entre los componentes del modelo de extensión es imprescindible para la seguridad de la información que se procesa o almacena fuera de la tarjeta inteligente.

Como se aprecia en la Figura 14, los comandos APDU que se utilizan en el protocolo de autenticación mutua del canal seguro ISO/IEC 7816 son el *Get Challenge*, *Mutual Authenticate* y *Manage Security Enviroment* (MSE). Además del *middleware* y el *applet* se introducen los componentes del modelo que participan en esta autenticación mutua: el Componente Proxy, el API *Java Card* de Extensión, el Administrador de seguridad y el KMS. El Componente Proxy mantiene su comportamiento de intermediario de la comunicación entre los componentes. El API *Java Card* de Extensión y el KMS son los entornos seguros que comparten las llaves para realizar la autenticación mutua ISO/IEC 7816. El KMS almacena de manera segura las llaves estáticas S-ENC y S-MAC y brinda una interfaz para que le sea posible al Administrador de seguridad acceder a ellas.

El flujo de comunicación inicia cuando se envía un comando a la tarjeta que requiere una operación de extensión y aún no hay una sesión abierta del canal seguro. Los pasos más importantes de esta autenticación mutua, se describen a continuación en la Tabla 11.

Tabla 11.Descripción de la autenticación mutua simétrica ISO/IEC 7816 tarjeta-servidor

Administrador de seguridad	**API *Java Card* de Extensión**
1. Envía un comando *Get Challenge* solicitando un número aleatorio que servirá de reto, para que la tarjeta verifique al servidor.	2. Genera el número aleatorio (reto) y lo devuelve en la respuesta del comando *Get Challenge*.
3. Genera un número aleatorio. 4. Genera el reto del servidor 5. Genera un criptograma concatenando ambos (número aleatorio y reto del servidor) 6. Encripta la cadena cifrando estos datos con la llave estática S-ENC y le calcula un MAC con la llave estática S-MAC. 7. Envía el comando *Mutual Authenticate* con el criptograma.	8. Descifra el criptograma utilizando la misma llave estática S-ENC. 9. Valida la MAC y verifica que el reto enviado anteriormente coincide con el que recibió del servidor. 10. Genera un número aleatorio. 11. Genera las llaves de sesión S-ENC y S-MAC. 12. Genera criptograma con el mismo formato que el recibido del servidor.

	13. Genera el contador de secuencia de envío. 14. Envía el criptograma en la respuesta del comando al servidor.
15. Descifra el criptograma, obtiene el reto enviado anteriormente y lo valida. 16. Genera las llaves de sesión S-ENC y SMAC, que deben coincidir con las de la tarjeta. 17. Genera el contador de secuencia de envío. 18. Envía un comando *MSE-Set* para establecer una plantilla de confidencialidad, indicando a la tarjeta que desde ese momento se establece un canal seguro con cifrado y MAC de los comandos y sus respuestas.	19. Establece la plantilla de confidencialidad y comienza la comunicación segura entre los componentes del modelo de extensión.
Si el flujo de comunicación es exitoso, se ha podido establecer el canal seguro ISO/IEC 7816.	

Luego de realizada satisfactoriamente la autenticación mutua simétrica ISO/IEC 7816, durante la etapa de mensajería segura, todos los comandos y respuestas que se intercambian entre el servidor web y la tarjeta inteligente deben ir con cifrado y MAC, como se especificó durante la negociación del canal de comunicación. Con este nivel de seguridad se garantiza la integridad y confidencialidad de la información que sale de la tarjeta, para ser procesada o almacenada fuera de esta, en todo el flujo de comunicación.

Conclusiones del capítulo

En este capítulo se presentó un modelo para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes. Este modelo cubre las insuficiencias que fueron detectadas en la revisión de la literatura sobre el tema, obteniéndose un modelo que tiene como base los principales estándares y tecnologías existentes para tarjetas inteligentes. La propuesta se basa en la utilización de los recursos de *hardware* de un ordenador, con mayores prestaciones que las que posee una tarjeta inteligente estándar, permitiendo:

- Almacenar datos asociados a aplicaciones *Java Card* fuera de la tarjeta inteligente, en un repositorio de datos, de manera que se extienda la capacidad

de memoria de estos dispositivos y sea recuperable su información ante una pérdida o deterioro.

- La ejecución de algoritmos de alto costo computacional que por el tiempo de ejecución y/o complejidad sea más factible realizar fuera de la tarjeta.

El modelo de extensión además proporciona mecanismos de seguridad basados en los canales seguros de comunicación que proponen *GlobalPlatform* e ISO/IEC 7816. Con la implementación de estos canales seguros de comunicación se garantizan tres niveles de seguridad fundamentales: la autenticación entre las entidades del modelo mediante un protocolo de autenticación mutua en que dos entidades demuestran tener conocimiento sobre un mismo secreto; la integridad de la información asegurando que viene en de una entidad autenticada, en la secuencia correcta y no ha sido alterada; y la confidencialidad ya que al viajar cifrada no es posible ver los datos que se transmiten por entidades no autorizadas.

Capítulo 3. Plataforma de desarrollo y ejecución de operaciones de extensión para aplicaciones *Java Card* en tarjetas inteligentes

Como validación práctica del modelo de extensión propuesto se desarrolló una plataforma que permite la implementación y ejecución de operaciones para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes. En este capítulo se describe el ambiente de desarrollo seleccionado, la arquitectura de software, así como elementos de su implementación y despliegue. Para terminar el capítulo se muestran resultados de operaciones implementadas sobre la plataforma de software, en diferentes escenarios, que demuestran la factibilidad del modelo de extensión propuesto.

3.1. Características generales de la Plataforma

La plataforma fue diseñada teniendo en cuenta el modelo para la extensión de las capacidades de procesamiento y memoria de las tarjetas inteligentes *Java Card* presentado como resultado de esta investigación. Los componentes que se proponen como parte del modelo se concretan en los siguientes componentes de software:

El componente ***Java Card Applet Extension Client*** **(JCAEC):** Implementa el API *Java Card* de Extensión del modelo. 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. Debe ir desplegado en la tarjeta inteligente, de conjunto con un *applet*.

El componente ***Java Card Applet Extension Proxy*** **(JCAEP)**: Implementa el Componente Proxy del modelo de extensión, encargado de comunicarse en tiempo de ejecución con el *applet* mediante el lector de tarjetas y con el servicio publicado en el servidor web mediante un mensaje *SOAP*[43]. Su principal responsabilidad es servir de intermediario para dirigir las transmisiones de comandos y respuestas APDU entre los componentes de la plataforma.

[43] Protocolo simple de acceso a objetos (traducido de las siglas en inglés SOAP: *Simple object access protocol*).

El componente ***Java Card Applet Extension Server*** **(JCAES)**: Integra la implementación de varios componentes del modelo de extensión como el Servicio de extensión, las Operaciones de extensión, el Administrador de seguridad y el Repositorio de datos. La lógica de estos componentes del modelo se implementa en lo siguientes componentes de software:

Extension Service: Es un servicio web SOAP genérico, que se utiliza para la ejecución de una operación de extensión publicada en el servidor.

Server Components**:** Provee un conjunto de clases para funcionalidades auxiliares que utiliza el *Extension Service* durante la ejecución de una operación de extensión. Entre estas funcionalidades se encuentra el acceso a la base de datos para las operaciones de almacenamiento o consulta de datos, así como la implementación de la carga dinámica de las operaciones de extensión previamente implementadas y publicadas en el servidor.

Extension Operation**:** Operaciones que implementan una lógica determinada y deben ser implementadas y publicadas en el servidor web a priori, para que sea posible su invocación.

Security Manager: Se encarga de la implementación del canal seguro de comunicación entre la tarjeta y el servidor.

Base de Datos**:** Estructura de datos para el almacenamiento de información asociada a una aplicación de una tarjeta dada.

3.2. Ambiente de desarrollo

Para definir las herramientas y tecnologías a utilizar durante la construcción de la plataforma se tuvieron en cuenta los tres ambientes donde se debía desarrollar: el terminal, la tarjeta inteligente y el servidor web. En aras de lograr una homogeneidad entre los productos que se desarrollan en el Departamento de Tarjetas Inteligentes del Centro de Identificación y Seguridad Digital (CISED), se hizo la siguiente selección:

Se seleccionó la metodología de desarrollo ágil XP para guiar el proceso de desarrollo. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Well, 2013).

Para la implementación del lado de la tarjeta se utilizó *Java Card* 2.1.1. Se seleccionó esta versión debido fundamentalmente a que es la que poseen las tarjetas inteligentes con que se trabaja en el CISED. Como entorno de desarrollo integrado (IDE) para *Java Card* se utilizó el *Developer Suite*, extensión del Proyecto Eclipse desarrollado por (Gemalto, 2013) y que proporciona un conjunto de herramientas para crear y depurar *applets Java Card*. Este IDE brinda un ambiente favorable para el diseño y la implementación de *applets* y posibilita simular las funcionalidades de los *applets* antes de ser instalados en las tarjetas inteligentes.

Para la implementación del lado del terminal se seleccionó como lenguaje C# y entorno de desarrollo el Visual Studio. NET 2005. Además de las facilidades que representan las tecnologías de .NET se seleccionaron porque era necesaria la integración con el *SmartCard Framework* (Ramírez & Legón, 2011) que está implementado sobre estas tecnologías. Este *framework* fue desarrollado en el Departamento Tarjetas Inteligentes y permite el desarrollo de *middlewares* y aplicaciones clientes sobre la tecnología Microsoft .Net Framework o Mono .Net Framework, que permitan la interacción con tarjetas inteligentes, sus aplicaciones y los lectores para establecer la comunicación con estas. El *framework* facilita en gran medida el trabajo ya que implementa varios estándares para la comunicación con tarjetas inteligentes y sus aplicaciones, entre los que se destacan: PC/SC, ISO/IEC 7816-4 y *GlobalPlatform*.

Para la implementación del lado del servidor se seleccionaron tecnologías libres. Se utilizó el Apache Tomcat 7.0.23 que es un servidor web multiplataforma que funciona como contenedor de *servlets*. Como IDE se utilizó el NetBeans 7.4 y Axis 2 como *framework* que permite generar y desplegar aplicaciones de servicios web en *Java* (Apache Software Foundation, 2012). Como sistema gestor de base de datos se seleccionó PostgreSQL 9.2-1002 debido a que se considera uno de los sistemas de gestión de bases de datos de código abierto más potente del mercado (PostgreSQL Development Group, 2014). Para la implementación de operaciones de extensión criptográficas sobre la plataforma de desarrollo, se utilizó la biblioteca para *Java BouncyCastle* (Bouncy Castle Inc, 2013). *BouncyCastle* contiene la implementación de los principales algoritmos de criptografía simétrica y asimétrica utilizados en la actualidad.

3.3. Arquitectura de software de la Plataforma

La arquitectura de la plataforma de extensión en su vista más abstracta es un modelo Cliente- Servidor. Este modelo está basado en el principio fundamental de que un cliente

realiza peticiones a otro programa, el servidor, que le da respuesta. Para la solución propuesta el componente *Java Card Applet Extension Proxy* que se encuentra en el terminal se comporta como cliente, manejando todas las peticiones que le llegan tanto de la tarjeta inteligente (con el componente *Java Card Applet Extension Client*) como del servidor web (con el *Extension Service*). Una representación de la arquitectura se puede ver en la Figura 15.

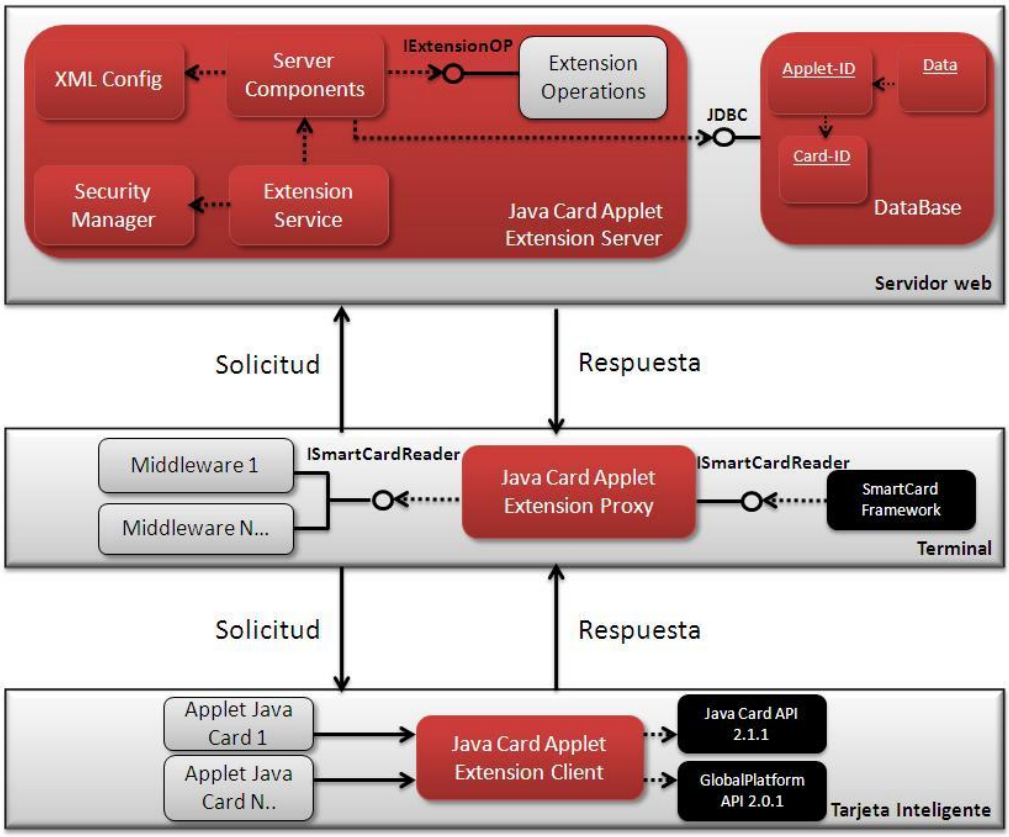

Figura 15. Arquitectura de la Plataforma de desarrollo para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en Tarjetas Inteligentes

Se destacan de color rojo los componentes que provee la plataforma para el desarrollo y ejecución de aplicaciones que requieran extender las capacidades de procesamiento y memoria de una tarjeta inteligente; y en color gris aquellos componentes que deben ser creados por el desarrollador, y se integran a la plataforma en una solución para brindar un servicio final a un usuario. Los componentes que deben ser creados de manera independiente son el *middleware* y su *applet*, de igual manera que en el desarrollo tradicional, y la operación de extensión con la lógica que se quiera ejecutar fuera de la tarjeta inteligente (*Extension Operation*).

En la plataforma también puede verse el uso de una arquitectura basada en componentes, donde se definen componentes funcionales o lógicos que exponen interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor y se desacoplan las responsabilidades de todos los componentes, lográndose simplicidad en la ejecución de las operaciones de extensión para *applets Java Card*, que por sus características requieran mayor capacidad de memoria o procesamiento de la que la tarjeta es capaz de proveer.

3.4. Diagrama de Componentes de la Plataforma

En los diagramas de componentes se muestran los elementos de diseño e implementación de un sistema de software. A continuación se muestran los componentes de software implementados para la plataforma de extensión.

Para el componente *Java Card Applet Extension Client* (Figura 16) se implementó un API con el mismo nombre. El componente *CustomApplet* hace referencia a un *applet* determinado que hace uso de este API para el procesamiento de las operaciones de extensión. El paquete *Java Card Framework* 2.1.1 es utilizado ya que posee el conjunto de clases provistas por *Java Card* para el desarrollo de aplicaciones para tarjetas inteligentes. Por su parte, el API *GlobalPlatform* 2.0.1 permite el acceso a las funciones del *CardManager* (*applet* que representa el proveedor de la tarjeta) de la tarjeta inteligente.

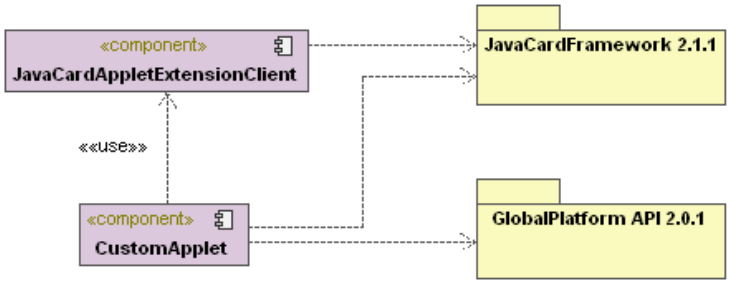

Figura 16. Diagrama de componentes del *Java Card Applet Extension Client*

Para el componente *Java Card Applet Extension Proxy* (Figura 17) de igual manera se implementó un componente encargado de toda la comunicación tanto con el *applet* mediante el lector, como con el servicio web en el servidor de extensiones. El *Java Card Applet Extension Proxy* implementa la interfaz *ISmartCardReader* del *SmartCard Framework*, con la idea de que cualquier *middleware* lo pueda usar como lector de tarjetas, y a la vez implementa toda la lógica de la comunicación con un lector de tarjetas

inteligentes genérico. De esta manera este componente va a funcionar como un lector (*reader*) e interceptar todos los comandos APDU que se envíen desde el *middleware* para la comunicación con la tarjeta inteligente. El componente *CustomMiddleware* representa a un *middleware* determinado y es desarrollado de manera independiente como una solución *applet-middleware* que puede integrarse con el JCAEP para la ejecución de operaciones fuera de una tarjeta inteligente.

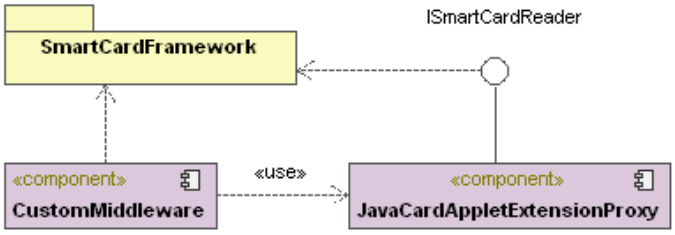

Figura 17. Diagrama de componentes del *Java Card Applet Extension Proxy*

Para el componente *Java Card Applet Extension Server* (Figura 18) son varios los componentes implementados como parte de la plataforma. El componente *ExtensionService* es el servicio web que se encuentra publicado en el servidor TomCat y utiliza un conjunto de utilidades que brinda *ServerComponent*s para la ejecución de las operaciones de extensión. Entre estas utilidades se encuentran clases para el manejo del acceso a datos, la carga dinámica de operaciones de extensión y una interfaz llamada *IExtensionOPServer* que define el comportamiento que deben cumplir todas las implementaciones de las operaciones de extensión.

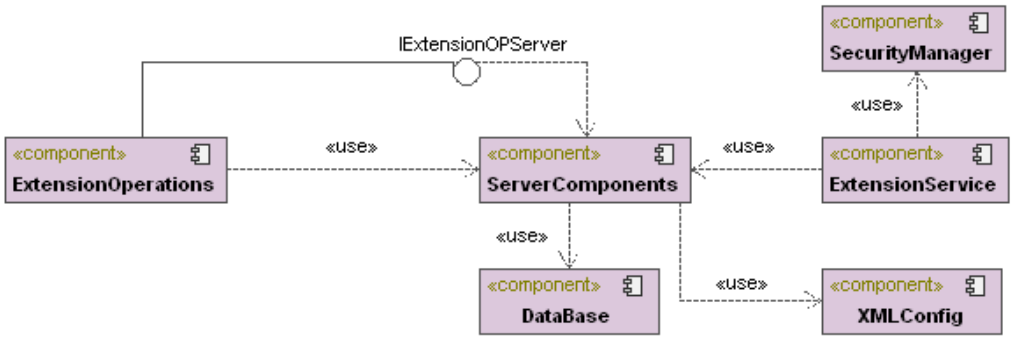

Figura 18. Diagrama de componentes del *Java Card Applet Extension Server*

La base de datos (*DataBase*) almacena información asociada a los *applets* de las tarjetas inteligentes. En el fichero de configuración (*XMLConfig*) se almacena el nombre de las clases y la dirección de los archivos *.jar* que contienen las implementaciones de

las operaciones de extensión (*ExtensionOperations*), para ser invocadas dinámicamente. Estas operaciones de extensión no forman parte de la solución que provee la plataforma, sino que deben implementarse para operaciones específicas (excepto las operaciones de consulta y almacenamiento de información que se provee una implementación genérica). La ventaja de esta implementación del servidor radica en el hecho de que una vez que esté desplegada la plataforma en determinado entorno, la posibilidad de agregar nuevas operaciones de extensión o modificar las ya existentes es tan sencilla como copiar el archivo *.jar* donde están implementadas y actualizar el archivo de configuración (*XML Config*). De esta manera la implementación y actualización de las operaciones de extensión prácticamente no tiene impacto en la plataforma ya desplegada.

3.5. Despliegue de la Plataforma

Para mostrar la distribución física de los componentes de software implementados como parte de la plataforma se muestra el diagrama de despliegue de la Figura 19. Se representan además todos los componentes que se integran a la plataforma para brindar una solución completa usando tarjetas inteligentes. Se utilizan tres nodos de ejecución en ambientes diferentes: la tarjeta inteligente, un ordenador que funciona como un terminal de servicio y un servidor *Java* Apache TomCat con Axis 2, que expone un servicio web SOAP.

En la tarjeta debe encontrarse instalado el JCAEC de conjunto con los *applets* que requieran su uso. En el terminal se debe encontrar el JCAEP y los *middlewares* para el acceso a los *applets* que se encuentran en la tarjeta inteligente. Por último, en el servidor web van publicados los componentes del JCAES, que se exponen como un servicio web. Los componentes en cada uno de estos nodos se muestran en la Figura 19 y fueron descritos en el epígrafe anterior. Por la simplicidad de los datos que se manejan en la base de datos se propone que esta se encuentre alojada en el mismo servidor web.

La comunicación entre el lector de tarjetas y un *applet* en la tarjeta inteligente se realiza mediante el envío de comandos APDU. Los lectores se conectan a cada una de las terminales de servicio mediante el puerto USB y utilizan el protocolo PC/SC para la comunicación con estos ordenadores. Para la comunicación con el servicio web desplegado en el servidor TomCat se utiliza el protocolo SOAP.

Las tarjetas inteligentes deben poseer un mínimo de 5 kB de memoria libre en EEPROM (para la instalación del *Java Card Applet Extension Client y el applet*). Además se

requiere que cuenten con el sistema operativo *Java Card* 2.1.1 o una versión superior, así como *GlobalPlatform* 2.0.1 o superior. El terminal de servicio debe contar con un lector de tarjetas USB con los drivers PC/SC instalados. En cuanto a los requerimientos de software se requiere tener instalado Microsoft .Net Framework 1.1 o superior, o Mono .Net Framework 2.0 o superior. Además es necesario tener acceso al servidor web donde se encuentran publicado el servicio de extensión.

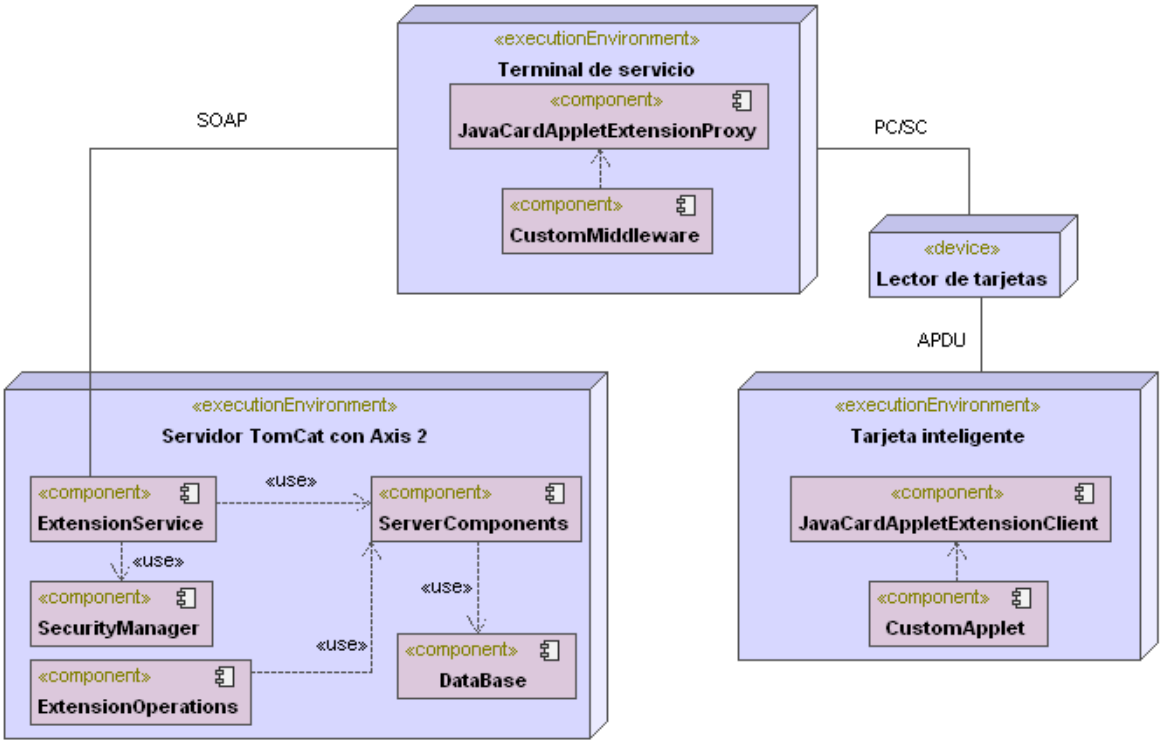

Figura 19. Diagrama de despliegue de la Plataforma

Los requerimientos mínimos de *hardware* recomendados para el servidor son un procesador Intel Pentium 4 a 2.80 GHz y 1 Gb de memoria RAM. Los requerimientos de software son los siguientes:

- Sistema operativo GNU/Linux *kernel* 2.6.x o superior o Microsoft Windows XP Server 2003/Vista/Server 2008/7/8/Server 2012.
- Plataforma *Java* 6.0 o superior.
- Servidor Apache TomCat 7.0.23 o superior.
- *Framework* para servicios web Axis 2 1.6.1 o superior.

- Servidor de bases de datos PostgreSQL 9.2-1002.

3.6. Pruebas de software realizadas a la Plataforma

Con el objetivo de comprobar el funcionamiento correcto de la plataforma implementada se realizaron pruebas de software. Entre las pruebas realizadas se encuentran las de unidad, de integración y de sistema. El objetivo fundamental de las pruebas de unidad es aislar cada parte del programa y probar que cada una de las partes individuales funciona de forma correcta (Beck, 2003). Las pruebas de integración son aquellas en las cuales los distintos módulos que conforman el software se combinan y son probados en grupos. Las pruebas de integración se realizaron después de las pruebas de unidad y antes de las pruebas de sistema como se recomienda en la literatura (Lewis, 2004). Las pruebas de sistema se realizaron teniendo en cuenta la plataforma de forma integral, para evaluar si cumplía con todos los requerimientos especificados.

Las pruebas de unidad se definieron y realizaron basándose primeramente, en los requerimientos definidos para la plataforma, así como en los errores detectados durante el desarrollo, tal y como expone la metodología XP. Estas pruebas se fueron desarrollando de forma paralela a la implementación de los componentes de la plataforma para asegurar que las principales funcionalidades se comportaban de la forma esperada. Entre estas se encuentran las funcionalidades definidas para conformar una petición de operación de extensión (en el JCAEC), la interpretación de comandos APDU (en el JCAEP) y la ejecución de una operación de extensión (en el JCAES).

De conjunto con los componentes de la plataforma se implementaron los componentes que se integran a esta para una solución completa. Estos son un *applet* con su *middleware* correspondiente y las operaciones de extensión específicas. Las pruebas de integración se organizaron en dos partes. Primeramente se probó el funcionamiento del envío y respuesta de los comandos APDU entre el *middleware*, el componente JCAEP, el *applet* y el componente JCAEC. De esta manera se comprobó que el JCAEP funcionaba correctamente de intermediario entre estos componentes de la plataforma, interpretando los comandos recibidos y enviándolos al componente correspondiente. En la segunda parte se probó el funcionamiento de los componentes antes mencionados, integrándose los componentes desplegados en el JCAES, verificándose la correcta ejecución del servicio web para una operación de extensión determinada, implementada a priori. Además se probó la persistencia fuera de la tarjeta de variables con sus valores, y su recuperación con posterioridad.

Finalmente se sometió la plataforma a pruebas de sistema, con todos los componentes integrados, con el objetivo de comprobar su comportamiento en un entorno real cercano a un entorno en el que pueda desplegarse, para detectar posibles dificultades. Con este fin se implementó una aplicación visual sobre el *middleware,* que permitiera enviar comandos APDU a una tarjeta y recibir las respuesta. Además, como parte de estas pruebas se realizaron pruebas de seguridad, para verificar el funcionamiento del canal seguro implementado, y el envío de los comandos con MAC y cifrado sobre este canal. En todas las pruebas realizadas la plataforma debía demostrar su capacidad de procesar los comandos recibidos y de ejecutar operaciones de extensión para los comandos que así lo requirieran, lo cual realizó de forma satisfactoria.

3.7. Implementación de operaciones de extensión sobre la Plataforma

Además de las pruebas de software realizadas, con el objetivo de demostrar la efectividad del modelo de extensión propuesto, se realizó la implementación de operaciones de extensión sobre la plataforma, que permitieron valorar la extensión de las capacidades de procesamiento y memoria de tarjetas inteligentes *Java Card* en diferentes escenarios. Con este propósito se utilizaron tarjetas inteligentes del modelo *Gemplus GemXpresso Pro R3.2 E32 PK* (Gemalto, 2004). Estas tarjetas poseen la máquina virtual y el API *Java Card* en su versión 2.1.1. Además tienen soporte para los algoritmos criptográficos DES, 3DES, SHA[44], MD5[45], RSA 1024, RSA 2048, DAP[46]. No soportan el tipo de dato *integer* de 32 bits y los recursos de *hardware* que poseen son los siguientes:

- Tamaño del *buffer*: 256 bytes (+ 5-byte de encabezado del comando)
- Tamaño de pila persistente (ROM): 29,6 kB.
- Tamaño de pila transitoria (RAM) disponible para *applets*: 1,2 kB.
- EEPROM disponible: 29,7 kB

Para el servidor web se utilizó un ordenador Dell con un procesador Intel Core 2 Quad CPU Q9400 y 2GB de memoria RAM. En este mismo servidor se encuentra instalado el

[44] *Secure Hash Algoritm.*

[45] *Message-digest Algorithm*

[46] *Data Authentication Pattern.*

servidor de bases de datos. Todos los experimentos se realizaron en un ambiente controlado con buen enlace de red, para que no fuese un costo adicional elevado la comunicación con el servidor web.

3.7.1. Escenario 1. Cálculo del factorial de un número entero positivo

Como primer escenario de prueba se seleccionó una operación aritmética simple como lo es el factorial de un número entero positivo. La operación factorial aparece en muchas áreas de las matemáticas, particularmente en combinatoria y análisis matemático, además de que es un algoritmo sencillo de implementar ya sea con un método iterativo o recursivo. Las limitaciones de esta operación para su implementación en *Java Card* usando las tarjetas GemXpresso Pro R3.2 E32 PK, radican, en primer lugar, en que no es posible calcular un factorial que sea expresable en un número mayor a un entero de 16-bits (*short*), debido a que este el tipo de dato mayor que soporta la tarjeta. Además de la limitación de los tipos de datos y la memoria que estos ocupan, se encuentra también la limitación asociada al costo computacional e incluso de memoria volátil (RAM) dentro de la tarjeta, debido a que el factorial de un número crece exponencialmente muy rápido.

Con la ejecución de la operación factorial en el servidor, es posible obtener el factorial de números expresables en 32 bits y 64 bits. En la Tabla 12 se muestran los resultados obtenidos en pruebas donde se hizo el cálculo del factorial tanto en la tarjeta inteligente como en el servidor. Como es posible observar, en el caso de las operaciones que se pueden ejecutar tanto en la tarjeta como en el servidor, el tiempo de respuesta de las operaciones en la tarjeta es significativamente menor. Este resultado se debe fundamentalmente al tiempo asociado al establecimiento de una conexión con el servidor web, y a que el flujo de comunicación se complejiza, teniendo que regresar este resultado a la tarjeta, para finalmente ser devuelto al *middleware* en el terminal.

Tabla 12. Resultados obtenidos en el cálculo del factorial

Operación	**Tiempo de operación en el servidor**	**Tiempo de operación en la tarjeta**	**Cantidad de instrucciones en la tarjeta**	**Cantidad de instrucciones en el servidor**
Factorial (5)	949 ms	21 ms	9 instrucciones	14 instrucciones
Factorial (10)	969 ms	-		
Factorial (20)	1 s 4 ms	-		

Respecto al número de instrucciones que se requieren para esta ejecución fuera de la tarjeta, usando la plataforma, en la Tabla 12 se muestra también que no varía mucho con respecto a las que requiere la tarjeta de manera tradicional. Este resultado se debe a que los componentes implementados como parte de la plataforma fueron diseñados teniendo en cuenta que se minimizase el esfuerzo que implicaba el procesamiento fuera de la tarjeta inteligente. Esta es una fortaleza de la plataforma, debido a que es importante para los desarrolladores que sea mínimo el impacto que genera el desarrollo de nuevas aplicaciones que requieran operaciones de extensión.

3.7.2. Escenario 2. Generación de llaves usando Curvas Elípticas con *Diffie-Hellman* y RSA

Entre las aplicaciones de las tarjetas inteligentes se encuentra las relacionadas con los documentos electrónicos de viaje. Estos documentos han sido estandarizados por ICAO[47] en varios documentos (ICAO, 2006a, 2006b) y se encuentran aprobados como normas por la ISO. ICAO ha definido una serie de medidas de seguridad para estos documentos electrónicos de viaje, entre las que se encuentra el Control de Acceso Extendido (EAC por sus siglas en inglés). El EAC es el responsable por la protección de los datos biométricos, llamados datos sensibles, a los cuales solo pueden tener acceso los sistemas de inspección que hayan sido autorizados por el emisor del pasaporte electrónico (Federal Office for Information Security, 2007).

Dentro de las implementaciones para el EAC se encuentran la de Singapur (EAC-Singapur) y la de la Unión Europea (EAC-UE). El EAC-UE basa su implementación en dos mecanismos: la autenticación del *chip* y la autenticación del terminal de lectura. El primero de estos mecanismos demuestra la veracidad del *chip* basado en una fuerte encriptación mientras que el segundo le permite comprobar al *chip* que el sistema de inspección está autorizado a leer la información sensible que contiene (Valdes & Portilla, 2013). En la ejecución del EAC-UE versión 2, para la autenticación del *chip* es necesaria la generación de llaves de sesión (MAC y ENC) a partir de un secreto compartido, dentro de la tarjeta inteligente. Para la generación de estas llaves se suelen usar algoritmos de cifrado asimétricos como RSA, DH[48], ECDH[49].

---

[47] Organización de la Aviación Civil Internacional (traducido de las siglas en inglés ICAO: *International Civil Aviation Organization*).

[48] *Diffie – Hellman.*

[49] *Elliptic Curve Diffie – Hellman.*

El ECDH está basado en la utilización de Curvas Elípticas (Federal Office for Information Security, 2012) de conjunto con las operaciones de *Diffie- Hellman*, posibilitando generar claves más cortas dado el aumento significativo de la complejidad de las operaciones matemáticas que se realizan. Debido a que las tarjetas *GemXpresso Pro R3.2 E32 PK* no soportan este algoritmo de cifrado, no es posible generar las llaves de este tipo dentro de la tarjeta como se requiere. A través de la implementación de una operación de extensión sobre la plataforma, se logró la generación de llaves de sesión para la tarjeta, usando el algoritmo ECDH de 128,192 y 256 bits. De esta manera se demuestra que es posible realizar un Control de Acceso Extendido versión 2 europea, usando tarjetas con bajas prestaciones que no soportan Curvas Elípticas.

Como otra de las implementaciones que se realizaron se encuentra la generación de llaves usando el algoritmo RSA, debido a que es un algoritmo ampliamente usado tanto en conexiones de Internet, como en protocolos seguros y cifrados de datos. Aunque las tarjetas soportan el algoritmo RSA, es solamente para llaves con tamaños hasta 2048 bits, no pudiéndose generar llaves de mayor tamaño dentro de la tarjeta inteligente, las cuales son significativamente más seguras. Se implementó una operación de extensión que permitió generar un par de llaves RSA de 3072 y 4096 bits en el servidor web, usando la plataforma de extensión. De esta manera se solventa la limitación de la generación de llaves de mayor tamaño en este modelo de tarjetas.

En la Tabla 13 se resumen las operaciones implementadas y los resultados en cuanto a los tiempos de respuesta de la ejecución de estas operaciones en el servidor.

Tabla 13. Resultados obtenidos en la generación de llaves ECDH y RSA

Operación	**Tamaño**	**Tiempo de operación en el servidor**
ECDH *Session Keys*	128 bits	987 ms
ECDH *Session Keys*	192 bits	994 ms
ECDH *Session Keys*	256 bits	1 s
RSA *keys*	3072 bits	8 s 299 ms
RSA *keys*	4096 bits	12 s 850 ms

3.7.3. Escenario 3. Almacenamiento y consulta de información asociada a una aplicación de una tarjeta

En cuanto a la extensión de la capacidad de almacenamiento, la memoria EEPROM disponible en la tarjetas GemXpresso Pro R3.2 E32 PK es de tan solo 29.7 kB, luego de ser personalizada por el emisor. Este tamaño limita el número de aplicaciones que pueden ser instaladas en la tarjeta, así como las variables e información que estas pueden manejar. La plataforma de extensión por su parte, provee un mecanismo para el almacenamiento en una base de datos relacional, donde el tamaño está en el orden de los terabytes y está asociado a los recursos del servidor. Mediante el almacenamiento y la recuperación de datos, se evidencia que también es posible extender la capacidad de memoria de las tarjetas inteligentes *Java Card*. Esta posibilidad ofrece ventajas en diversos escenarios, como por ejemplo donde se necesite que los datos de las aplicaciones estén disponibles para realizar un *backup* en caso de pérdida de la tarjeta inteligente.

En la Tabla 14 se resumen las características seleccionadas en los escenarios descritos con anterioridad, que evidencian la extensión de las capacidades de procesamiento y memoria de las tarjetas inteligentes *Gemplus GemXpresso Pro R3.2 E32 PK.*

Tabla 14. Extensión de las capacidades de procesamiento y memoria de las tarjetas Gemplus GemXpresso Pro R3.2 E32 PK

Características	**Modelo tradicional de desarrollo**	**Implementación de operaciones de extensión sobre la plataforma**
Tipos de datos soportados	*boolean, byte, short*	*boolean, byte, short,* ***char, double, float, int, long, BigNumbers***
Algoritmos criptográficos soportados	DES, 3DES, SHA, MD5, RSA 1024, RSA 2048, DAP	DES, 3DES, SHA, MD5, RSA 1024, RSA 2048, **RSA 4096**, DAP, **ECDH** (posibilidad de otras implementaciones)
Capacidad de almacenamiento	Hasta 29.7 kB disponible en memoria EEPROM	**En el orden de los terabytes** (según los recursos del servidor)

Conclusiones del capítulo

La plataforma para la extensión de las capacidades de procesamiento y memoria de aplicaciones *Java Card* en tarjetas inteligentes fue desarrollada como aplicación práctica del modelo de extensión presentado como resultado de esta investigación, permitiendo la implementación de operaciones para el almacenamiento de datos o la ejecución de algoritmos costosos fuera de la tarjeta inteligente. La plataforma brinda tres

componentes fundamentales para la implementación y ejecución de estas operaciones, los cuales fueron concebidos con responsabilidades bien definidas y con el objetivo de minimizar el desarrollo que implica la ejecución de nuevas operaciones de extensión fuera de la tarjeta inteligente. El componente de la tarjeta (JCAEC) permite abstraer al desarrollador de la codificación de las operaciones necesarias para la solicitud de una operación de extensión y su posterior procesamiento en la tarjeta. El componente en el terminal (JCAEP) sirve de intermediario de la comunicación entre los componentes de la plataforma, abstrayendo al *middleware* en el terminal de la ejecución de alguna operación fuera de la tarjeta. Por su parte, el *Extension Service* es un servicio web genérico que permite la ejecución de las operaciones de extensión previamente implementadas y publicadas.

Las tecnologías y herramientas seleccionadas para el desarrollo de la plataforma brindan la posibilidad de ser utilizada en los sistemas operativos Linux y Windows. La independencia del servidor web facilita una posible migración.

La primera versión de la plataforma fue sometida a pruebas de software que permitieron la corrección de algunas fallas y demostrar su capacidad de procesar los comandos recibidos, así como ejecutar operaciones de extensión para los comandos que así lo requirieran de forma satisfactoria. La implementación de operaciones de extensión sobre la plataforma permitió evidenciar la extensión de las capacidades de procesamiento y memoria de las tarjetas inteligentes *Java Card* Gemplus GemXpresso Pro R3.2 E32 PK en diferentes escenarios, sirviendo de guía además para el desarrollo de otras aplicaciones de este tipo con el uso de la plataforma.

Conclusiones

En esta investigación se dio respuesta al problema planteado referente a las limitaciones de los recursos de *hardware* de las tarjetas inteligentes *Java Card*, que inciden en el desarrollo de aplicaciones para estos dispositivos; y se arribaron a las siguientes conclusiones:

- Del estudio realizado sobre las soluciones de software existentes para combinar el uso de los recursos de *hardware* de un ordenador de conjunto con una tarjeta inteligente, resultó que son insuficientes las soluciones de este tipo que explotan este enfoque para contrarrestar las limitaciones del *hardware* de estas tarjetas. El análisis de las soluciones encontradas a nivel mundial permitió identificar algunos requisitos deseados que se tuvieron en cuenta para el diseño del modelo que se propone como parte de esta investigación.

- El modelo de extensión obtenido cumple con los principales estándares para tarjetas inteligentes y permite el uso de un ordenador de conjunto con una tarjeta *Java Card*, para la ejecución de operaciones de almacenamiento de datos y/o ejecución de algoritmos de alto coste computacional, en tarjetas con bajas prestaciones.

- La plataforma desarrollada permite la implementación y ejecución, de manera simple y con poco esfuerzo, de operaciones de extensión que requieren recursos que la tarjeta no es capaz de proveer. Esta plataforma cumple con los estándares y principales componentes propuestos en el modelo de extensión.

- La implementación de operaciones de extensión para diferentes escenarios, haciendo uso de la plataforma de desarrollo, permitió demostrar la factibilidad del modelo de extensión propuesto.

Recomendaciones

Se recomienda para futuras versiones de esta investigación:

- Implementar una aplicación que facilite la administración de las operaciones de extensión en el servidor web.
- Integrar un Sistema de Administración de llaves para el manejo seguro de las llaves en el servidor y el cifrado de la información en la base de datos.
- Utilizar la plataforma implementada en futuros desarrollos del Centro de Identificación y Seguridad Digital que utilicen tarjetas inteligentes de bajas prestaciones.

.

Referencias Bibliográficas

Anciaux, N., Bobineau, C., Bouganim, L., Pucheral, P., & Valduriez, P. (2001). PicoDBMS: Validation and Experience. In ACM (Ed.), *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)* (pp. 709–710). Roma, Italy: Morgan Kaufmann Publishers Inc.

Apache Software Foundation. (2012). Apache Axis2. Retrieved December 12, 2013, from http://axis.apache.org/axis2/java/core/docs/

Auerbach, N., & Maibaum, N. (2002). FASME-A Step Towards European E-Government Solutions. *Proceedings of the10th European Conference on Information Systems. Information Systems and the Future of the Digital Economy (ECIS)* (pp. 1197–1205). Gdansk, Poland: O'Reilly.

Beck, K. (2003). *Test-Driven Development By Example*. Addison-Wesley Professional.

Bergman, C. (2008). Match-on-card for secure and scalable biometric authentication. *Advances in Biometrics* (pp. 407–421). Springer London.

Blaze, M. (1996). High-Bandwidth Encryption with Low-Bandwidth. *Proceedings of the Fast Software Encryption Workshop* (pp. 33–40). Springer Berlin Heidelberg.

Blaze, M., Feigenbaum, J., & Naor, M. (1999). A formal treatment of remotely keyed encryption. *Proceedings of the 10th annual ACM-SIAM symposium on Discrete algorithms* (pp. 868–869). Society for Industrial and Applied Mathematics.

Bobineau, C., Bouganim, L., Pucheral, P., & Valduriez, P. (2001). PicoDBMS : Scaling down Database Techniques for the Smartcard. *The VLDB Journal*, *10*(2-3), 120–132.

Bouncy Castle Inc. (2013). Bouncy Castle Crypto API. Retrieved February 20, 2014, from http://bouncycastle.org/

Bourlai, T., Kittler, J., & Messer, K. (2010). On design and optimization of face verification systems that are smart-card based. *Machine Vision and Applications*, *21*(5), 695–711.

Bourlai, T., Messer, K., & Kittler, J. (2005). Scenario based performance optimisation in face verification using smart cards. *Audio-and Video-Based Biometric Person Authentication* (pp. 289–300). Springer Berlin Heidelberg.

Bringer, J., Chabanne, H., Kevenaar, T. A. M., & Kindarji, B. (2009). Extending match-on-card to local biometric identification. *Biometric ID Management and Multimodal Communication* (pp. 178–186). Springer.

Cap, C. H., Maibaum, N., & Heyden, L. (2001). Extending the data storage capabilities of a Java-based smartcard. *Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on* (pp. 680–685). IEEE.

CardLogix Corporation. (2009). Smart Card & Security Basics. Retrieved March 7, 2014, from http://www.smartcardbasics.com/pdf/7100030_BKL_Smart-Card-&-Security-Basics.pdf

Castel, B. du. (2013). Personal History of the Java Card. Retrieved February 15, 2014, from https://sites.google.com/site/personalhistoryofthejavacard/

Chen, Z. (2000). *Java card technology for smart cards: architecture and programmer's guide*. Addison-Wesley Professional.

Federal Office for Information Security. (2007). Advanced Security Mechanisms for Machine Readable Travel Documents - Extended Access Control (EAC). Bonn, Germany.

Federal Office for Information Security. (2012). *Elliptic Curve Cryptography versión 2.0* (p. 23). Bonn, Germany.

Friedrich, E., & Seidel, U. (2006). The introduction of the German e-passport. Biometric passport offers first-class balance between security and privacy. *Keesing Journal of Documents and Identity*, *16*, 2006.

Gemalto. (2004). *GemXpresso Pro R3.x. Reference Manual* (p. 146). France.

Gemalto. (2013). Gemalto Developer Suite. Retrieved February 8, 2014, from http://www.gemalto.com/

Global Platform. (2003). Global Platform Card Specification Version 2.1.1. GlobalPlatform.

Global Platform. (2011). Global Platform Card Specification Version 2.2.1.

GlobalPlatform. (2014). Global Platform Our Mission. Retrieved March 12, 2014, from http://www.globalplatform.org/aboutus.asp

Grother, P., Salamon, W., Watson, C., Indovina, M., & Flanagan, P. (2009). *Performance of Fingerprint Match-on-Card Algorithms* (Vol. 7477, p. 33). National Institute of Standards and Technology.

Hunt, J., & Holcombe, B. (2004). *Government smart card handbook*. U.S General Services Administration Office of Governmentwide Policy and the Smart Card Interoperability Advisory Board.

ICAO. (2006a). Documento 9303. Parte 1 Pasaportes de lectura mecánica. Volumen 1 Pasaportes con datos de lectura mecánica almacenados en formato óptico de reconocimiento de caracteres.

ICAO. (2006b). Documento 9303. Parte 1 Pasaportes de lectura mecánica. Volumen 2 Especificaciones para pasaportes electrónicos con capacidad de identificación biométrica.

ISO/IEC. (1999). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 7: Interindustry commands for Structured Card Query Language (SCQL).

ISO/IEC. (2004a). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 5: Registration of application providers.

ISO/IEC. (2004b). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 6: Interindustry data elements for interchange.

ISO/IEC. (2004c). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 8: Commands for security operations.

ISO/IEC. (2004d). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 9: Commands for card management.

ISO/IEC. (2004e). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 11: Personal verification through biometric methods.

ISO/IEC. (2005a). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 4: Organization, security and commands for interchange.

ISO/IEC. (2005b). 19794-2 Information technology — Biometric data interchange formats. Finger minutiae data.

ISO/IEC. (2007). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 2: Cards with contacts — Dimensions and location of the contacts.

ISO/IEC. (2008a). ISO/IEC 14443. Identification cards-- Contactless integrated circuit cards - Proximity cards - Part 4: Transmission protocol.

ISO/IEC. (2008b). ISO/IEC 14443.Identification cards- Contactless integrated circuit cards- Proximity cards - Part 1: Physical characteristics.

ISO/IEC. (2008c). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 15: Cryptographic information application.

ISO/IEC. (2010). ISO/IEC 14443. Identification cards - Contactless integrated circuit cards - Proximity cards - Part 2: Radio frequency power and signal interface.

ISO/IEC. (2011a). ISO/IEC 7816. Identification cards — Integrated circuit cards-Part 1: Cards with contacts- Physical characteristics.

ISO/IEC. (2011b). ISO/IEC 14443.Identification cards - Contactless integrated circuit cards - Proximity cards - Part 3: Initialization and anticollision.

Jardines, A. (2013). Guía para establecer una autenticación mutua y abrir un canal seguro GlobalPlatform 2.1.1 en Tarjetas Inteligentes. *XI Seminario Iberoamericano de Seguridad en las Tecnologías de la Información. XV Convención de Informática*. Havana, Cuba.

Kleef, F. va. (2006). Biometrics in the Dutch passport. *Journal of Documents & Identity*, (16).

Legón, A. C., Ramírez, S. M., Landrian, J., Almeida, D., Surós, A., & Carratala, F. A. (2013). Sistema de gestión de servicios para documentos de identificación nacional electrónicos. *XI Seminario Iberoamericano de Seguridad en las Tecnologías de la Información. XV Convención Informática*. Havana, Cuba.

Lewis, W. E. (2004). *Software Testing and Continuous Quality Improvement*. CRC Press.

Lucks, S. (1997). On the security of remotely keyed encryption. *Proceedings of the Fast Software Encryption Workshop* (pp. 219–229). Springer Berlin Heidelberg.

Lucks, S. (1999). Accelerated remotely keyed encryption. *Proceedings of the Fast Software Encryption Workshop* (pp. 112–123). Springer Berlin Heidelberg.

Márquez, J. T. (2006). *Nuevo marco de autenticación para tarjetas inteligentes en red. Aplicación al pago electrónico en entornos inalámbricos*. Universidad Carlos III de Madrid.

Mayes, K. E., & Markantonakis, K. (2008). *Smart cards, Tokens, Security and Applications*. London, UK: Springer.

Mohammed, L., Rahman, A., Prakash, V., & Daud, M. (2004). Smart Card Technology: Past, Present, and Future. *International Journal of The Computer, the Internet and Management*, *12*(1), 12–22.

Noore, A. (2000). Highly robust biometric smart card design. *IEEE Transactions on Consumer Electronics*, *46*(4), 1059–1063.

Oracle, & Java Card Forum. (2012). Java Card Forum. Retrieved March 25, 2014, from http://www.javacardforum.org

Ortiz, C. (2003). An Introduction to Java Card Technology - Part 3. The Smart Card Host Application. Retrieved February 20, 2014, from http://www.oracle.com/technetwork/java/javacard/javacard3-138837.html

PC/SC Workgroup. (2013). PC/SC Specification Version 2.01.11 Release. (En linea). Retrieved February 12, 2014, from http://www.pcscworkgroup.com/

PostgreSQL Development Group. (2014). PostgreSQL. Retrieved March 12, 2014, from http://www.postgresql.org/

Ramírez, S. M., & Legón, A. C. (2011). Documento de arquitectura de software. Proyecto Secure Data Manager. Universidad de las Ciencias Informáticas.

Ramírez, S. M., Legón, A. C., & Valle, Y. (2014). Modelo para la extensión de las capacidades de procesamiento y memoria de tarjetas inteligentes Java Card. *Revista Cubana de Ciencias Informáticas*, *8*(1), 112–126. Retrieved from http://rcci.uci.cu

Rankl, W., Effing, W., & Cox, K. (2010). *Smart card handbook* (Fourth Edi.). John Wiley & Sons.

Rouine, M., & Murphy, C. (2005). The New Irish Passport - New passport, new system, new processes. *Journal of Documents & Identity*, (14).

RSA Laboratories. (2002). PKCS #1 v2.1: RSA Cryptography Standard.

Sanchez, R., Mengibar, L., & Sanchez, C. (2003). Microprocessor smart cards with fingerprint user authentication. *Aerospace and Electronic Systems Magazine*, *18*(3), 22– 24.

Smart Card Alliance. (2012). Smart Card Technology in US Healthcare: Frequently Asked Questions.

Smart Card Alliance. (2013a). About Smart Cards: Introduction: Primer – Smart Card Alliance. Retrieved December 6, 2013, from http://www.smartcardalliance.org/pages/smart-cards-intro-primer

Smart Card Alliance. (2013b). Smart Cards Applications. Retrieved January 15, 2013, from http://www.smartcardalliance.org/pages/smart-cards-applications

Sun Microsystems. (2008). *THE JAVA CARD ™ 3 PLATFORM* (p. 30). Retrieved from http://www.oracle.com/technetwork/articles/javase/javacard3-whitepaper-149761.pdf

SUN Microsystems. (2009). Java Card Connected Platform Specification 3.0.1. Oracle. Retrieved from http://www.oracle.com/technetwork/java/javame/javacard/download/default-1492179.html

SUN Microsystems. (2011). Java Card Classic Platform Specification 3.0.4. Oracle. Retrieved from http://www.oracle.com/technetwork/java/javacard/specs-jsp-136430.html

Valdes, D., & Portilla, R. D. (2013). *Módulo de Control de Acceso Extendido para el sistema de verificación de seguridad de documentos de viajes de lectura mecánica (DocSec)*. Universidad de las Ciencias Informáticas.

Well, D. (2013). Extreme Programming: A gentle introduction. Retrieved December 12, 2013, from http://www.extremeprogramming.org/