ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

Diseño e implementación de un lector PC/SC inalámbrico para tarjeta inteligente basado en plataformas móviles NFC

(Design and development of a PC/SC wireless smart card reader using NFC enabled mobile devices)

Para acceder al Titulo de

INGENIERO DE TELECOMUNICACIÓN

Autor: José María Alonso Rodríguez

Marzo - 2013

Resumen

En los últimos años las tarjetas inteligentes han experimentado un gran crecimiento, siendo habitual su presencia en las carteras de muchos usuarios para tareas como pagos e identificación. Sin embargo, en el ámbito doméstico aún no son utilizadas por la mayor parte de los usuarios debido al coste de los lectores y su escasa disponibilidad. Este trabajo busca posibilitar el uso de las tarjetas inteligentes desde los ordenadores del hogar, empleando como lector algo tan extendido como los teléfonos móviles, aprovechando las amplias opciones de conectividad de los modelos actuales. De esta forma se abre la posibilidad de realizar tareas tales como por ejemplo la recarga de una tarjeta de transporte o la identificación desde el hogar con la comodidad que supone no tener que desplazarse, y sin tener que hacer un desembolso adicional. Como consecuencia se prevé incentivar la creación de aplicaciones que hagan uso de las tarjetas inteligentes.

Palabras clave: NFC, Tarjeta Inteligente, Android, PC/SC, Lector Inalámbrico, Bluetooth

Abstract

During the last few years smart cards have become widely spread, being easily found among the wallets of many users, for tasks such as payments or identification. However, due to the cost of readers and the scarce amount of choices, smart cards are not yet being used at home. The present work intends to facilitate the use of smart cards from the users' computers, using a cell phone as a reader, taking advance of the connectivity options of the newest models. This will offer the possibility to do things such as, for instance, recharging a transport card or identifying comfortably without having to go to a specific place, and without an additional cost. As a consequence, a rise in the development of applications involving smart cards is foreseen.

Key Words: NFC, Smart Card, Android, PC/SC, Wireless Reader, Bluetooth

Agradecimientos

A Sara, que siempre me apoya (y me soporta).

A mis padres, que saben que si a veces no me empujan un poco no avanzo.

A Manuel Menchaca, con quien me es imposible empezar más proyectos inacabados, aunque espero seguir haciéndolo.

A Cristino Salcines, que se apareció en mis exámenes para gritar: *¡piensa!*

A toda la gente del laboratorio de Telemática, que nunca dudan en echarte una mano.

A Jorge Lanza, por posibilitar este trabajo.

Al maestro Yoda por la fuerza.

Gracias.

Índice

Índice de figuras

Índice de tablas

1.Introducción

Hoy en día no se entiende una sociedad sin telecomunicaciones: telefonía, informática, radio, televisión y otras muchas tecnologías forman parte de la vida cotidiana de la práctica totalidad de las personas. En España, con 47 millones de habitantes [1], el número de líneas de telefonía móvil activas en 2011 era ya de 56 millones. Asimismo, los ordenadores han alcanzado el 71, 5% de los hogares [2].

Las tecnologías, además de expandirse en alcance, lo hacen en funcionalidades. Ya no existe una división clara de las funciones que realiza cada tipo de dispositivo. En menos de 20 años, los teléfonos móviles han pasado de objeto de lujo a ser simultáneamente una herramienta de comunicación, de trabajo, una consola de videojuegos y un centro multimedia, entre otros muchos usos. Del mismo modo, los ordenadores, que en un principio fueron herramientas difíciles de manejar y prácticamente imposibles de obtener, son hoy una necesidad para cualquier persona que desee trabajar, estudiar, o sencillamente realizar una gestión, desde papeleo oficial a la compra de un billete de avión.

La progresiva miniaturización y abaratamiento de los componentes electrónicos ha propiciado que los ordenadores lleguen, casi de manera inadvertida, hasta nuestros bolsillos. Dispositivos con una potencia superior a los sistemas que se implantaban en la década de los 80, son hoy del tamaño de una uña, ofrecen un mayor rendimiento y están totalmente inmersos en nuestra vida cotidiana. Un ejemplo son las conocidas tarjetas inteligentes, empleadas en entornos financieros, transportes, etc. Si bien el chip que emplean ocupa un pequeño tamaño, por comodidad y familiaridad se suelen distribuir con el tamaño genérico de una tarjeta de crédito. Otro lugar en que se encuentran estas tarjetas es la tarjeta SIM de los teléfonos móviles. Así, muchos usuarios llevan varios ordenadores en miniatura en sus bolsillos sin ser siquiera conscientes de ello.

Muchas empresas distribuyen este tipo de tarjetas para facilitar la identificación de sus empleados en sus instalaciones. Las entidades bancarias han ido sustituyendo paulatinamente las tarjetas de banda magnética por otras que incorporan chips. Existen distintos tipos de tarjetas inteligentes dependiendo del interfaz de comunicación que apliquen, con contactos y sin contactos. Aunque las primeras son las que tienen un mayor volumen, actualmente las tarjetas sin contacto están tomando la alternativa dado que manteniendo las funcionalidades incorporan mejoras que facilitan su uso. Estas tarjetas funcionan simplemente con aproximarlas a un lector. En ocasiones se emplean tarjetas duales, que mantienen el chip visible para realizar operaciones que requieran interfaz con contactos y además tienen capacidad inalámbrica. Uno de los entornos de uso más conocidos es el transporte público, donde los ayuntamientos progresivamente han sustituido los clásicos “bono de transporte” de cartón por este tipo de tarjeta sin contactos (figura 1.1). De esta forma las operaciones resultan más cómodas, sencillas y rápidas.

La tecnología empleada en las tarjetas que incorporan comunicaciones sin contactos se basa en la identificación por Radio Frecuencia (RFID), donde la inducción de campos magnéticos hace posible la comunicación entre dispositivos a corta distancia. Del masivo uso de estas tecnologías en la actualidad y del gran abanico de funcionalidades que proporcionan ha surgido la necesidad de incorporarlas en otros dispositivos además de las tarjetas inteligentes.

Figura 1.1: Sistema de pago sin contactos en el autobús

Los teléfonos móviles se erigen como punta de lanza por ser el dispositivo actualmente más vinculado a los usuarios. Además se trata de herramientas que disponen de todo lo necesario para realizar múltiples tareas: teclado, pantalla, conexión a Internet... Esto propicia que se puedan realizar diversas operaciones sin un interfaz externo, a diferencia de las tarjetas, que requieren de un dispositivo lector. No ajenos a la creciente importancia de las tecnologías de campo cercano (NFC), cada vez más fabricantes lo incluyen en sus terminales: Blackberry, LG, Nokia, Motorola o Samsung, entre otros muchos, ya tienen en el mercado dispositivos capaces de leer tarjetas sin contactos.

Pese a la creciente tendencia en el uso de tarjetas inteligentes a medida que más empresas se pasan a esta tecnología, el uso de las mismas está restringido a los lugares en que se dispone de un lector. A pesar del extensivo uso que se hace de las tarjetas, la mayoría de los dispositivos de informática de usuario no tienen funcionalidad lectora, lo que frena la expansión del uso de las tarjetas en los hogares y limita la explotación de las capacidades de esta tecnología. La mayoría de usuarios no está dispuesta a asumir el gasto adicional que supone la adquisición de un lector dada la escasez de aplicaciones disponibles, limitadas prácticamente al uso del Documento Nacional de Identidad electrónico (DNIe) o la identificación en redes privadas virtuales (VPN) de algunas empresas. La aparición del DNIe en el año 2006 (figura 1.2), podría haber impulsado el uso y conocimiento de las funcionalidades de las tarjetas inteligentes, puesto que gracias a él los usuarios podrían realizar múltiples gestiones de manera segura y sencilla desde cualquier localización [3]:

- Realizar compras seguras a través de Internet
- Hacer trámites completos con las Administraciones Públicas a cualquier hora y sin tener que desplazarse ni hacer colas.
- Realizar transacciones seguras con entidades bancarias.
- Utilizar de forma segura nuestro ordenador personal.
- Participar en una conversación por Internet con la certeza de que nuestro interlocutor es quien dice ser.

Sin embargo, el no disponer de lectores hace que el progreso de este tipo de soluciones se ralentice. Por eso parece necesario buscar diferentes metodologías de usar la tarjeta inteligente, es decir, analizar tecnologías para implementar un lector.

Figura 1.2: DNI electrónico

Es en este contexto en el que surge la idea para este trabajo, que combina las tecnologías antes citadas con el objetivo de usar un dispositivo móvil, como dispositivo que hoy en día tienen prácticamente todos los ciudadanos, como lector de tarjetas para un PC.

1.1. Objetivos

El propósito de este trabajo es el diseño y desarrollo de un sistema que permita hacer de un teléfono móvil o *smartphone* con capacidades NFC un lector de tarjeta inteligente de un ordenador personal. Así los ordenadores que no disponen de lectores de tarjeta inteligente empelarán el terminal móvil como pasarela de comunicaciones. Buscando facilitar la usabilidad y aprovechando las capacidades de los terminales móviles actuales, para la conexión del lector al PC se empleará un protocolo inalámbrico para la comunicación, en lugar de la tradicional conexión por USB o puerto serie.

Figura 1.3: Sistema de comunicación propuesto

Se pretende implementar una solución en la que tanto la instalación del sistema como su utilización resulten sencillas para el usuario. En ese sentido, se procurará que las modificaciones sobre los dispositivos del usuario sean mínimas y con procedimientos familiares, tratando de evitar la instalación de imágenes específicas del sistema operativo (conocidas como ROMs) o la realización de operaciones en modo superusuario que impliquen la invalidación de la garantía del dispositivo o su incorrecto funcionamiento.

Para materializar un sistema completamente funcional que permita la utilización de un teléfono o tablet como lector inalámbrico en un PC, las etapas que se plantea seguir son las siguientes:

- Comprensión y desarrollo de controladores de lectores de tarjeta inteligente que permitan considerar al teléfono móvil como lector.
- Diseño de un protocolo de intercambio que permita comunicar el PC con el móvil para que opere como lector de tarjeta inteligente.
- Desarrollo de un aplicativo en teléfono móvil que permita la gestión de la comunicación entre el PC y la tarjeta inteligente. Hacerlo de manera que sea lo más genérico posible y que no dependa del dispositivo.

1.2. Estructura de la memoria

Esta memoria se ha estructurado en 8 capítulos como se indica a continuación:

- En el **capítulo 2** se introducirá la teoría relativa a las tarjetas inteligentes, acerca de su historia, tipos, especificaciones, y usos.
- El **capítulo 3** tratará brevemente la tecnología NFC, sus usos y principales características.
- El **capítulo 4** explicará todo lo relativo al entorno del PC, tanto la aplicación creada para operar con tarjetas inteligentes como el diseño e implementación del protocolo de comunicaciones y el controlador para el lector inalámbrico.
- El **capítulo 5** describe el aplicativo móvil que permite la operativa de éste como lector de tarjeta inteligente. Se detallaran las fases de su desarrollo y la operativa implementada, así como su integración con el controlador desarrollado en el el equipo PC.
- Las conclusiones del presente trabajo serán expuestas en el **capítulo 6**.

Adicionalmente se incluye un **anexo** en el que se incluye una breve descripción del procedimiento de instalación y uso del sistema desarrollado.

2.La tarjeta inteligente

Desde que a mediados del siglo XX surgieran las primeras tarjetas plásticas hasta nuestros días, muchas y muy variadas han sido las mejoras introducidas en el sector. Aunque de apariencia muy similar a una tarjeta de crédito, la tarjeta inteligente se presenta como un dispositivo muy diferente. Dotadas de un circuito integrado en su cuerpo de plástico, las tarjetas inteligentes disponen de la capacidad para proteger y procesar datos almacenados en ellas. Esta inteligencia que les proporciona el chip, las convierte en un ordenador portable y las diferencia de sus predecesoras, las tarjetas de banda magnética.

El nacimiento de la tarjeta con chip integrado, *Chip Card*, tiene lugar a principios de los 70 cuando un ingeniero japonés presenta la primera tarjeta de plástico que incorpora, embebido en ella, un circuito integrado. Se trataba en realidad no de una tarjeta inteligente sino de un dispositivo de almacenamiento en memoria. Cuatro años más tarde, el francés Ronald Moreno ideó lo que podemos considerar como la primera tarjeta inteligente. Dotada de ciertos mecanismos de seguridad, entre los que cabe destacar el empleo de un número de identificación personal o PIN (*Personal Identification Number*), aún usado en la actualidad, cubría las necesidades demandadas en la época en cuanto a capacidad de almacenamiento y seguridad. Sin embargo, no fue hasta la década de los 80 cuando se inició la explotación comercial de las tarjetas inteligentes, también denominadas *smart cards*. Empleadas con éxito como tarjetas bancarias o de prepago en telefonía, solventaron los problemas de fraude bancario con tarjeta de crédito y las falsificaciones. Los avances tecnológicos en circuitos integrados y criptografía no han hecho sino permitir la evolución de las tarjetas inteligentes hacia unos dispositivos más potentes, seguros y baratos.

Por su seguridad y portabilidad son el elemento idóneo para el almacenamiento y procesado de información confidencial, haciéndolas especialmente atractivas en la generación y custodia de claves secretas y la ejecución de los algoritmos criptográficos involucrados. Además de en las aplicaciones bancarias y telefonía anteriormente mencionadas, se puede utilizar una tarjeta inteligente como llave de acceso, como identificador en telefonía móvil o para otros múltiples servicios.

2.1. Tipos de tarjetas

Las tarjetas chip presentan numerosas ventajas en comparación con las tarjetas de banda magnética, superando las debilidades de éstas:

- Mayor capacidad de almacenamiento de información.
- Protección de accesos no autorizados de la información almacenada en su memoria.
- Memoria resistente al deterioro de la información almacenada.

Es bastante frecuente denominar tarjetas inteligentes a todas las tarjetas que poseen contactos dorados o plateados sobre su superficie. Sin embargo este término es bastante ambiguo y conviene hacer una clasificación rigurosa. Agrupándolas según las características del chip, su componente más importante, éstas están clasificadas según el

tipo de circuito en tarjetas de memoria y tarjetas inteligentes (figura 2.1). Existen otras posibles clasificaciones según el método de acceso, su forma, . . .

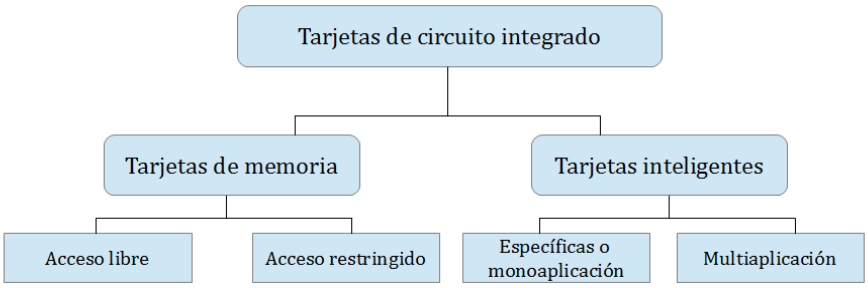

Figura 2.1: Clasificación de tarjetas chip

Atendiendo a distintos criterios se pueden hacer varias clasificaciones generales de las tarjetas inteligentes, aquí se detallarán dos de ellas: según las características del chip, y según el método de acceso.

2.1.1. Según las características del chip

En las **tarjetas de memoria** el chip únicamente dispone de capacidades como almacén de datos. En función de la seguridad que requiere el contenido que guardan, el acceso a su memoria puede estar protegido con circuitos de seguridad habilitados a partir del empleo de una clave (figura 2.2). Ejemplos de este tipo son las tarjetas telefónicas, que únicamente almacenan el saldo restante dejando el control sobre los datos a una aplicación exterior.

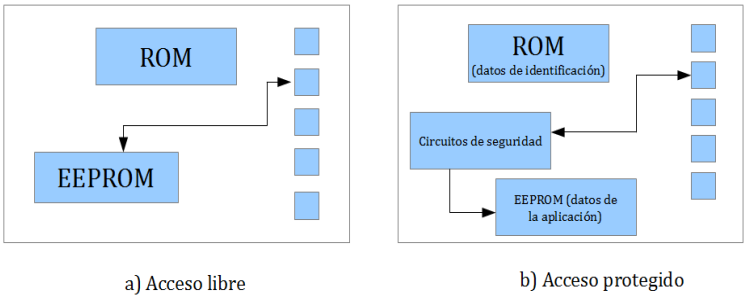

Figura 2.2: Pantallas de inicio del proceso de firma

La introducción de un microprocesador en su circuito integrado es lo que diferencia a las **tarjetas inteligentes** de las anteriores. Este elemento es el que las hace inteligentes, permitiéndoles ejecutar comandos y trabajar con los datos que contiene y otros que pueda suministrar algún dispositivo externo. Las tarjetas con microprocesador son bastante flexibles puesto que pueden realizar muy diversas funciones. En el caso más simple sólo contienen datos referentes a una aplicación específica. No obstante, los sistemas operativos de las tarjetas más modernas hacen posible que puedan integrar varios programas sobre un único soporte.

2.1.2. Según el método de acceso

Las **tarjetas con contactos** transfieren los datos a través de los contactos de su superficie. El deterioro de estos contactos es, en base a la experiencia, el mayor causante de fallo, limitación superada por las tarjetas sin contactos, que se verán a continuación.

Las **tarjetas sin contactos** realizan las transferencias de datos mediante campos magnéticos evitando así el deterioro de los contactos que lleva a la inutilización de otros tipos de tarjeta. Sin embargo, esta ventaja conlleva inconvenientes como la potencia necesaria para su activación. Además debido a que las tarjetas sin contactos combinan tecnología analógica y digital, se hace más difícil su integración, elevando su coste muy por encima de las con contactos.

Con la intención de combinar las características de los dos tipos de tarjetas anteriores han surgido las **tarjetas combi**. Algunos diseños presentan dos circuitos independientes según la forma de acceso, mientras que otros comparten la información para ambos métodos de acceso. El coste de fabricación de este modelo de tarjeta es muy alto, lo que unido a la baja demanda, hace que los precios de venta sean elevados y, por tanto, su penetración en el mercado menor que en los otros modelos.

2.2. Arquitectura

La mayoría de las tarjetas inteligentes adoptan una arquitectura de máquina de Von Neumann, esto es, están constituidas por una unidad de proceso o CPU, encargada de ejecutar en modo secuencial una serie de instrucciones almacenadas, junto a los datos requeridos, en una unidad de memoria y disponen de unos dispositivos de entrada y salida para comunicarse con el exterior. El microcontrolador de una tarjeta inteligente está compuesto por un microprocesador y tres tipos de memoria: ROM (*Read Only Memory*), EEPROM (*Electrically Erasable Programmable Read Only Memory*) y RAM (*Random Access Memory*) (Figura 2.3). El microprocesador y la memoria están fabricados sobre el mismo chip, haciendo caro y difícil interceptar las señales que se intercambian entre ambos, proporcionando así una alta seguridad física de los datos almacenados en memoria.

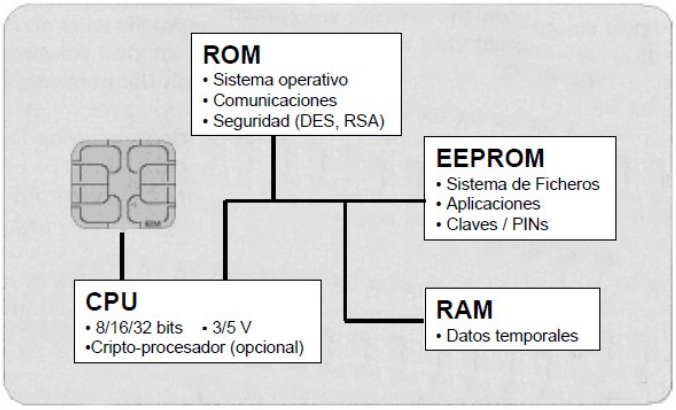

Figura 2.3: Arquitectura del circuito integrado de una tarjeta inteligente

- La **Unidad Central de Proceso** o **CPU** es la encargada de realizar los cálculos y procesar la información. Los procesadores más extendidos son los de 8 bits,

aunque también se han desarrollado de 16 y 32 bits. Algunas tarjetas incluyen un coprocesador criptográfico, preparado para realizar de forma eficiente operaciones aritméticas, que incrementa el rendimiento de las operaciones de seguridad y criptografía.

- La memoria **ROM** se utiliza para almacenar los datos permanentes de la tarjeta, que se escriben en la memoria durante la fase de fabricación de ésta: sistema operativo y aplicaciones y datos de usuario fijos. No se necesita alimentación para mantener este tipo de memoria, pero no se puede escribir en ella después de la fase de fabricación.
- La memoria **EEPROM** es una memoria no volátil en la que se encuentran datos de usuario y aplicación, así como información bajo el control del sistema operativo. La EEPROM es similar a la ROM, con la diferencia que se puede escribir en ella con algo de tiempo. Es una memoria lenta pero persistente. Funcionalmente equivale al disco duro de un ordenador, puesto que los datos permanecen en esta memoria y se pueden modificar cuando sea necesario. Los sistemas operativos de las tarjetas más modernas hacen posible integrar programas para distintas aplicaciones en una sola tarjeta. En este caso, la ROM contiene sólo el sistema operativo con las instrucciones básicas, mientras que el programa específico de cada aplicación se graba en EEPROM después de la fabricación.
- La memoria **RAM** es la memoria de trabajo del microprocesador. Al ser volátil, toda la información se pierde al desconectar la alimentación. Se usa para mantener los datos temporales durante una sesión. La lectura en la EEPROM es tan rápida como en la RAM, pero la escritura es mil veces más lenta.

De las tres memorias, la memoria ROM es la más barata, pues ocupa menos espacio por celda. Una celda de EEPROM ocupa hasta cuatro veces más que una de ROM, y una de RAM aproximadamente cuatro veces una de EEPROM. Ésta es la razón por la que las tarjetas contienen pequeñas cantidades de RAM. En el mercado encontramos chips que poseen desde 4K a 256K de ROM, de 1K a 256K de EEPROM y de 256 bits a 10K de RAM.

Como se ha explicado anteriormente, la capacidad de memoria de una tarjeta inteligente está claramente limitada, por lo que se usan algoritmos de compresión de datos para alojar la información en su interior.

2.3. Características físicas

El rasgo más distintivo de una tarjeta es, sin duda, su aspecto físico. Por razones de compatibilidad, la forma y el tamaño están estandarizados. Los formatos más utilizados son el ID-1 (SIM Standard) e ID-000 (Mini SIM) (figura 2.4 y tabla 1 [4], [5]).

- El formato **ID-1,** también conocido como **SIM standard** o **1FF** es el más usado por su semejanza con las tarjetas de banda magnética. Proporciona al usuario bastante comodidad en su manejo, ya que no es demasiado grande para llevarla en una cartera, ni demasiado pequeña para perderla con facilidad.
- El tipo **ID-000**, **Mini SIM** o **2FF** surge como solución a la necesidad de un formato más pequeño requerido en tecnologías en miniatura, como es la telefonía móvil. Actualmente sólo es empleado en teléfonos con tecnología GSM (*Global System for Mobile Communications*), que hacen uso una tarjeta inteligente denominada SIM (*Subscriber Identity Module*) o UICC (Universal Integrated Circuit Card).

- Los formatos **Micro SIM** (**3FF**) y **Nano SIM** (**4FF**) mantienen el tamaño de los conectores y el espesor, pero su anchura y altura se han reducido para adaptarlos a dispositivos más pequeños.

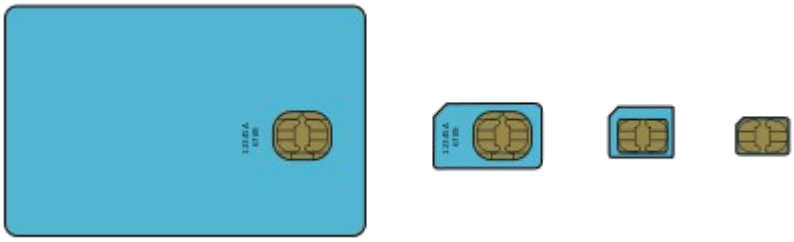

Figura 2.4: SIM Standard (1FF), Mini SIM (2FF), Micro SIM (3FF) y Nano SIM (4FF)

Tarjeta SIM	Estándar	Largo (mm)	Ancho (mm)	Espesor (mm)
Tamaño completo	ISO/IEC 7810:2003, ID-1	85,60	53,98	0,76
Mini SIM	ISO/IEC 7810:2003, ID-000	25	15	0,76
Micro SIM	ETSI TS 102 221 V9.0.0, Mini-UICC	15	12	0,76

Tabla 1: Estándares de tarjeta SIM

2.4. Protocolos de comunicación

Toda comunicación con una tarjeta inteligente siempre es iniciada por un dispositivo externo. Una tarjeta nunca transmite información sin que se haya producido antes una petición externa, lo que supone una relación maestro-esclavo, siendo el terminal el maestro y la tarjeta el esclavo. La transmisión de datos se realiza empleando un modelo serie asíncrono. Debido a la existencia de un único canal entre los dispositivos externos y la tarjeta, ambos deben turnarse para llevar a cabo la transmisión de datos, estableciéndose lo que se denomina canal *half-duplex* (figura 2.5). El procedimiento *full-duplex* aún no se encuentra disponible. Sin embargo, la mayoría de los microprocesadores incorporan un canal de entrada y otro de salida y, como dos de los ocho contactos disponibles en la tarjeta están sin uso, se podría añadir una segunda vía de comunicación.

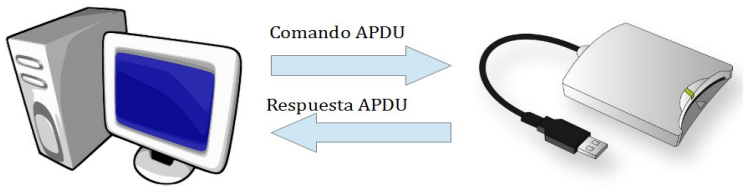

Figura 2.5: Modelo de comunicación

Cada vez que se inserta una tarjeta en el lector, sus contactos se conectan a los del terminal y se procede a activarlos. La tarjeta inicia entonces su encendido y envía una respuesta llamada ATR (*Answer To Reset*) hacia el terminal. El ATR contiene información referente a cómo ha de ser la comunicación tarjeta-lector, estructura de los datos intercambiados, protocolo de transmisión,... Una vez que el lector interpreta el ATR, procede a enviar la primera instrucción. La tarjeta procesa dicha instrucción y envía la respuesta asociada. Este intercambio de comandos y respuestas concluye una vez que la tarjeta es desactivada.

La información de niveles superiores se intercambia mediante unidades de protocolo APDU (*Application Protocol Data Unit*). Su formato está especificado en ISO 7816-4 [6], pero el contenido y significado son específicos de la aplicación. Las unidades de datos APDU encargadas de portar la información son de dos tipos: comando, encargado de transferir datos de la aplicación externa a la tarjeta, y respuesta, enviada de tarjeta al lector (tablas 2 y 3).

CLA	INS	P1	P2	Lc	Datos Opcionales	Le

Tabla 2: Comando APDU

Datos Opcionales	SW1	SW2

Tabla 3: Respuesta APDU

Cada comando está formado por:

- Un byte **CLA** que indica la clase de instrucción. Por ejemplo, referencia el tipo de seguridad que se emplea.
- Un byte de instrucción, **INS**, que determina el comando enviado.
- Dos bytes, **P1** y **P2**, que se utilizan como parámetros específicos del comando enviado.
- Un byte **Lc** especificando la longitud de los datos enviados a continuación, si los hubiera.
- Una serie de bytes de longitud variable con datos opcionales.
- Un byte **Le** con la longitud prevista de los datos en la APDU de respuesta.

La respuesta contiene:

- Datos opcionales como respuesta al comando anterior.
- Dos bytes de estado, **SW1** y **SW2**, conocidos como *Status Word*, que indican el resultado de la operación ejecutada. El valor hexadecimal 90 00, por ejemplo, indica que el comando fue ejecutado con éxito.

Las APDUs son transmitidas por el protocolo de nivel inferior a través de TPDUs (*Transmission Protocol Data Unit*). En ISO 7816-3 se especifican varios protocolos de transmisión para este nivel, dos de los cuales son los más utilizados en la actualidad. El primero, T=0, es un protocolo orientado a byte (cada byte es transmitido de forma independiente). Se emplea un bit de paridad como método de detección de errores, siendo todo byte erróneo retransmitido. El segundo, T=1, es un protocolo orientado al bloque (paquetes de bytes). Este modo de intercambio permite control de flujo en ambas

direcciones, con lo que la tarjeta puede tomar el control de la comunicación. T=0 es un protocolo cuyas ventajas residen en su sencillez y es empleado por la mayoría de las aplicaciones. Los lectores implementan ambos protocolos, siendo usado el que se determine a partir del ATR.

Como cualquier otro modelo de transmisión, la comunicación tarjeta-dispositivo lector se puede representar por medio de una pila de protocolos. En la tabla 4 se muestra esta pila de protocolos y su equivalencia con el sistema OSI.

Capas OSI	ISO 7816
Capas superiores	Protocolos propietarios
Transporte Red	APDU
Enlace	T = 0 ó T = 1
Física	ISO 7816-3

Tabla 4: Comparativa OSI vs protocolos de tarjeta inteligente

2.5. Protocolo PCSC

PC/SC (Personal Computer/Smart Card) [7] es un estándar para el uso de tarjetas inteligentes en sistemas operativos Windows, desarrollado por el PC/SC Workgroup, que trata de proporcionar una API de alto nivel para acceder a las tarjetas inteligentes y a los lectores de manera homogénea. Tiene la ventaja de que las aplicaciones no necesitan conocer los detalles de un lector para poder utilizarlo y de que una aplicación que sigue este estándar funciona con cualquier lector compatible con el mismo. La estructura de la arquitectura de PC/SC se muestra en la figura 2.6, que representa la jerarquía de los programas y elementos de hardware, desde el usuario hasta la propia tarjeta. La aplicación es la que ofrece al usuario el interfaz de comunicación, y accede al lector por medio del gestor de recursos, que se encarga de controlar qué lectores tiene disponibles el PC y si éstos tienen o no una tarjeta conectada. Para poder hacer uso de los lectores es necesario que el controlador correspondiente esté instalado, así como el del dispositivo de entrada/salida, pues un lector puede conectarse de diversas formas, ya sea de manera inalámbrica, por USB o puerto serie.

Siendo PC/SC un entorno ideado para su uso en entornos Microsoft, quedan fuera de su ámbito de actuación otros múltiples entornos como Linux. El proyecto M.U.S.C.L.E. (*Movement For The Use Of Smart Cards In A Linux Environment*) surge con el objetivo de proveer *middleware* para tarjetas inteligentes en una gran variedad de plataformas, incluidos sistemas operativos GNU/Linux [8]. M.U.S.C.L.E. sigue el estándar PC/SC para lo que emplea un conjunto de librerías de código abierto denominadas PCSCLite. Estas librerías proporcionan un conjunto de herramientas para trabajar con las tarjetas inteligentes, que se comentarán brevemente en el capítulo 4.

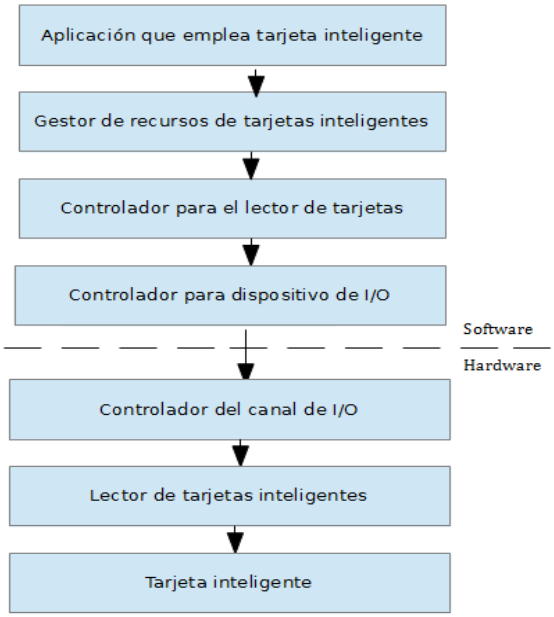

Figura 2.6: Estructura de la arquitectura de PC/SC

3.Tecnologías sin contacto

Tradicionalmente, los periféricos de los ordenadores y otros dispositivos electrónicos se han conectado físicamente, mediante cables. Estas conexiones, aunque fiables, presentan obvios inconvenientes. La movilidad de un periférico unido a un PC mediante un cable no es comparable a la de uno inalámbrico, el tener que tender cables resulta incómodo y además pueden ser causa de accidentes. Aparte de la comodidad, otra razón para la eliminación de los cables es la apariencia y limpieza de lugares de trabajo, habitaciones, etc. Es por ello que cada vez más dispositivos se conectan mediante diversas tecnologías inalámbricas, como los ratones y teclados Bluetooth, las impresoras con conexión por Wifi, o los mandos para juegos de Microsoft con sus protocolos privativos.

De la misma manera que los periféricos tradicionales, las tarjetas inteligentes buscan la comodidad y la inmediatez que ofrece la tecnología inalámbrica, y además cuentan con la gran ventaja de no necesitar baterías, ya que reciben la energía del propio lector. Estas tarjetas frecuentemente se denominan etiquetas o *tags* NFC .

A lo largo de este capítulo se expondrán algunas de las clases de tarjetas sin contactos disponibles en el mercado, haciendo especial hincapié en las tarjetas MIFARE Classic, que es el tipo de tarjeta empleada en este trabajo. Se tratará la tecnología NFC, sus campos de aplicación y los estándares que la regulan.

3.1. Tarjetas inteligentes sin contacto

A continuación se comentan dos de los tipos de tarjetas inteligentes sin contactos más extendidas, que son la familia de tarjetas MIFARE, de NXP, y FeliCa, de Sony.

3.1.1. La familia MIFARE

MIFARE [9] es una de las tecnologías de tarjeta sin contactos más extendidas. Es propiedad de NXP Semiconductors, una antigua rama de Philips Electronics, y está basada en el estándar ISO/IEC 14443 Tipo A. Tiene una distancia típica de lectura/escritura de 10 cm. Se emplea en más del 80% de las tarjetas sin contactos de hoy en día y tiene múltiples aplicaciones, tales como peajes, gestión de accesos, tarjetas de regalo... .

Es común ver referencias a MIFARE como tecnología de tarjeta inteligente, aunque en realidad es una tarjeta de memoria que permite además ciertas operaciones sencillas sobre bloques de datos.

Existen distintos tipos de tarjetas MIFARE dependiendo del tamaño de la memoria, capacidad de operación y tipo de cifrado [10]:

- **MIFARE Classic** es un tipo de tarjeta de memoria disponible en tamaños de 1K y 4K. Se comentará en profundidad a continuación.
- **MIFARE DESFire** es similar a MIFARE Classic pero viene con el sistema operativo DESFire preinstalado, que añade protocolos de seguridad según el modelo. Existen 4 disponibles: uno con Triple_DES [11], y tres con AES de 2, 4 y 8K [12].
- **MIFARE Plus** supone una actualización con respecto a la tarjeta MIFARE Classic, sobre todo en temas de seguridad. La gestión de la información se realiza de la

misma manera, pero la gestión de seguridad se realiza de manera diferente y no es compatible con todos los lectores.

- **MIFARE Ultralight** es un tipo de tarjeta con memoria muy pequeña (512 bits), y que dado su bajo coste se emplea en ocasiones como tarjeta desechable, por ejemplo para entradas de eventos deportivos.

Para el presente trabajo se emplea la conocida como tarjeta estándar, la **MIFARE Classic 1K**, que es ampliamente utilizada en aplicaciones tales como cartera electrónica, control de acceso, identificación en entornos corporativos, transportes y *ticketing*. Este tipo de tarjeta, como indica su nombre, tiene 1 KByte de memoria para el almacenamiento de datos, dividido en 16 sectores, cada uno dividido a su vez en 4 bloques de 16 bytes (figura 3.1). Cada uno de estos sectores está protegido por dos claves diferentes, A y B.

Cada sector se compone de tres bloques dedicados al almacenamiento de información y un cuarto bloque, denominado *trailer*, que contiene las condiciones de acceso a cada uno de los bloques del sector, junto con las claves A y B (ver tabla 5).

Nº de byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Descripción	Clave A						Bits de acceso				Clave B (opcional)					

Tabla 5: Tráiler del sector

Para poder realizar cualquier operación sobre la tarjeta es necesario una autenticación mutua previa y una aseguración del canal. Para ello se realiza un triple intercambio según la norma ISO 9798-2 [13], empleando mecanismos de cifrado simétrico. Haciendo uso de las claves alojadas en la tarjeta, se genera un número aleatorio que se envía al lector. Éste responde a su vez con un mensaje de confirmación. La operación se realiza tres veces, y en caso de autenticación correcta se garantiza el acceso al sector solicitado.

En cuanto a las operaciones sobre bloques de datos, estas tarjetas permiten realizar sumas y restas de manera atómica (permiten restaurar valor) sobre la información almacenada. Dicha información debe estar dispuesta según el formato prefijado de un bloque de valor (tabla 6). Esta estructura permite la detección y corrección de errores, así como el soporte para copia de seguridad.

Nº de byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Descripción	Valor				$\overline{\text{Valor}}$				Valor				Adr	$\overline{\text{Adr}}$	Adr	$\overline{\text{Adr}}$

Tabla 6: Bloque de valor

3.1.1.1 Seguridad en tarjetas MIFARE

El sistema de seguridad de las tarjetas MIFARE autentica la tarjeta frente al lector y viceversa, pero por sí mismo no autentica al poseedor de la tarjeta. La validación del poseedor debe realizarse con métodos adicionales haciendo uso de PIN, claves, etc., que puedan estar almacenados en la propia tarjeta, o en entidades externas.

Gran parte de la seguridad del sistema de autenticación en una tarjeta MIFARE reside en la robustez de las claves empleadas. No obstante, el sistema es vulnerable a ataques de fuerza bruta, ya que no dispone de ningún método que impida el acceso tras un

determinado número de intentos fallidos. Debido a esto, es conveniente cambiar la clave de unas tarjetas a otras, para que así el obtener una clave no proporcione acceso a la información residente en otras tarjetas.

Aparte de este rudimentario método para eludir la seguridad de estas tarjetas, en los últimos años han surgido diversos métodos para saltarse la seguridad inherente en las tarjetas de la familia MIFARE. En 2008, la Universidad de Nijmegen hizo público un método para clonar y manipular el contenido de una tarjeta MIFARE Classic mediante ingeniería inversa [14]. Los ataques sobre este tipo de tarjetas no necesitan un equipo de grandes características para llevarse a cabo y pueden obtenerse las claves en unos 200 segundos.

En 2011 David Oswald y Christof Paar, de la Universidad de Bochum, Alemania, detallaron cómo llevaron a cabo un ataque exitoso sobre una tarjeta DESFire, en el que fueron capaces de obtener las claves en unas 7 horas [15].

		Byte Number within a Block															
Sector	**Block**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Description**
15	3	Key A						Access Bits				Key B						Sector Trailer 15
	2																	Data
	1																	Data
	0																	Data
14	3	Key A						Access Bits				Key B						Sector Trailer 14
	2																	Data
	1																	Data
	0																	Data
:	:																	
:	:																	
:	:																	
1	3	Key A						Access Bits				Key B						Sector Trailer 1
	2																	Data
	1																	Data
	0																	Data
0	3	Key A						Access Bits				Key B						Sector Trailer 0
	2																	Data
	1																	Data
	0																	Manufacturer Block

Figura 3.1: Mapa de memoria de una tarjeta MIFARE 1k

En cuanto a las tarjetas Ultralight, en 2012 un miembro de la consultoría de seguridad Intrepidus demostró que las tarjetas de este tipo empleadas en los transportes de San Francisco y Nueva Jersey se podían manipular con una aplicación de Android pudiendo cambiar el saldo y obteniendo viajes gratuitos [16].

3.1.1.2 Coste de algunos tipos de tarjeta MIFARE

En la tabla 7 se muestra el precio de algunos tipos de tarjetas MIFARE, lógicamente, cuanto mayores son las prestaciones más elevado es su coste [17].

Tipo	Modelo	Precio ud. <100 [$]	Precio ud. +1000 [$]	Precio ud. +1000 [$]
Sin contactos	Mifare Classic 1K	0.99	0.85	0.77
	Mifare Classic 4K	2.00	1.95	1.50
	Mifare DESFire 4K	2.90	2.76	2.20

Tabla 7: Precios de algunas tarjetas MIFARE

3.1.2. Sony FeliCa

FeliCa es un tipo de tarjeta inteligente propiedad de Sony, usada principalmente como monedero electrónico y en aplicaciones de transporte. Si bien se encuentra muy extendida en países asiáticos, sobre todo en Japón, su alcance es limitado fuera de esta región geográfica. Algunas de las aplicaciones en países occidentales son como tarjetas de estudiante en Estados Unidos o Finlandia (figura 3.2), en algunos casos integradas en plataforma de educación Blackboard [18].

FeliCa se comunica a una frecuencia de 13.56 Mhz con una velocidad de 212 kbps o 424kbps, y emplea codificación Manchester, que la hace tolerante al ruido generado por la fluctuación de distancia entre la tarjeta y el lector. Emplea un método de autenticación mutua muy eficiente que permite que la transacción entre lector y tarjeta se complete en un intervalo aproximado de 0.1 segundos, incluyendo encriptación (figura 3.3).

Figura 3.2: Tarjetas de identificación FeliCa

Uno de los puntos fuertes de esta tecnología es la seguridad, es el primer tipo de tarjeta inteligente en obtener la certificación ISO 15408 [19], también conocido como *Common Criteria Toolkit*. En cada autenticación la clave de seguridad es generada dinámicamente, evitando el problema de escucha intencionada o *eavesdropping*.

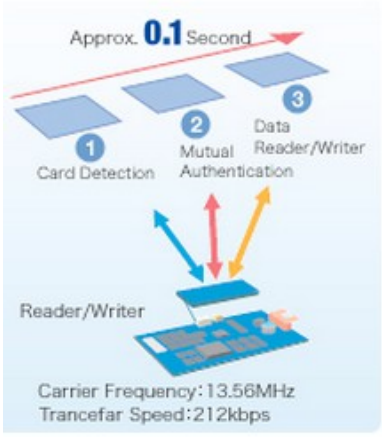

Figura 3.3: Transacción con tarjeta FeliCa

3.2. PC/SC e ISO 14443

El ATR de las tarjetas ISO 14443 no es devuelto de manera automática por la tarjeta al aproximarla al lector, es necesario generarlo. Se trata de una operación sencilla ya que todas las tarjetas ISO 14443 Part 4 tienen un ATR con un formato similar, en el que difiere la longitud de alguno de sus campos. El formato ATR de una tarjeta MIFARE Classic de 1k se muestra en la tabla 8.

Cabecera	T0	TD1	TD2	T1	Tk	Longitud	RID	Standard	Nombre	RFU	TCK
3B	8F	80	01	80	4F	0C	A0 00 00 03 06	03	00 01	00 00 00 00	6A

Tabla 8: ATR de tarjeta MIFARE

Los bytes del campo nombre son los que diferencian el tipo de tarjeta. Los campos nombre para algunas tarjetas ISO 14443 se muestran en la tabla 9.

Asimismo, los APDUs transmitidos al lector no pueden mandarse directamente a la tarjeta, deben ser interpretados y emplear funciones específicas para la comunicación. En este caso concreto, los APDUs que se han empleado para comunicar con la tarjeta MIFARE de 1K se muestran en las tablas 10 y 11.

Nombre	Tipo
00 01	MIFARE 1k
00 02	MIFARE 4k
00 03	MIFARE Ultralight
00 26	MIFARE Mini
F0 04	Topaz and Jewel
F0 11	FeliCa 212k
F0 12	FeliCa 424k

Tabla 9: Campo nombre para distintas tarjetas ISO 14443 Part 4

Comando	CLA	INS	P1	P2	Lc	Datos Opcionales	Le
Cargar Clave	FF	82	Estructura de clave	Número de clave	06	Clave (6 bytes)	-
Autenticar	FF	86	00	00	05	Bytes de autenticación (5 bytes)	
Leer	FF	B0	00	Número de bloque	-	-	Número de bytes a leer
Escribir	FF	D6	00	Número de bloque	Número de bytes a actualizar	Datos (16 bytes)	

Tabla 10: APDUs para MIFARE 1k

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Version 0x01	0x00	Número de bloque	Tipo de clave	Número de Clave

Tabla 11: Bytes de autenticación

La manera en que el APDU se interpreta para comunicar con la tarjeta de detalla en el apartado 5.3.2.

3.3. Near Field Communication

Near Field Communication (NFC) es una tecnología de comunicaciones inalámbrica surgida en 2002 que opera en distancias muy cortas, a altas frecuencias, permitiendo el intercambio de datos entre dispositivos. Debido a su corto alcance (< 10cm), está orientada a intercambios en que los dos extremos están prácticamente en contacto, lo cual no constituye un problema, sino que es en sí una característica deseada. Dado lo limitado del alcance, es muy difícil intervenir una comunicación NFC, lo que traduce en un buen instrumento de seguridad.

Entre los usos de esta tecnología se encuentran:

- **Pago sin contactos.** El usuario realiza el pago sin que exista conexión entre el dispositivo que emplea y la máquina que cobra, como por ejemplo en autobuses.
- **Control de acceso.** El torno de acceso se activa aproximando un dispositivo NFC, como es el caso de ciertas empresas, que facilitan tarjetas con esta tecnología a sus empleados.
- **Conectividad.** NFC puede usarse para transmitir información de manera rápida acercando dos teléfonos, como
- **Control de asistencia**. Se puede emplear una *tag* para registrar la asistencia a una clase o puesto de trabajo.
- **Aplicaciones médicas.** Existen aplicaciones de NFC en el ámbito sanitario, para seguimiento de pacientes, recetas, etc., como Sesame Vitale [20]en Francia.

- **Publicidad.** Mediante pósteres inteligentes se puede informar de un producto o servicio a un usuario que acerque su teléfono.

Figura 3.4: Pago mediante NFC

Seguidamente se profundizará en los detalles técnicos que hacen posible estas aplicaciones.

3.3.1. Estándares

NFC es una tecnología descrita por los protocolos NFCIP-1 y NFCIP-2. El primer protocolo está estandarizado según la ISO 18092 [21], ECMA 340 [22] y ETSI TS 102 190 [23], y especifica sus capacidades básicas, tales como velocidad de transferencia, esquemas de codificación de bits, modulación, y protocolo de transporte. Además se describen los modos activo y pasivo y las precauciones necesarias para prevenir colisiones durante la fase de inicialización. El segundo está estandarizado según la ISO 21481 [24], ECMA 352 [25] y ETSI TS 102 312 [26] y permite elegir entre los tres siguientes modos de operación:

- Transferencia de datos NFC (NFCIP-1).
- Dispositivo de acoplo cercano (PCD, Proximity Coupling Device), definido en la ISO
- 14443.
- Dispositivo de acoplo en los alrededores (VCD, Vicinity Coupling Device), definido en la ISO 15693 [27].

Para que un dispositivo NFC sea compatible con los principales estándares de tarjetas inteligentes, debe ser acorde a tres estándares, que son:

- ISO 14443, entre las cuales se encuentran las tarjetas MIFARE.
- ISO 15693 [27].
- Sistema FeliCa (ISO 18092 [21]), que pertenece a Sony y se usa principalmente en tarjetas monedero.

Por tanto, se puede considerar que NFC agrupa en cierta manera las tecnologías de tarjetas inteligentes y de interconexión en proximidad.

El **NFC Forum** es una asociación de empresas sin ánimo de lucro que promueve la interacción inalámbrica a corta distancia por medio de NFC en la electrónica de consumo, teléfonos móviles y PCs. Fue formado en 2004 para contribuir al avance de la tecnología NFC, desarrollar estándares, asegurar la interoperabilidad entre dispositivos y servicios, y educar al mercado acerca de esta tecnología. Los objetivos de esta entidad son [28]:

- Desarrollar especificaciones basadas en estándares que definan una arquitectura modular y unos parámetros de interoperabilidad para dispositivos y protocolos NFC.
- Promover el desarrollo de productos que empleen las especificaciones del NFC Forum.
- Asegurar que los productos que anuncian capacidades NFC estén de acuerdo a las especificaciones del NFC Forum.
- Educar a consumidores y empresas de manera global acerca de NFC.

Para la consecución de estos objetivos, el NFC Forum define las siguientes especificaciones [29]:

- **LLCP (Logical Link Control Protocol)**: define un protocolo de la capa 2 OSI para dar soporte a la comunicación *peer-to-peer*. Permite servicios tanto orientados a conexión como no orientados a conexión.
- **NDEF (NFC Data Exchange Format)**: permite el intercambio entre dispositivos con una relación *peer-to-peer.*
- **Tag Type Technical Specifications**: se definen cuatro tipos de *tags* para establecer su compatibilidad con los dispositivos NFC de los distintos fabricantes y asegurar así una experiencia de usuario consistente.
- **RTD (Record Type Definition)**: provee una forma de definir eficientemente los tipos de registro para nuevas aplicaciones y proporciona a los usuarios la oportunidad de crear sus propias aplicaciones de acuerdo a las especificaciones del NFC Forum.
- **Connection Handover**: define la estructura y secuencia de interacciones que permiten a dos dispositivos NFC establecer una conexión empleando otro tipo de tecnología, como Bluetooth o Wifi.

3.3.2. Interoperabilidad

NFC representa una evolución convergente de distintos estándares inalámbricos hacia la meta global de interoperabilidad. Para alcanzar esta meta son necesarios distintos modos de comunicación, dependiendo de los dispositivos empleados. En la figura 3.5 se muestran los distintos componentes de la arquitectura técnica de NFC, que se comentan a continuación:

- ***Peer-to-peer* (entre iguales)**: en este modo de comunicación, los dispositivos NFC intercambian datos entre sí, para, por ejemplo, emparejarse para establecer una comunicación Bluetooth ahorrando el engorroso proceso de hacerlo a mano.

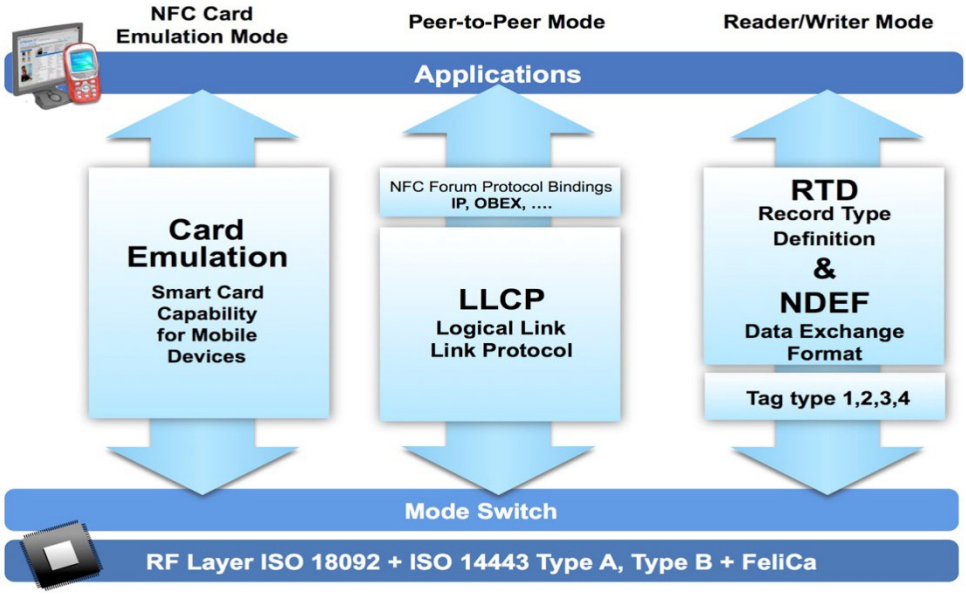

Figura 3.5: Componentes de la arquitectura técnica de NFC

- **Lectura/escritura**: en este modo se transmite información acorde a los estándares NDEF o RTD, por ejemplo en aplicaciones de publicidad, en que acerca un teléfono a un póster con un área resaltada que contiene un chip NFC.
- **NFC Card Emulation**: en este modo el terminal NFC actúa como si fuera una tarjeta, con lo que puede ser leído o escrito por otro terminal que esté en modo lectura/escritura. Tiene la gran ventaja de que el dispositivo no necesita estar encendido, lo que en una época en que una gran carencia de los terminales es la autonomía supone una gran ventaja.

3.3.3. Single Wire Protocol (SWP)

El protocolo SWP es la especificación para la conexión de una tarjeta SIM con un elemento NFC, de manera que permite el uso de la SIM como almacén de servicios distintos a los de telefonía. La especificación la realiza la ETSI [30] y existen recomendaciones proporcionadas por la GSM Association [31].

El uso de este protocolo requiere la existencia de una antena de proximidad en el dipositivo, un controlador NFC conectado al SE, una conexión estandarizada entre el SE y el controlador NFC y un procesador banda base con las funciones necesarias para las aplicaciones NFC.

Las aplicaciones que emplean SWP pueden hacer uso del elemento seguro o del UICC para realizar las operaciones de manera segura y fiable. Como ejemplo, si se usa el teléfono para pagar en un control de acceso, en modo *card emulation*, el SE se comporta como una tarjeta, realizando las operaciones igual que una tarjeta de proximidad física.

4.Controlador inalámbrico PC/SC

Tras estudiar la tecnología de tarjeta inteligente y su vinculación con NFC, es el momento de comenzar con el desarrollo de la solución que permita acceder a una tarjeta inteligente desde cualquier ordenador empleando como lector un teléfono móvil con capacidades NFC.

Dicho desarrollo se puede desglosar en dos partes principales, un controlador para el ordenador y otro para el móvil. Ambos se relacionarán mediante un protocolo de comunicaciones a definir también en el ámbito de este proyecto.

Este capítulo se centra en el estudio práctico del entorno de PC, buscando familiarizarse con la operativa de las tarjetas inteligentes y la arquitectura PC/SC. A partir de este análisis se plantean, definen e implementan las funcionalidades residentes en el ordenador, dejando la descripción del desarrollo en el entorno del teléfono móvil para el siguiente capítulo.

Para comunicar el terminal con el PC se ha elegido emplear Bluetooth, dado que el interfaz de comunicaciones Bluetooth se encuentra hoy en día presente en la práctica totalidad de teléfonos móviles, tablets y PCs. El perfil de comunicaciones entre dos puntos que proporciona Bluetooth se ajusta a las necesidades del proyecto. En la Figura 4.1 se ilustra la arquitectura del sistema completo. Comparando con la arquitectura genérica PC/SC, el controlador del lector a desarrollar se corresponde con los bloques de "controlador para lector de tarjetas" y "controlador para dispositivo de entrada/salida". El primero hace las veces de interfaz con el API PC/SC mientras que el segundo implementará la comunicación inalámbrica Bluetooth con el dispositivo móvil.

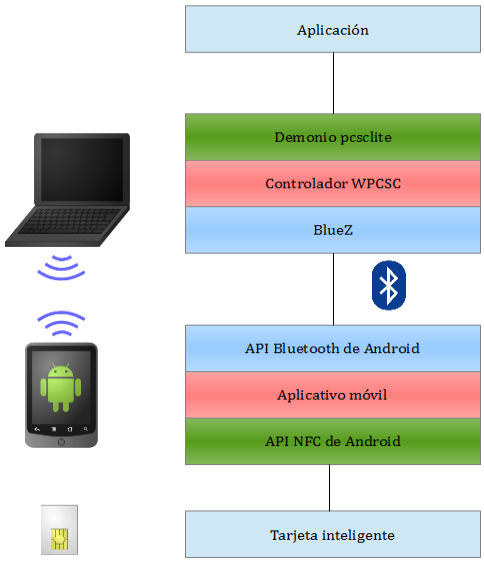

Figura 4.1: Arquitectura del sistema

Los pasos seguidos y que a continuación se describen en mayor profundidad son:

- Desarrollo de un **aplicativo** para PC que permita operar con tarjetas inteligentes. La aplicación se plantea de manera genérica para ser independiente del lector empleado.
- Desarrollo de un **controlador para un dispositivo virtual**. Según esto se pretende emular la presencia de un lector, con el objetivo de aprender cómo funciona el sistema PCSC, y qué funciones tiene que implementar el controlador, así como la tarea que realiza cada una de ellas.
- Desarrollo e integración de un **cliente Bluetooth**, al controlador de un dispositivo virtual, sirviendo de habilitador de intercambio de mensajes con el terminal móvil.
- Diseño e implementación del **protocolo de comunicaciones**, que encapsulan los diferentes comandos que se intercambian con el terminal móvil y que incluyen los mensajes de control y APDU a ejecutar sobre la tarjeta inteligente. El protocolo propuesto se ha denominado Wireless PC/SC (WPCSC).

4.1. Operativa con tarjeta inteligentes

Antes de profundizar en la operativa de bajo nivel de un entorno PC/SC es necesario conocer la operativa básica de comunicaciones con una tarjeta inteligente. Seguidamente se tratará el desarrollo de un aplicativo para realizar la lectura, escritura y autenticación sobre tarjetas sin contactos. Este proyecto se centra en el uso de tarjetas de tipo MIFARE, modelo más extendido actualmente. La aplicación desarrollada se corresponde con el nivel más alto de la arquitectura PC/SC (ver figura 2.6).

Aunque la aplicación desarrollada tenía inicialmente el objetivo de comprender la operativa con tarjetas inteligentes, a lo largo del proyecto dada la independencia del lector subyacente que se emplee, se ha usado esta aplicación para ir validando los diferentes pasos en el desarrollo que se han ido acometiendo.

La aplicación en cuestión se desarrolló primero en el lenguaje Java, ya que éste dispone de una API sencilla y rápida de aprender. Posteriormente se rehízo en lenguaje C, con el fin de aprender más sobre el funcionamiento de las tarjetas y las distintas opciones disponibles a la hora de hacer aplicaciones. Adicionalmente el uso del lenguaje C permite un mayor control sobre la operativa y un mayor nivel de depuración ya que se trabaja con librerías nativas, evitando así problemas de funcionamiento que puedan estar vinculados a librerías de alto nivel como Java. Adicionalmente la programación en C permite emplear la librería nativa PC/SC de comunicaciones contra una tarjeta, lo que permite una mejor comprensión de la operativa del sistema. Finalmente, con el fin de ofrecer una experiencia de usuario más amigable, se ha optado por crear una interfaz gráfica de usuario (GUI) en Java sencilla e intuitiva que evita tener que realizar todas las operaciones desde una línea de comandos.

4.1.1. Entorno de desarrollo

Como sistema operativo para PC se ha elegido Ubuntu, una distribución de GNU/Linux basada en Debian que se distribuye como software libre, si bien permite la instalación de software privativo. Es desarrollado por Canonical, compañía que también ofrece soporte y formación a empresas. Ubuntu busca, ante todo, la sencillez de uso, el rendimiento incluso en sistemas de prestaciones modestas, y la seguridad. Se trata de la distribución GNU/Linux más extendida a nivel de usuario, aunque también tiene cierta

relevancia en servidores y computación en la nube [32] [33]. Éste ha sido uno de los motivos de la elección, ya que cuenta con buen soporte de la comunidad de usuarios, facilitando la resolución de problemas. Además, todas las herramientas empleadas existen para Ubuntu, tales como el SDK de Java o el IDE Eclipse, que se comentará brevemente en el siguiente capítulo.

Como lector de tarjeta inteligente se ha empleado el modelo ACR 122U de ACS, que se puede ver en la figura 4.2. Se trata de un lector/escritor para PC que permite el acceso a tarjetas ISO 14443-4 tipo A y B, MIFARE, ISO 18092 o NFC y FeliCa. El ACR 122U sirve como dispositivo intermedio entre un PC y una tarjeta sin contactos por medio del interfaz USB. Este lector sigue el estándar PC/SC para comunicar con las tarjetas por medio de APDUs [34].

Las librerías instaladas para trabajar con tarjetas inteligentes en este entorno han sido:

- **pcscd**: demonio encargado de gestionar los lectores en tiempo de ejecución y controlar las conexiones a éstos. Implementa los interfaces necesarios para permitir el uso de la API PC/SC desde cualquier aplicación de usuario y gestionar las tarjetas conectadas a los lectores.
- **pcsc-tools**: conjunto de utilidades que permiten acciones como mostrar los lectores de tarjetas conectados, comprobar la presencia de una tarjeta y establecer comunicación con ésta.
- **libacr38u**: se trata de un controlador para el lector ACR38U, que también funciona con el modelo 122U.

Figura 4.2: Lector ACR 122U

En cuanto a la tarjeta NFC, se ha empleado una MIFARE Classic 1KB, fabricada por Philips (ver apartado 3.1.1).

4.1.2. Funciones a implementar

Dado que el objetivo del aplicativo es servir de toma de contacto con el entorno y conocer las posibilidades de las tarjetas inteligentes y del sistema PC/SC, se plantea el desarrollo de las funcionalidades básicas que todo aplicativo de control de tarjeta inteligente sin contacto MIFARE debe soportar. Entre las operaciones implementadas se encuentran:

- **Carga de claves:** para poder efectuar las operaciones deseadas con las tarjetas hay que cargar previamente las claves de autenticación en el lector. Por defecto la aplicación tiene cargada la clave FF FF FF FF FF FF, que es la que se ha usado en la tarjeta de pruebas, tanto para la clave A como la B. Si se especifica lo contrario se cambia el valor de la clave.
- **Autenticación:** una vez cargada la clave se autentica el sector sobre el que se quiere operar con la misma. Para ello, se selecciona el bloque al que se quiere acceder. Una vez autenticado un sector, se pueden realizar operaciones en sus bloques hasta la desconexión de la tarjeta.
- **Lectura:** permite escribir el contenido de un bloque de la tarjeta. Requiere una autenticación previa.
- **Escritura:** permite leer un bloque de la tarjeta. Requiere una autenticación previa.

4.1.3. Interfaz de control

La interfaz diseñada consta de 3 paneles para la entrada de datos. En la figura 4.3 se muestra el aspecto que tiene tras su arranque:

- **Entrada de datos**: en este panel se introduce la información que se quiere escribir en un bloque de la tarjeta, como máximo 32 caracteres hexadecimales, pues el tamaño de bloque es de 16 bytes.
- **Número de bloque**: es el número del bloque sobre el que se desea operar, en hexadecimal.
- **Clave**: incluye los 6 bytes de la clave a emplear para la autenticación. El selector inferior determina si la clave es de tipo A o B.

Cuando se pulsa el botón de una de las operaciones, el programa recoge la información de los paneles y genera los comandos correspondientes para cometer las siguientes acciones sobre la tarjeta:

- **ATR:** Devuelve el ATR (*Answer To Reset*) de la tarjeta conectada.
- **Cargar clave**: carga la clave escrita bajo *introduzca la clave* en el lector.
- **Autenticar**: autentica el bloque indicado en *número de bloque* con la clave que haya cargada en el lector.
- **Leer Bloque**: lee el bloque que aparece en el campo *número de bloque*.
- **Escribir bloque**: escribe el contenido del campo *entrada de datos* en el bloque que indica el campo *número de bloque*.
- **Salir**: cierra la aplicación.

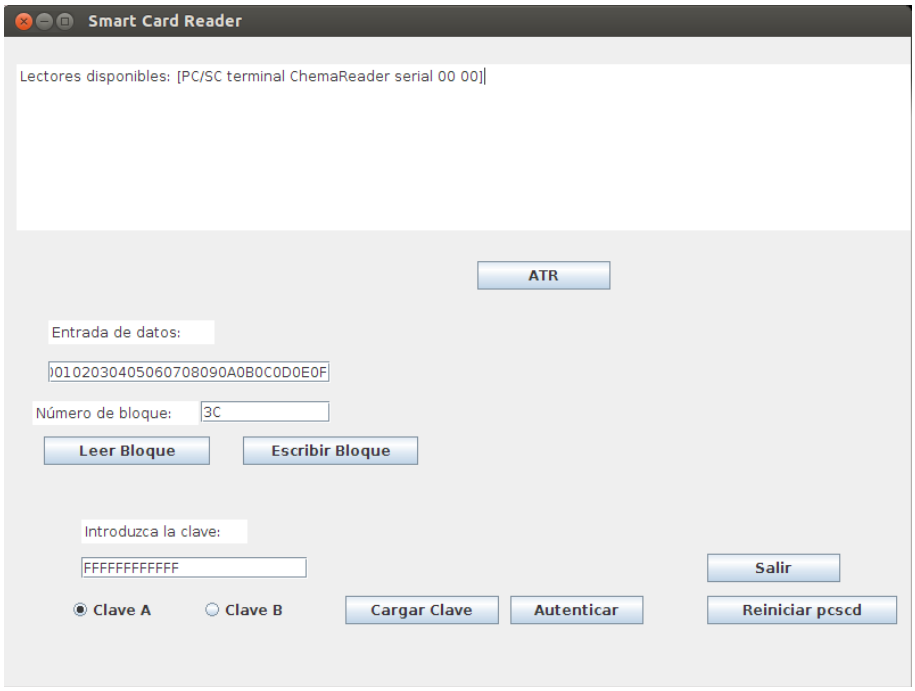

Figura 4.3: GUI de la aplicación

Hasta ahora el trabajo realizado se ha basado en un lector de tarjeta inteligente cuyo controlador de comunicaciones PC/SC se encuentra ya disponible. Al igual que otros muchos lectores de tipo CCID (*Chip/Smart Card Interface Devices*), el trabajo con ellos es transparente, operando correctamente tras conectarlos al equipo.

No obstante, el proyecto pretende emplear un terminal móvil como elemento lector. En los siguientes apartados se tratará el funcionamiento y diseño del controlador empleado para reconocer el dispositivo móvil como lector de tarjetas, así como el protocolo de comunicación propuesto para permitir la conexión inalámbrica entre el equipo de control y el elemento que haga las veces de lector de tarjeta inteligente.

4.2. Dispositivo virtual

Tras haberse familiarizado con el entorno PC/SC a nivel de usuario se acomete a continuación el estudio en profundidad de los diferentes niveles de la arquitectura PC/SC, en particular los niveles más bajos y cercanos a la interacción con el lector.

Con el fin de analizar el funcionamiento de la pila PC/SC, se plantea el desarrollo de un controlador para un lector de tarjetas virtual o *dummy driver*. Dicho controlador emula la presencia de un lector conectado al puerto serie con una tarjeta insertada. El disponer de un controlador propio permite analizar qué funciones del controlador son ejecutadas para cada una de las acciones que se invocan desde el API.

Para desarrollar el controlador se han tomado como base los prototipos de las funciones de un controlador de lector de tarjeta inteligente proporcionado por M.U.S.C.L.E.

Las funciones básicas a implementar por el desarrollador son las siguientes:

- **IFDHCreateChannelByName (DWORD Lun, DWORD Channel)**: esta función se encarga de abrir un canal de comunicaciones al puerto indicado por *Channel*. *Lun*

(Logical Unit Number) se emplea para distinguir entre múltiples *slots* o tarjetas, si los hubiera. Una vez abierto el canal, el lector debe quedar en un estado en que se pueda ejecutar la función *IFDHICCPresence*. Puede devolver *IFD_SUCCESS* ó I*FD_COMMUNICATION_ERROR,* dependiendo de que la operación se realice correctamente o no. En general, esta es la respuesta de todas las funciones, a no ser que se indique lo contrario.

- **IFDHCloseChannel()**: cierra el canal de comunicaciones con el lector
- **IFDHGetCapabilities(DWORD Lun, DWORD Tag, PDWORD Length, PUCHAR Value)** : obtiene las características de cada *slot* o tarjeta. *Tag* es la etiqueta para la información que se solicita, por ejemplo, para pedir el ATR de una tarjeta, así como el tamaño del mismo, se pasa como argumento *TAG_IFD_ATR*. El valor de la *tag* se define en el código.
- **IFDHPowerICC (DWORD Lun, DWORD Action, PUCHAR Atr, PDWORD AtrLength)**: controla las señales de encendido y reset del lector. *Action* indica la acción a realizar sobre la tarjeta. *Atr* y *AtrLength* son, respectivamente, las variables en donde se almacenan el ATR y su longitud. Devuelve *IFD_ICC_PRESENT* o *IFD_ICC_NOT_PRESENT* en función de que la tarjeta esté disponible o no, y *IFD_COMMUNICATION_ERROR* en caso de error.
- **IFDHICCPresence(DWORD Lun)**: es la función encargada de comprobar el estado de la tarjeta. Puede responder que la tarjeta está presente, no presente, o que se ha producido un error de comunicación. Tras la ejecución de *IFDHCreateChannelByName*, esta función se ejecuta en bucle hasta que se retira la tarjeta o se produce un error.
- **IFDHTransmitToICC (DWORD Lun, SCARD_IO_HEADER SendPci, PUCHAR TxBuffer, DWORD TxLength, PUCHAR RxBuffer, PDWORD RxLength, PSCARD_IO_HEADER RecvPci)**: envía a la tarjeta el comando guardado en *TxBuffer,* con longitud *TxLength,* y almacena una respuesta de longitud *RxLength* en *RxBuffer.*

El controlador debe implementar adicionalmente otras funciones: *IFDHSetCapabilities, IFDHSetProtocolParameters, IFDHControl,* sin embargo, su desarrollo queda fuera del objetivo de este proyecto, aunque, como es necesario, se han incluido retornando el correspondiente mensaje de error.

Para que el PC detecte que el lector virtual está conectado, es necesario escribir un archivo de configuración, que se guarda en la carpeta */etc/reader.conf.d.* Este archivo contiene los siguientes campos:

- **FRIENDLYNAME**: es el nombre con que se identifica el lector de cara al usuario.
- **DEVICENAME**: en este campo se especifica el número de puerto al que se conecta el lector. Como el controlador diseñado no emplea ningún puerto físico, el valor de este campo es *null*.
- **LIBPATH**: es el *path* de la librería dinámica que se obtiene al compilar el código fuente del controlador.
- **CHANNELID**: este campo no es necesario, se emplea en el caso de que se tengan varios lectores conectados físicamente al puerto serie para distinguirlos.

El empleado para este trabajo en concreto es como sigue:

```
FRIENDLYNAME   "Smartcard Reader"
DEVICENAME     /dev/null
LIBPATH     /home/user/smartcard/libgen_ifd.dylib
CHANNELID      1
```

El campo *ChannelID* no es relevante en este caso, ya que no se emplea un canal por el puerto serie del PC, sino por Bluetooth. El archivo de configuración empleado es el mismo siempre, no es necesario modificarlo para que funcione el controlador real.

Resta, por tanto, definir el comportamiento de las funciones. Sin embargo, en un principio se desconocía cuándo se llama a cada una de las funciones del controlador, y por tanto cómo y para qué desempeña cada una de ellas su tarea. Para descubrir la operativa se ha llevado un registro del orden de ejecución de las distintas funciones. De esta manera, se ha podido averiguar qué tareas lleva a cabo el controlador en respuesta a cada una de las acciones del usuario.

Así por ejemplo, al ejecutar el demonio de control, pcscd, en primer plano, leyendo la información de depuración se ha podido comprobar que chequea la presencia de una tarjeta en un bucle infinito hasta que muere. Tomando como referencia la arquitectura mostrada en la figura 4.1, la arquitectura del lector *dummy* corresponde a la parte superior, la del PC, eliminando la pila BlueZ, ya que no se establece comunicación con el exterior.

La primera versión del controlador incluyendo todas las funciones con mensajes de depuración proporcionó información para conocer el flujo de llamadas, así como los posibles valores que los diferentes parámetros, tales como *Lun, Tag, o Length*, pueden tomar en correspondencia con el programa de control que se ha realizado.

Por ejemplo, para simular que una tarjeta está presente en el lector virtual, se completa la función retornando el código de error *IFD_PRESENT*. De esta manera, en cada iteración del bucle, *pcscd* detectará una tarjeta insertada en el lector. Del mismo modo se han adaptado los parámetros de retorno para que la operativa de la aplicación no se viera afectada. De esta forma se ha conseguido que el lector virtual tenga un comportamiento similar al del ACR122U.

Este procedimiento se ha seguido con el resto de las funciones, permitiendo evaluar la comunicación con el dispositivo/lector virtual PC/SC. En la figura 4.4 se muestra un diagrama de comunicación con el dispositivo virtual en el que se incluyen las operaciones de carga de clave en el lector, autenticación, lectura y escritura. El intercambio ilustrado por dicha figura se ha realizado con el aplicativo para PC desarrollado, abriendo un canal de comunicación con una tarjeta y enviando los comandos que se muestran. Se puede observar que la carga de la clave responde éxito en todo momento, pues no hace más que almacenarla en memoria, sin realizar ninguna operación con ella. No obstante, al intentar autenticar un bloque con la clave incorrecta el código devuelto es de error. Posteriormente se carga la clave correcta y la autenticación se resuelve con éxito. Completada la autenticación se puede leer y escribir el bloque.

Aparte de servir para el estudio de PCSC, el *dummy driver* ha sido el punto de partida para el controlador realizado posteriormente para el lector. Si bien la información proporcionada desde M.U.S.C.L.E. está pensada para desarrollar un controlador para un dispositivo conectado al puerto serie, realizando unas modificaciones y adecuando el comportamiento del controlador es posible generar un lector conectado mediante Bluetooth, tal como se describe en el siguiente apartado.

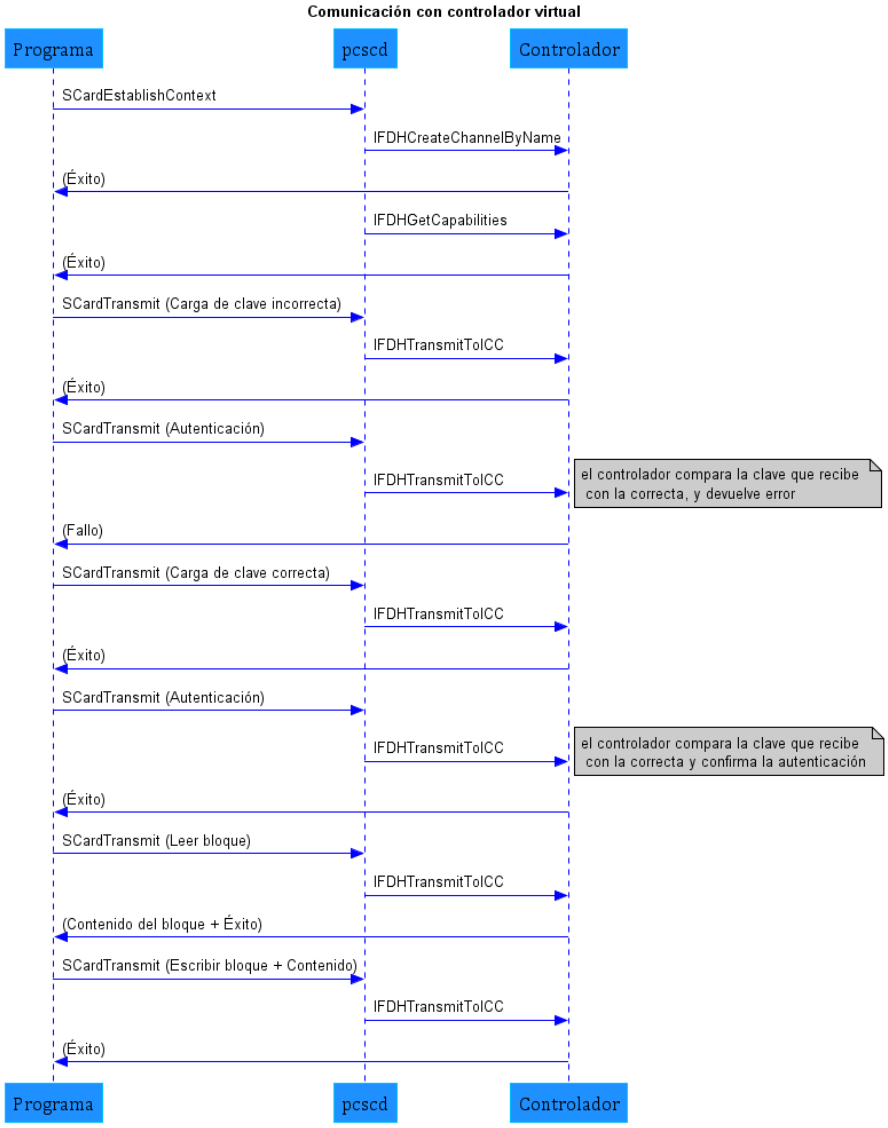

Figura 4.4: Comunicación con el driver virtual

4.3. Protocolo WPCSC

Una vez analizado el comportamiento de un controlador para un lector virtual y disponiendo de un código funcional básico, se han ido incluyendo nuevas funcionalidades completando el código del *dummy driver* para que se comunique con un terminal móvil.

Antes de proceder a realizar el controlador es necesario definir el protocolo de intercambio de los datos entre las dos entidades. A partir del estudio realizado con el *dummy driver* se han extraído las necesidades o requerimientos que debe cumplimentar el protocolo de intercambio. Así debe soportar las siguientes acciones:

- Reportar la presencia de una tarjeta en lector.
- Recoger el ATR de dicha tarjeta.
- Envío de los comandos y respuestas APDU.

- Mantenimiento de la comunicación y reporte de errores.

Para cumplir con estos requerimientos se propone el protocolo WPCSC, que habilita la comunicación entre PC y dispositivo móvil. Dicho protocolo se encapsulará en un canal de comunicación Bluetooth. Para realizar esta acción, se ha extendido la funcionalidad del controlador que permitía establecer una comunicación PC/SC con un lector conectado al puerto serie. De esta forma se crea un canal de comunicación que conecte a un servidor Bluetooth disponible en el dispositivo móvil. Esto supone un avance sobre el controlador virtual, por el que, tal como se refleja en la figura 4.5, el PC utiliza PC/SC y se comunica hacia el exterior empleando Bluetooth. En esta sección se detallan únicamente los componentes cliente alojados en el PC, siendo la parte servidor descrita en el siguiente capítulo. Pero dado que para poder probar la comunicación se necesita tener un servidor, el desarrollo del mismo se ha llevado a cabo en paralelo con el de esta parte. Por tanto, en los diagramas de comunicación se incluirá el terminal para poder detallar el funcionamiento de WPCSC.

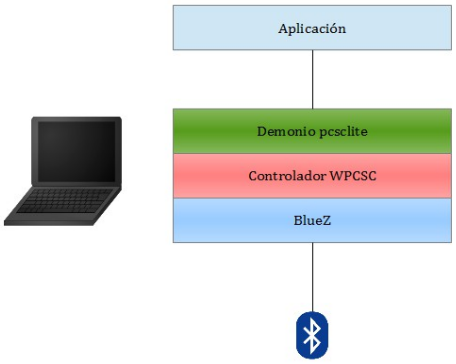

Figura 4.5: Arquitectura de la parte del PC

4.3.1. Primitivas WPCSC

Para el desarrollo del protocolo WPCSC se ha empleado un formato de trama TLV (*Tag, Length, Value*). En la tabla 12 se muestra dicho formato. En algunos tipos de mensajes se han eliminado los campos de longitud y value por compresión, ya que sólo es necesaria la cabecera.

Número de byte	0	1	2	3	...	x
Descripción	TAG	LENGTH	VALUE			

Tabla 12: Formato de trama WPCSC

A continuación se enumeran los distintos formatos de trama propuestos para el protocolo WPCSC. Se enumerarán como trama comando y su correspondiente trama respuesta.

4.3.1.1 Confirmación de conexión

W_PCSC_ACK_CONNECTION (0x00): consta de un solo byte (los campos *length* y *value* tienen longitud 0 en este caso), y se emplea para confirmar al cliente que se ha conectado a un servidor WPCSC.

Valor	W_PCSC_ACK_CONNECTION
Longitud en bytes	1

4.3.1.2 Control de presencia

Trama Comando

W_PCSC_CMD_CARD_PRESENCE (hexadecimal: 0x01): trama para comprobar la presencia de la tarjeta en el lector. Sólo consta de un byte.

Valor	W_PCSC_CMD_CARD_PRESENCE
Longitud en bytes	1

Trama Respuesta

W_PCSC_RESP_CARD_PRESENCE (hexadecimal: 0x02): se envía en respuesta a una trama de la misma cabecera, contiene el tipo de trama y booleano que indica la presencia de la tarjeta, *true* para indicar que está presente, y *false* para indicar que está ausente.

Valor	W_PCSC_RESP_CARD_PRESENCE	LENGTH	BOOL
Longitud en bytes	1	2	1

4.3.1.3 Envío de un comando

Trama comando

W_PCSC_CMD_APDU (hexadecimal: 0x10): este tipo de trama se emplea para mandar un APDU a la tarjeta, consta de una cabecera de 3 bytes que contiene el tipo de comando y la longitud del mismo, y un APDU de tamaño variable.

Valor	W_PCSC_CMD_APDU	CMD_LENGTH	COM_APDU
Longitud en bytes	1	2	CMD_LENGTH

Donde CMD_LENGHT es la longitud del comando APDU.

Trama respuesta

W_PCSC_RESP_APDU (hexadecimal: 0x11): es la trama de respuesta a W_PCSC_CMD_SEND_APDU, tiene el mismo formato que ésta.

Valor	W_PCSC_RESP_APDU	RESP_LENGTH	RESP_APDU
Longitud en bytes	1	2	CMD_LENGTH

Donde CMD_LENGHT es la longitud de la respuesta APDU.

4.3.1.4 Petición del ATR

Trama comando

W_PCSC_CMD_GET_ATR (0x03): se emplea para pedir el ATR de la tarjeta conectada. Sólo consta de un byte. Por coherencia debería tener un campo de longitud de valor 0, no obstante, para comprimir la trama se ha eliminado.

Valor	W_PCSC_CMD_GET_ATR
Longitud en bytes	1

Trama respuesta

W_PCSC_RESP_GET_ATR (hexadecimal: 0x04)**:** se envía en respuesta a una trama de la misma cabecera, se añaden 2 bytes para indicar la longitud del ATR y un número variable conteniendo el propio ATR.

Valor	W_PCSC_RESP_GET_ATR	ATR_LENGTH	ATR
Longitud en bytes	1	2	ATR_LENGTH

Donde ATR_LENGTH es la longitud del ATR.

4.3.2. Operativa de WPCSC

Como requisito para poder usar el terminal móvil como lector se ha establecido que ambos dispositivos han de ser emparejados, haciendo uso de las herramientas que proporciona cada sistema operativo. En el caso que concierne se ha empleado la librería BlueZ, pila de Bluetooth oficial de Linux [35], para establecer la comunicación. El servidor móvil ejecutará un servicio Bluetooth específico para su operativa como lector, al que el PC se conectará tras haberse emparejado. Se emplea el protocolo SDP (*Service Discovery Protocol*), que necesita que el cliente conozca dos parámetros:

- **Dirección MAC (*Media Access Control*)**: se trata de un conjunto de 6 bytes que identifica unívocamente un dispositivo o tarjeta de red.
- **UUID** (*Universal Unique Identifier*): es un identificador estandarizado por la OSF (*Open Software Fundation*) como parte del *Distributed Computing Environment*. El propósito del UUID es ofrecer un identificador "prácticamente único" en sistemas de computación distribuida. Es posible que los identificadores coincidan, pero dada la cantidad de combinaciones posibles es algo altamente improbable.

Conocidos los dos, el controlador se conecta a la dirección MAC del dispositivo en que se ejecuta el servidor. Una vez conectado realiza una búsqueda entre las UUID de todos los servicios disponibles. Si el resultado de dicha búsqueda es satisfactorio se establece la conexión. Para evitar el improbable aunque posible caso en que alguien trate se suplantar la identidad de un dispositivo móvil servidor empleando el mismo UUID, el servidor remite una secuencia al cliente, tras cuyo asentimiento se confirma que el servicio al que se ha conectado es un lector WPCSC.

Una limitación del sistema se encuentra en una característica del demonio *pcscd*. Al haber diseñado el controlador del lector como si se tratase de un dispositivo serie, el demonio no es capaz de detectar su desconexión, puesto que no existe enlace físico entre

el PC y el lector. Esto provoca que el lector quede guardado en memoria, teniendo que reiniciar *pcscd* o esperar a que muera si queremos realizar una nueva conexión. Por tanto, si al arrancar la aplicación en el PC no hay posibilidad de conectarse, es posible que se deba a ésto.

Una vez definido el protocolo, es hora de ver cómo fluye la información. En la figura 4.6 se observa un diagrama de la comunicación entre una aplicación y un terminal móvil que actúa como lector. A continuación se explican los pasos seguidos durante la comunicación:

- En primer lugar, al establecerse el contexto de comunicación se crea el canal Bluetooth, cuando el teléfono acepta la conexión entrante, responde con un ACK (ver apartado 4.3.1.1) para indicar que efectivamente es un servidor WPCSC.
- *pcscd* comienza a comprobar la presencia de la tarjeta invocando *IFDHICCPresence*. La respuesta a esta llamada contiene un campo de información con un valor booleano que vale "0" en caso de que no haya tarjeta y "1" cuando se inserta una. Esta operación se produce continuamente mientras dure la comunicación con la tarjeta.
- **Carga de la clave de autenticación**: cuando la aplicación del PC ejecuta *ScardTransmit*, *pcscd* llama a la función *IFDHTransmitToICC* del driver, la cual encapsula la el APDU en el formato WPCSC, añadiendo tipo de mensaje y longitud, y lo envía al terminal. El terminal desencapsula la trama y responde confirmando que tiene la clave cargada.
- **Autenticación**: el proceso es igual que en el caso anterior, pero en lugar de confirmar la carga de la clave, se realiza una operación sobre la tarjeta, con lo cual la respuesta lleva en su campo *value* el APDU de respuesta que informa de si el bloque ha sido autenticado correctamente o no.
- **Lectura de un bloque**: de nuevo el proceso de comunicación es el mismo que en el caso anterior, pero ahora el campo *value* será considerablemente más largo, ya que transporta todo el contenido de un bloque de la tarjeta.

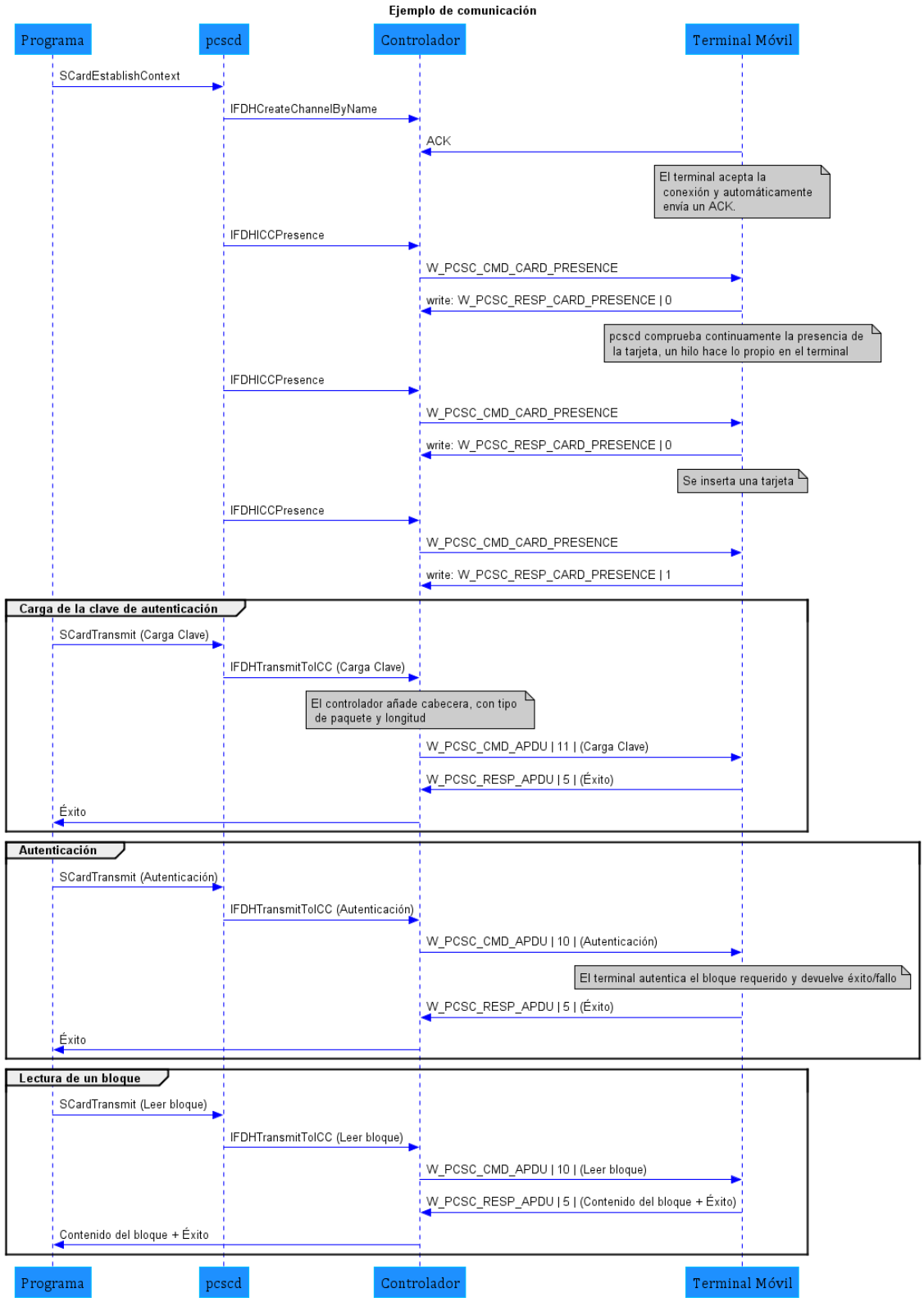

Figura 4.6: Ejemplo de comunicación

5.Desarrollo del aplicativo móvil

Tras haber tratado el entorno del PC, y haber descrito el protocolo de comunicaciones que se ha diseñado, es el momento de introducir el aplicativo desarrollado para los terminales móviles y que hará las veces de servidor WPCSC y de controlador del interfaz NFC del móvil.

El protocolo WPCSC encapsula el conjunto de comandos y respuestas necesarios para habilitar el correcto funcionamiento del controlador PC/SC. En este sentido, y tal como se ha planteado la solución, se define un servidor ejecutándose en el terminal móvil aceptando mensajes de un PC conectado a él. El conjunto de las tres herramientas y el control del interfaz NFC confeccionan el lector PC/SC móvil:

- **Servidor Bluetooth.** Parte encargada de gestionar nuevas conexiones, mantener activo el hilo que las acepta, así como encapsular y desencapsular los APDUs para adecuar los mensajes al protocolo WPCSC.
- **Lector de tarjetas.** Es la parte de la aplicación que recibe los APDUs del servidor, comunica con la tarjeta, devuelve las respuestas, y comprueba la presencia de la tarjeta.
- **Integración con la interfaz de usuario.** Para que el usuario interaccione de manera sencilla con el teléfono y reciba información se han integrado las dos partes anteriores con una interfaz gráfica de usuario.

Este capítulo describirá el trabajo realizado para implementar las tres funcionalidades necesarias, que una vez operativas permitirán el uso del terminal móvil como periférico para la lectura de tarjetas.

El aplicativo se ha desarrollado como aplicación para un teléfono o tableta que funcione con el sistema operativo Android. La elección de esta plataforma se ha basado en la disponibilidad de dispositivos que implementen tanto NFC como Bluetooth y en la apertura que tiene este sistema a los desarrolladores, con un gran número de herramientas.

5.1. Entorno de desarrollo

5.1.1. El sistema operativo Android

Android surge en 2005 de la mano de Android Inc., compañía financiada y posteriormente adquirida por Google [36], y hace su primera aparición como sistema operativo de un teléfono móvil en 2008 [37]. Actualmente su desarrollo es llevado a cabo por Google y la Open Handset Alliance, y su código fuente publicado bajo la licencia Apache.

Android consiste en un kernel basado en Linux más un conjunto de librerías, APIs y otro *middleware* escrito en C. Emplea una máquina virtual de Java, llamada Dalvik, encargada de ejecutar las aplicaciones. Aunque basado en Linux, Android carece del X Window System, que da soporte a las interfaces de usuario en los sistemas Linux, así como de compatibildad con las bibliotecas estándar GNU. Debido a esto, es difícil portar las aplicaciones diseñadas para sistemas GNU/Linux a Android.

Android funciona mayoritariamente sobre procesadores ARM, aunque existen proyectos para dar soporte a sistemas con arquitectura x86, si bien éstos no se encuentran en un estado muy avanzado.

5.1.2. Herramientas

Para el desarrollo de la aplicación se ha empleado el entorno de desarrollo (IDE) Eclipse, desarrollado por la fundación homónima y publicado como software libre bajo la *Eclipse Public License*. Está programado principalmente en Java, y permite la programación en este lenguaje. Además, por medio de *plugins* permite la programación en otros lenguajes, como C, C++, Ruby, etc., así como el desarrollo de aplicaciones para el sistema operativo Android [38]. Para ello es necesario el *plugin* ADT (Android Development Tools) distribuido por Google [39]. Este *plugin* hace uso del SDK (*Software Development Kit*) de Android que debe ser instalado en el equipo. Dicho SDK provee todas las herramientas necesarias para el desarrollo de aplicaciones, tales como los controladores para dispositivos conectados por USB e imágenes del sistema operativo para realizar pruebas con emuladores.

Las herramientas que proporciona Eclipse agilizan el proceso de desarrollo, ofreciendo una interfaz amigable e intuitiva, con ayudas tales como corrección sintáctica o autocompletado.

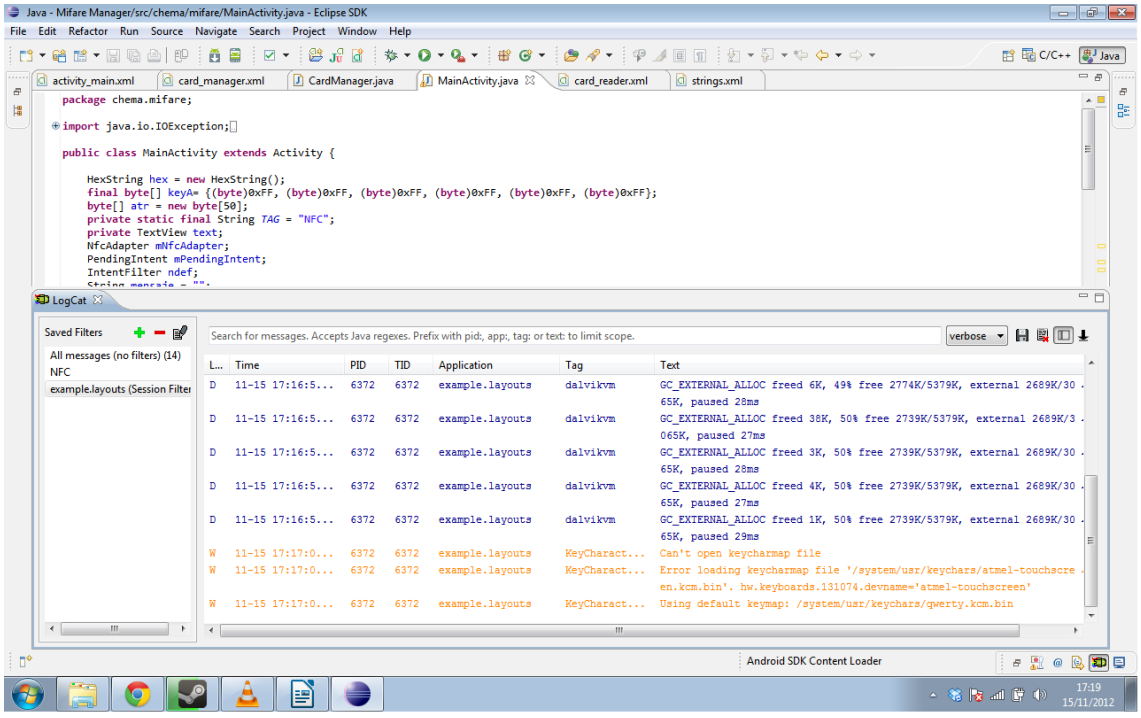

Figura 5.1: Eclipse durante la prueba de una aplicación

En su kit de desarrollo Google incluye una herramienta absolutamente imprescindible, el ADB (*Android Debug Bridge*). Esta herramienta permite instalar y probar la aplicación que se está desarrollando directamente en un dispositivo móvil y visualizar un *log* en el PC en tiempo real. Esto se puede hacer en Eclipse por medio de *logcat*. En la figura 5.1 se aprecia el aspecto que ofrece Eclipse durante la prueba de una aplicación. En el centro de la pantalla hay varias pestañas con los códigos fuente y en la

parte inferior se encuentra abierto *logcat*. A la izquierda aparecen los filtros y a la derecha los distintos mensajes. Obsérvese que el color es diferente en función del tipo de información, lo que ofrece cierta ayuda visual a la hora de buscar algo en concreto, por ejemplo, el salto de una excepción.

5.1.3. Tecnologías NFC soportadas

Android soporta varios tipos de tarjetas inteligentes. El objeto genérico que representa una tarjeta inteligente en una aplicación de Android recibe el nombre de *tag*. Para obtener la tarjeta en la aplicación, se registra un *Intent*, que notificará de la aproximación de una tarjeta. Un *Intent* es un objeto que se utiliza para la comunicación entre componentes de una aplicación [40]. Existe la posibilidad de elegir a qué tipos de tecnologías reaccionará una aplicación, especificándolas en una lista. Todas aquellas que no estén presentes en la lista serán ignoradas.

Cuando el objeto *tag* es creado, es preciso obtener la tecnología concreta con la que se quiera operar, conocida como *TagTechnology*. Por ejemplo, si se dispone de una tarjeta que sea de tipo *MifareClassic* y *NdefFormatable* y se quiere trabajar con la tarjeta MIFARE, habrá que llamar al método *MifareClassic.get()* para obtener la *TagTechnology* correspondiente.

Los tipos de tecnologías soportados de manera obligatoria para los dispositivos Android con funcionalidad NFC son:

- NfcA (ISO 14443-3A)
- NfcB (ISO 14443-3B)
- NfcF (JIS 6319-4)
- NfcV (ISO 15693)
- IsoDep
- Ndef en las *tags* que cumplan NFC Forum Type 1, Type 2, Type 3 o Type 4.

Opcionalmente los terminales pueden proporcionar las siguientes implementaciones de *TagTechnology*:

- MifareClassic
- MifareUltralight
- NdefFormatable

5.2. Planteamiento inicial

Una de las primeras ideas fue la de seguir el proyecto seek-for-android (*Secure Element Evaluation Kit for the Android platform*) [41], que busca un fin parecido al de este trabajo, aunque emplea métodos diferentes. En la figura 5.2 se muestra la solución propuesta por dicho proyecto.

Este proyecto tiene algunas opciones interesantes, como son el acceso al elemento seguro del teléfono, que permite endurecer las medidas de seguridad de aplicaciones de pago o identificación.

A pesar de las opciones adicionales en cuanto a seguridad que ofrece el acceso al elemento seguro, el proyecto tiene múltiples inconvenientes. El más notable es la dificultad para instalar el sistema. Aunque en la parte de PC, la solución ofrecida es similar a la planteada en este proyecto, la parte del sistema que opera en el terminal móvil difiere notablemente. La solución del proyecto seek-for-android requiere la instalación de un nuevo kernel y una ROM (imagen del sistema operativo), lo que hace al móvil poder operar como lector PC/SC. La complicación que supone adaptar el teléfono supera el conocimiento medio de la mayoría de los usuarios de telefonía móvil e incluso aplicar estos cambios implica la pérdida de la garantía del dispositivo, ya que la mayoría de compañías la invalidan si detectan que se ha obtenido acceso de superusuario en los teléfonos o tablets, lo cual es necesario para cambiar el sistema operativo.

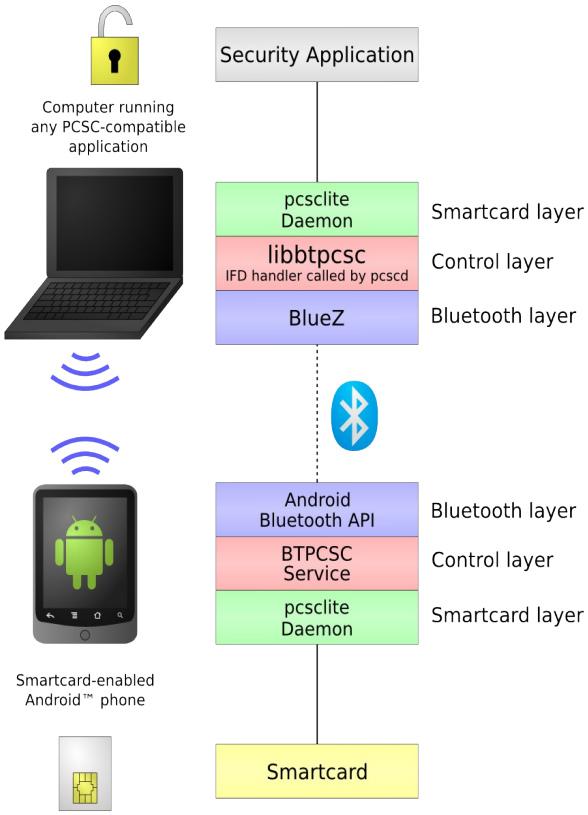

Figura 5.2: Arquitectura de BTPCSC propuesta por seek-for-android

El presente proyecto plantea una solución diferente, que no supone operaciones que puedan perjudicar el funcionamiento del teléfono ni requieren conocimientos técnicos por parte del usuario. Inicialmente se quiso realizar como servicio, es decir, un programa que se ejecuta en segundo plano sin intervención directa del usuario ni necesidad de una GUI. Sin embargo esto no fue posible debido a que Android no permite la utilización de la tecnología NFC desde un servicio [42]. Por tanto, se ha elegido desarrollar el aplicativo móvil como una aplicación convencional, que el usuario debe ejecutar para hacer funcionar su terminal como lector y recibe notificaciones visuales indicándole el estado de la comunicación.

5.3. Desarrollo de la aplicación

5.3.1. Servidor Bluetooth

El primer paso para hacer funcionar el sistema completo de lectura de tarjetas ha sido la implementación de un servidor WPCSC Bluetooth en el teléfono móvil. Para ello, de la misma manera que se hizo en el capítulo 4, se ha desarrollado un dispositivo de lectura virtual. Esto responde al fin de probar desde un primer momento el funcionamiento del protocolo y ver que la comunicación entre PC y terminal móvil es sólida. Para hacer esto, se ha escrito en el código del servidor cada una de las posibles respuestas a las distintas tramas de WPCSC. Para probar cómo funcionaba el protocolo en caso de una desconexión repentina de la tarjeta desde un primer momento, lo que se hizo fue un botón en el programa que cambiaba el valor de un booleano que indicaba la presencia o no de la tarjeta.

Al recibir la conexión entrante, lo primero que hace el servidor es enviar una confirmación, una trama W_PCSC_ACK_CONNECTION, de esta forma el PC sabe que se ha conectado al servidor correcto. Tras esto, cuando el terminal recibe una trama WPCSC, primero tiene que leer la cabecera y analizar qué hacer con el mensaje. Algunos de los comandos tienen una respuesta inmediata, conocida por la propia aplicación y que no necesita de comunicación adicional con la tarjeta (en este caso virtual), como por ejemplo el caso de saber si hay una tarjeta conectada (comando W_PCSC_CMD_CARD_PRESENCE). En este caso simplemente hay que llamar a la función que comprueba la presencia de la tarjeta y devolver un paquete con el mismo byte de cabecera más un booleano como información, tal como se especificó en la sección 4.3.1. Sin embargo las operaciones de lectura, autenticación y escritura sobre la tarjeta tienen una operativa más compleja. Estos comandos son encapsulados dentro de una trama W_PCSC_CMD_APDU. Tras diseccionar el mensaje el APDU se analiza para saber qué comando debe ejecutarse sobre la tarjeta virtual.

Android no dispone de un mecanismo que notifique a una aplicación de que una tarjeta sin contactos se ha desconectado, por lo que es necesario desarrollar dicha funcionalidad. Para ello se implementado un hilo que periódicamente comprueba la presencia de una tarjeta. Buscando mantener la coherencia en la operativa con otras librerías de Android, la notificación de conexión y desconexión se realiza mediante mensajes de Broadcast encapsulados en un Intent.

La operativa de su aplicación en su modo de operación natural sigue el funcionamiento que se describe en la figura 5.3. Al arrancar comprueba si el adaptador Bluetooth del dispositivo está encendido, activándolo en caso contrario previa autorización del usuario. Una vez activado, inicia un hilo encargado de atender nuevas conexiones, que se mantendrá activo hasta el cierre de la aplicación. Este hilo a su vez crea uno nuevo para cada conexión, de forma que no se interrumpa su funcionamiento. El hilo principal es el encargado de gestionar todas las interacciones del usuario con la GUI. No obstante, dado que el terminal móvil empleado solo puede gestionar una tarjeta, se ha deshabilitado la posibilidad de realizar más conexiones por parte de otros equipos para evitar funcionamientos inesperados o accesos incorrectos a la tarjeta.

Para facilitar la depuración del programa se ha hecho que toda la información obtenida de la tarjeta virtual sea estática, es decir, está almacenada en el código fuente y es siempre la misma. En la figura 5.4 se ilustra la lectura de un bloque de información.

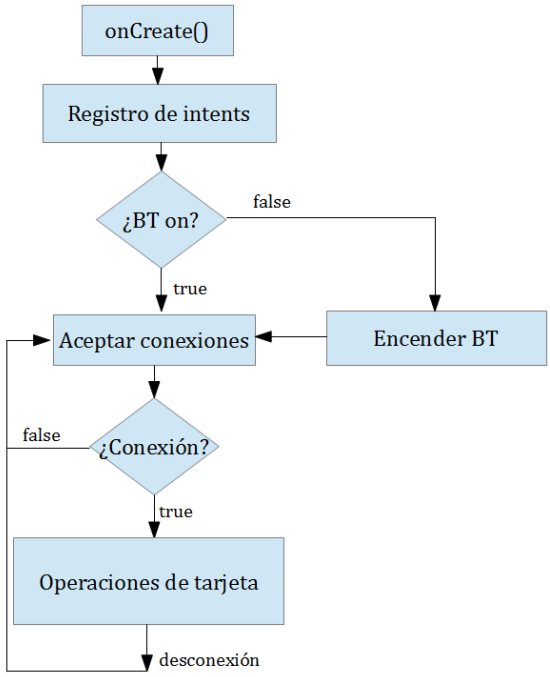

Figura 5.3: Funcionamiento de la aplicación

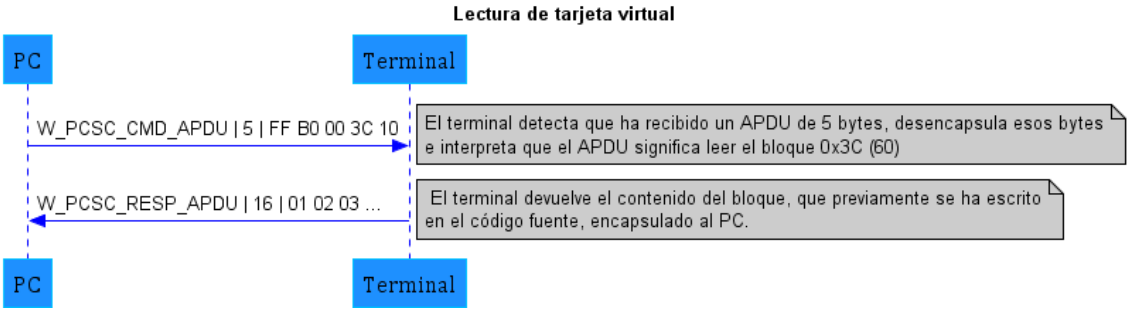

Figura 5.4: Lectura de la tarjeta virtual

5.3.2. Lectura de tarjetas

Con el intercambio de primitivas del protocolo WPCSC funcionando estable y correctamente, El siguiente paso es hacer que la información devuelta al PC no sea estática, sino que se obtenga de una tarjeta real. Se ha elegido trabajar con tarjetas MIFARE Classic, y se ha pensado la aplicación de manera que sólo dé soporte para la lectura de una tarjeta cada vez. Dado que la comunicación directa mediante APDUs con la tarjeta MIFARE no es posible, es necesario emplear las funciones del propio API de Android para llevar a cabo dichas operaciones. Dichas funciones son:

- **authenticateSectorWithKeyA(int sectorIndex, byte[] key)**: autentica el sector indicado por *sectorIndex* usando key como clave A.

- **authenticateSectorWithKeyB(int sectorIndex, byte[] key):** : autentica el sector indicado por *sectorIndex* usando *key* como clave B.
- **readBlock(int blockIndex)**: devuelve un *array* de bytes con la información del bloque indicado por *blockIndex*.
- **writeBlock(int blockIndex, byte[] data)**: escribe el contenido de *data* en el bloque que indica *blockIndex.*

Con la tarjeta real, tras desencapsular un APDU el programa interpreta de qué tipo es y en función de ello emplea una de las citadas funciones de la API para comunicar con la tarjeta. La respuesta que se recibe de la tarjeta es tratada, y tanto en caso de respuesta correcta como de error en la comunicación con la tarjeta, esta se encapsula en una trama W_PCSC_RESP_APDU remitida de vuelta al PC a través del canal Bluetooth.

La figura 5.5 muestra el nuevo diagrama de lectura de un bloque de la tarjeta. El APDU que aparece en la imagen, que en este caso solicita la lectura de un bloque, se manda a la tarjeta mediante la función *readBlock*, y la respuesta, que es el contenido del bloque (para este ejemplo se ha supuesto que se realiza la lectura con éxito) se devuelve encapsulada al PC.

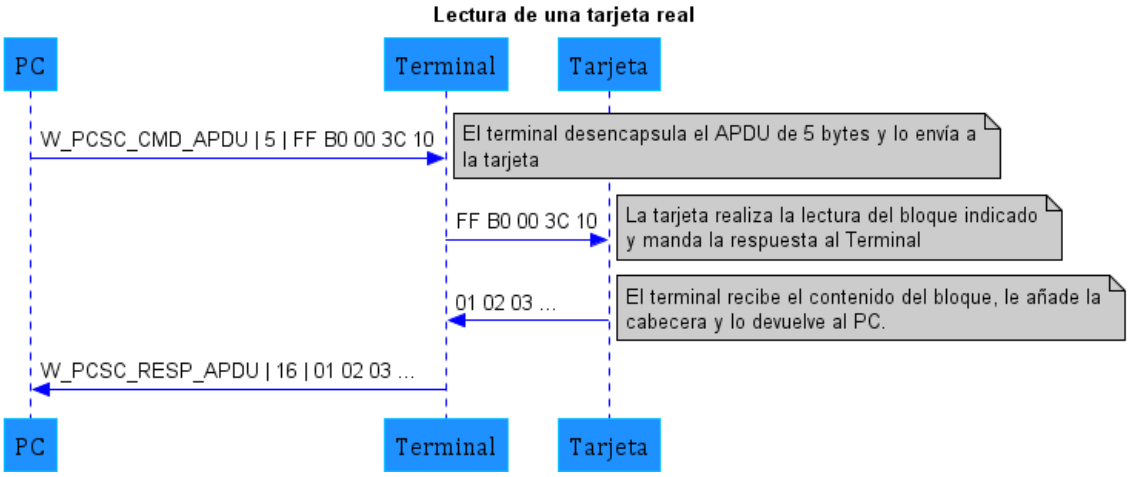

Figura 5.5: Lectura de la tarjeta real

5.3.3. Integración con la interfaz de usuario

Descartada la idea original de hacer funcionar la aplicación como servicio, se ha dotado de una interfaz de usuario (figura 5.6) que muestra el estado de conectividad del lector. Así es posible saber de manera inmediata si el lector está conectado a un PC y si ha detectado la aproximación de una tarjeta. Se ha elegido notificar cada conexión o desconexión de manera visual, de forma que tal como se aprecia en la figura 5.6, se puede ver por separado el estado de la conexión Bluetooth y la presencia de la tarjeta. La figura 5.6 a) ilustra el caso en que no existe conexión al PC ni hay una tarjeta cerca del lector. En la figura 5.6 b) se muestra el caso en que se ha establecido la conexión entre PC y terminal pero aún no se ha acercado ninguna tarjeta, el dibujo superior cambia para mostrar un PC conectado a un teléfono mediante Bluetooth. De la misma manera, la figura 5.6 c) el terminal tiene una tarjeta cerca, y el dibujo inferior cambia para notificar al usuario de que la tarjeta se ha detectado. Por último, en la figura 5.6 d) el sistema está completamente conectado y listo para comenzar a operar sobre la tarjeta. Las figuras c) y d) muestran

casos en que las tarjetas conectadas implementan, respectivamente, MifareClassic e IsoDep + MifareClassic. Como en ambos casos la tarjeta es compatible con el sistema diseñado, además se muestra una notificación de tipo *Toast*, que se superpone en la pantalla, indicando que la tarjeta es compatible. En el caso de aproximar una tarjeta no compatible, la notificación mostraría un mensaje informando al usuario, y el dibujo de la tarjeta no indicaría el tipo de tecnología.

Al desconectarse el PC del terminal, ya sea por un error de comunicación o por la interacción directa del usuario, la aplicación vuelve a aceptar conexiones del cliente Bluetooth, no es necesario reiniciarla ni interaccionar con ella. Si se activa la opción que previene la suspensión del dispositivo mientas éste tenga corriente, puede actuar como lector inalámbrico de manera indefinida.

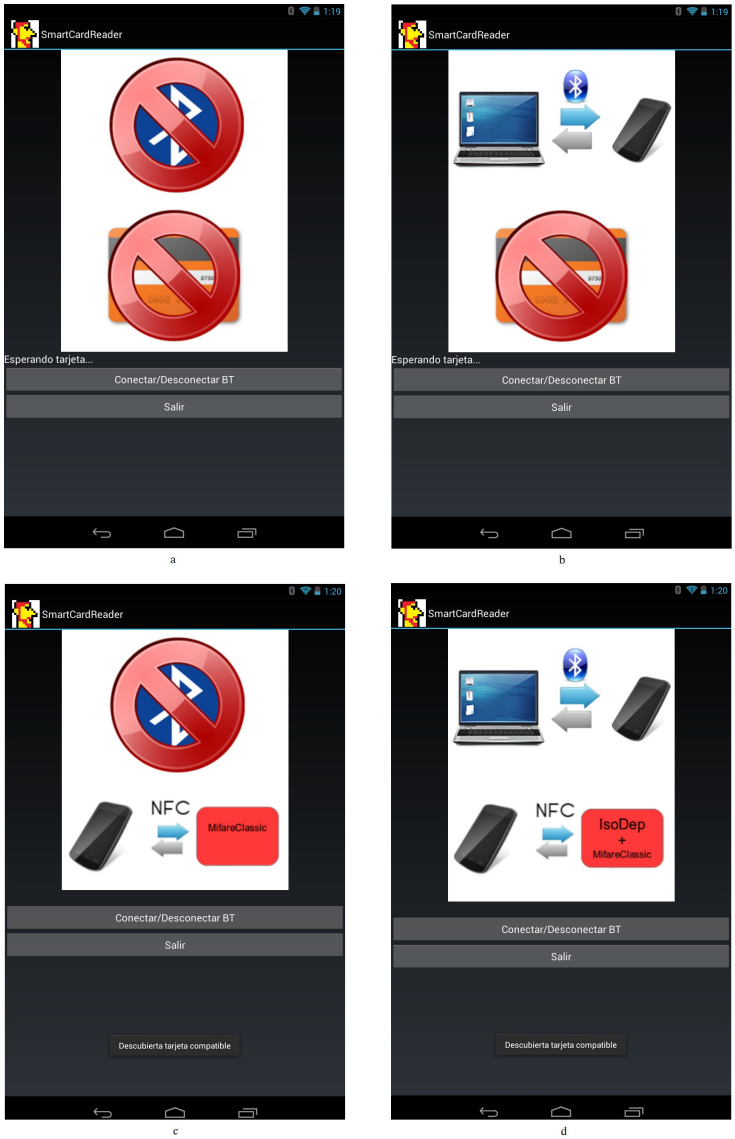

Figura 5.6: a) Todo desconectado, b) PC conectada, c) Tarjeta conectada, d) PC y tarjeta conectados

6.Conclusiones

Es indudable que la tecnología NFC está cada vez más presente en todos los ámbitos, su sencillez de uso y la apuesta de los fabricantes por ella están provocando una rápida expansión. Por ello, no está lejano el día en que el uso cotidiano de estos dispositivos llegue a los hogares, y resulta conveniente equipar a los usuarios con una herramienta que les permita hacer uso de sus tarjetas. Este es uno de los propósitos que se han alcanzado con este trabajo, ya que el sistema puede utilizarse por cualquier persona que disponga de un PC y un teléfono con NFC con un coste adicional nulo.

El desarrollo del lector inalámbrico realizado se ha evaluado comprobando que su modo de operación es similar al de lectores disponibles en el mercado, como ACR 122U. Por tanto, se puede afirmar que se ha logrado un lector plenamente funcional, que puede sustituir a los modelos que se pueden encontrar en las tiendas, con el ahorro que esto supone.

El protocolo diseñado se ajusta a las necesidades mínimas para poder entablar una comunicación con la tarjeta inteligente. Incluye no sólo el manejo de la operativa en casos libres de fallos, sino también en el caso en que se puedan dar problemas en la comunicación, como por ejemplo la desconexión de la tarjeta, la desconexión del canal inalámbrico, etc. Además no se limita al uso en este caso concreto, sino que podría implementarse en cualquier terminal que tuviese funcionalidades NFC y conectividad por Bluetooth.

Además de adquirir conocimientos sobre tarjeta inteligente y su ecosistema de aplicaciones, se han desarrollado las capacidades de programación en diferentes lenguajes, (Java, C y el entorno Android), así como la integración de las soluciones desarrolladas en estos entornos. Se han estudiado campos que no contempla el itinerario de la titulación de Ingeniero de Telecomunicación y que resultan de gran interés, como por ejemplo el funcionamiento de una arquitectura PC/SC, y además se ha modificado para ampliar su alcance. Se ha diseñado e implementado un protocolo de comunicaciones, lo que proporciona una visión más amplia que el estudio de uno ya existente, ya que obliga a razonar la existencia de cada trama y a entender en profundidad los mecanismos de control de errores y el por qué de la existencia de cada uno de los campos de cada trama.

No obstante, aunque la solución desarrollada muestra un nuevo tipo de aplicación de los teléfonos móviles para facilitar el uso de la tarjeta inteligente, no está libre de mejora. Así, en un principio, dar mayor robustez y estabilidad a la solución se antoja como primer paso.

Si bien el lector se ha desarrollado para realizar operaciones con tarjetas MIFARE, perfectamente se puede modificar el código de la aplicación para soportar otros tipos de tecnologías. Además de cambiar el tipo de tarjetas que la aplicación puede utilizar, otra opción es implementar todos simultáneamente, y que sea el propio terminal móvil el que reconozca el tipo de APDU que se envía y elija la tecnología correcta en consecuencia, devolviendo el error pertinente si el tipo de tarjeta no corresponde con el deseado. Así se abre la puerta a un sistema de lectura universal, que permita a los usuarios utilizar cualquier tipo de tecnología NFC sin gastar dinero en lectores específicos.

Otra opción con vistas al futuro es llevar el protocolo WPCSC a otros sistemas operativos móviles que soporten NFC, como Blackberry o Windows Phone, entre otros.

Dado que el protocolo no es específico del sistema operativo, se podría emplear de manera muy amplia, creando aplicaciones para cada terminal con las APIs que proporcione su fabricante. Otra opción viable sería el diseño de lectores inalámbricos, pero en tal caso no existe la ventaja del ahorro que supone usar un teléfono o tablet, ya que habría que adquirir el lector. De igual forma que WPCSC puede emplearse en cualquier sistema operativo móvil, también se puede crear un controlador para los sistemas operativos de PC que no sean distribuciones de GNU/Linux. Y dada la expansión de los sistemas Windows y Mac, esto posibilitaría el uso a muchos usuarios. Si se llegase a implementar, podría incluso proponerse como estándar para el uso de lectores inalámbricos en el entorno doméstico. Además, la aplicación realizada para el manejo de las tarjetas, aunque sencilla, tiene la ventaja de ser portable, ya que al estar hecha en Java puede ejecutarse en cualquier PC que disponga de un entorno de ejecución para este lenguaje de programación. Asimismo, se puede tomar esta aplicación como punto de partida para hacer otras más complejas, que soporten otros tipos de operaciones con las tarjetas.

Para el caso que nos ocupa no se ha implementado ningún mecanismo de seguridad para la comunicación inalámbrica. Como la comunicación es abierta se podría intervenir, aunque dado que el Bluetooth no tiene un alcance muy grande (en torno a 10 m en la mayoría de los casos), es una situación improbable. No obstante, se puede considerar el desarrollo de un protocolo de seguridad para los casos en que se quiera partir de este trabajo para la implementación de algún tipo de sistema se seguridad o identificación.

En cuanto a las posibles aplicaciones, la más obvia es usar el sistema como lector de tarjetas inalámbrico, lo que supone el ahorro de hardware adicional si el usuario dispone de un teléfono Android con NFC. En otros contextos podría usarse para identificación, por ejemplo en un almacén con productos etiquetados con *tags* NFC en que un empleado acerque un terminal y consulte una base de datos en un PC, o un guardia de algún edificio que requiera la tarjeta de identificación a un visitante.

En definitiva, este trabajo supone el comienzo de una iniciativa que busca facilitar la vida al usuario, que supone un ahorro económico y además un valor añadido a los teléfonos móviles más modernos, que adquieren así una nueva utilidad.

Acrónimos

ADB	Android Debug Bridge
ADT	Android Development Tools
APDU	Application Protocol Data Unit
API	Application Programming Interface
ATR	Answer to Reset
CCID	Chip/Smart Card Interface Devices
EEPROM	Electrically Erasable Programmable Read Only Memory
DNIe	Documento Nacional de Identidad Electrónico
GNU	GNU is Not Unix
GSM	Global System for Mobile Communications
GUI	Graphic User Interface
I/O	Input / Output
IDE	Integrated Development Environment
ISO	International Organization for Standardization
LLCP	Logical Link Control Protocol
MAC	Media Access Control
M.U.S.C.L.E.	Movement for the Use of Smart Cards in a Linux Environment
NDEF	NFC Data Exchange Format
NFC:	Near Field Communication
NFCIP	Near Field Communication Interface and Protocol
RFID	Radio Frequency IDentification
RAM	Random Access Memory
ROM	Read Only Memory
RTD	Record Type Definition

OSF	Open Software Foundation
OSI	Open System Interconnection
OTA	Over The Air
PC	Personal Computer
PCD	Proximity Coupling Device
PC/SC	Personal Computer / Smart Card
PIN	Personal Identification Number
SD	Secure Digital
SDK	Software Development Kit
SE	Secure Element
SIM:	Subscriber Identity Module
SMS	Short Message Service
SMSC	Short Message Service Centre
SWP	Single Wire Protocol
TPDU	Transmission Protocol Data Unit
TSM	Trusted Service Manager
UICC	Universal Integrated Circuit Card
UUID	Universal Unique Identifier
USB	Universal Serial Bus
VCD	Vicinity Coupling Device
VPN	Virtual Private Network
WPCSC	Wireless Personal Computer Smart Card

Anexo I: Instalación

Instalar el sistema completo resulta sencillo, incluso desde el código fuente. Los pasos a seguir son:

1. **Editar y guardar el archivo de configuración**. Se copia el archivo *libwpcsc* a la carpeta */etc/reader.conf.d* y se cambia el campo LIBPATH, escribiendo la ruta completa de la librería que se va a generar, que estará ubicada en la misma carpeta que el código fuente, para este ejemplo: */home/user/smartcard/libgen_ifd.dylib*. Se requiere ser superusuario. Para hacerlo, lo más sencillo es abrir la terminal y escribir:

   usuario@equipo:~$ sudo nautilus

   Se introduce la contraseña y se abrirá el navegador de archivos como superusuario. Se entra en la carpeta antes citada y se pega el archivo de configuración, que no tiene extensión, y cuyo contenido es el siguiente:

   FRIENDLYNAME "Smartcard Reader"
   DEVICENAME /dev/null
   LIBPATH /home/user/smartcard/libgen_ifd.dylib
   CHANNELID 1

2. **Compilar el controlador**. Desde una terminal se teclea la ubicación del código y una vez en la carpeta correcta se introduce el comando *make*.

   usuario@equipo:~$ cd /home/user/smartcard/

   usuario@equipo:~$ make

   Con esto el controlador está listo para usarse.

3. **Instalar el aplicativo móvil.** Se copia el archivo *.apk* al terminal, y desde el navegador de archivos se selecciona y se instala.

   Es necesario asegurarse de que en los ajustes de Android se permite la instalación de programas de fuentes desconocidas.

4. **Emparejar el PC con el terminal móvil.** Éste es el procedimiento habitual que se requiere para emplear cualquier hardware que se conecte al PC mediante Bluetooth, una vez emparejados ambos dispositivos el lector inalámbrico está listo para su utilización.

5. **Instalación del aplicativo para PC**. El programa para la lectura de tarjetas es portable, no requiere instalación, simplemente hay que copiar el archivo *.jar* a la ubicación desde la que se quiera ejecutar y abrirlo usando el entorno de ejecución de java.

Completados los pasos de instalación, no resulta necesario volver a emparejar los dispositivos a no ser que se elimine uno de ellos manualmente de la memoria el otro.

Bibliografía

[1] Instituto nacional de estadística, Población total, 2012, http://www.ine.es/

[2] Ministerio de Industria, Energía y Turismo, Evolución del número de clientes de telefonía móvil en España, 2011, http://www.ontsi.red.es/ontsi/es/indicador/evoluci%C3%B3n-del-n%C3%BAmero-de-clientes-de-telefon%C3%ADa-m%C3%B3vil-en-espa%C3%B1a

[3] Ministerio del Interior, Así es el DNI Electrónico, 2012, http://www.dnielectronico.es/Asi_es_el_dni_electronico/index.html

[4] ISO, Identification cards -- Physical characteristics, 2003

[5] ETSI, Smart Cards;UICC-Terminal interface;Physical and logical characteristics, 2010

[6] CardWerk Smarter Card Solutions, ISO 7816, 2013, http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx

[7] PC/SC Workgroup, PC/SC Workgroup Goals, 2012, http://www.pcscworkgroup.com/index.php?c

[8] Ludovic Rousseau, David Corcoran, Movement for the Use of Smart Cards in a Linux Environment, , http://www.musclecard.com/

[9] NXP Semiconductors Austria GmbH Styria, About MIFARE™, 2013, http://www.mifare.net/overview/

[10] NXP Semiconductors, Products by funcion, 2013, http://www.nxp.com/

[11] IBM, Triple DES Encryption, 1998, http://publib.boulder.ibm.com/infocenter/zos/v1r11/index.jsp?topic=/com.ibm.zos.r11.csfb400/tdes1.htm

[12] National Institute of Standards and Technology, Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[13] ISO, Information technology -- Security techniques -- Entity authentication -- Part 2: Mechanisms using symmetric encipherment algorithms, 2008

[14] Radboud University Nijmegen, Faculty if Science, Security Flaw in Mifare Classic, 2008, http://www.ru.nl/ds/@741694/security_flaw_in/

[15] David Oswald, Christof Paar, Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World, 2011, http://www.iacr.org/workshops/ches/ches2011/presentations/Session%205/CHES2011_Session5_1.pdf

[16] Intrepidus Group, NFC subway hack, 2012, http://intrepidusgroup.com/insight/2012/09/ultrareset-bypassing-nfc-access-control-with-your-smartphone/

[17] Smart Card Zone, Smart Cards Shop, 2013, http://www.smartcardzone.com/

[18] Blackboard Inc., Blackboard Sees Strong Momentum For New Contactless Campus Card Solution, 2011, http://www.blackboard.com/About-Bb/News-Center/Press-Releases/Archive.aspx?releaseid=1525424

[19] ISO, ISO/IEC 15408-1:2005, 2009, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40612

[20] Cetecom, SESAM-Vitale, 2013, http://www.cetecom.com/eu/zh/testing/smart-card-technologies/electronic-health-cards/sesam-vitale.html

[21] ISO, Information Technology - Telecommunications and information exchange between systems - Near Field Communicacions - Interface and Protocol (NFCIP-1) , 2004

[22] ECMA INTERNATIONAL, Standard ECMA-340, Near Field Communicaction Interface and Protocol (NFCIP-1), 2004

[23] ETSI, Near Field Communication Interface and Protocol (NFCIP-1), 2004

[24] ISO/IEC 21481, Information Technology Telecommunications and information exchange between systems Near Field Communication Interface and Protocol - 2 (NFCIP-2), 2005

[25] ECMA INTERNATIONAL, Near Field Communication Interface and Protocol -2 (NFCIP-2), 2003

[26] ETSI TS 102 312, Electromagnetic compatibility and radio spectrum matters (ERM); Normalized Site Attenuation (NSA) and validation of a fully lined anecoic chamber up to 40GHz, 2004

[27] ISO, ISO/IEC 15693-1:2010Identification cards -- Contactless integrated circuit cards -- Vicinity cards, 2010

[28] NFC Forum, The NFC Forum, 2013, http://www.nfc-forum.org/aboutus/

[29] NFC Forum, Specification Status, 2013

[30] ETSI, Smart Cards;Test specification for the Single Wire Protocol (SWP)interface;Part 1: Terminal features(Release 7), 2010

[31] GSM Association, Requirements for Single Wire Protocol NFC Handsets Version 4.0 March 2011 Pay-Buy-Mobile Initiative , 2011

[32] Stat Owl, Operating System Version Usage. Market Share of Operating System Versions (OS analysis), 2012, http://statowl.com/operating_system_market_share_by_os_version.php?1=1&timeframe=last_6&interval=month&chart_id=4&fltr_br=&fltr_os=&fltr_se=&fltr_cn=&limit%5B%5D=linux]

[33] Wikimedia, Wikimedia Traffic Analysis Report - Operating Systems, 2012, http://stats.wikimedia.org/wikimedia/squids/SquidReportOperatingSystems.htm

[34] Advanced Card Systems Ltd., ACR 122U NFC Reader Application Programming Interface,

[35] , BlueZ, , http://www.bluez.org/

[36] Bloomberg Businessweek, Google Buys Android for Its Mobile Arsenal, 2005,

[37] Infosite and comparisons, T-Mobile G1, 2012,
http://www.gsmarena.com/t_mobile_g1-2533.php

[38] Eclipse Foundation, The Eclipse Foundation open source community website, 2012,
http://www.eclipse.org/org/

[39] Google Inc., Android SDK | Android Developers, 2012,
http://developer.android.com/sdk/index.html

[40] Google Inc., Intent, 2012,
http://developer.android.com/reference/android/content/Intent.html

[41] Seek for Android Group, Secure Element Evaluation Kit for the Android platform, 2012, http://code.google.com/p/seek-for-android/

[42] Stack Overflow Forum, Can already started Android Service receive NFC tag in the form of Intent?, 2012, http://stackoverflow.com/questions/9804819/can-already-started-android-service-receive-nfc-tag-in-the-form-of-intent