

Universidad de Buenos Aires
Facultades de Ciencias Económicas,
Ciencias Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática



Trabajo Final

**Seguridad de servicios Host Card Emulation (HCE) de los
dispositivos Android**

Autora: Ing. Edith Rivero
Tutor: Mg. Juan Devincenzi

Año 2016

Cohorte 2014

Declaración Jurada

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales de Maestría vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

FIRMADO

Edith Pamela Rivero Tupac

Nro. Documento 95343972

DNI (Perú): 70303536

Resumen

El uso de los dispositivos móviles, en especial los de Sistema Operativo Android, tiene un crecimiento notable sobre todo por su accesibilidad para la creación de aplicaciones. Estos dispositivos son incorporados con varios sensores para mejorar la interacción del usuario, por ejemplo la detección de movimiento, proximidad, intensidad luminosa y otras propiedades del ambiente. Dentro de este conjunto de sensores, NFC (Near Field Communication) [1] proporciona singulares características de comunicación a una distancia máxima de 5 centímetros. Los sensores NFC basan su comunicación en el sistema de identificación por radiofrecuencia, conocido como RFID (Radio Frequency IDentification) [2]. Es así que un dispositivo Android con tecnología NFC puede comunicarse con otros compatibles que estén próximos entre sí. Esto es mayormente utilizado en los sistemas de pago, denominados *contactless payment*. En este entorno por ejemplo, un móvil Android resguarda la identificación del pago a realizar y lo comunica mediante NFC a un punto de venta POS (Point of Sale) para hacer efectiva la transacción. El resguardo de la información es realizada desde dos enfoques: (1) dentro de un elemento seguro, que es un entorno a prueba de falsificación e insertado como micro controlador dentro de Android, y (2) mediante servicios de Host Card Emulation (HCE) [3] que de forma similar protegen la aplicación y sus credenciales con la diferencia que no requiere elementos seguros físicos e implementa servicios en la nube. Un servicio HCE puede ser vulnerable a código malicioso y mayor es el riesgo debido a que Android se ha convertido en un punto de atención para el desarrollo de malware en general. Se considera entonces necesario desplegar una infraestructura segura para este tipo de servicios HCE desde el inicio de la interacción con el sensor de proximidad NFC hasta finalizar la transacción. La presente investigación abarcará la seguridad de los dispositivos Android que permite operar con servicios HCE considerando los riesgos y vulnerabilidades que puedan alterar las aplicaciones y evaluar un esquema de seguro en un determinado contexto de utilización.

Palabras clave: Android, NFC, HCE, elementos seguros, accesos, *contactless payment*, cloud, CoSE, tokenization.

Índice

Introducción	3
CAPÍTULO I: Dispositivos móviles Android y la interfaz NFC	4
1.1. Descripción del SO Android	4
1.2. Sensores incluidos	4
1.3. Near Field Communication	5
1.3.1. Descripción	5
1.3.2. Estándares de comunicación.....	5
1.3.3. NFC en dispositivos móviles.....	6
1.3.4. Modos de operación	7
1.3.4.1. Lectura/escritura de etiquetas	7
1.3.4.2. Comunicación peer-to-peer	8
1.3.4.3. Emulación de tarjeta - Card Emulation	9
CAPÍTULO II: Implementación en modo emulación de tarjeta.....	10
2.1. Basado en Secure Elements (Hardware)	10
2.1.1. UICC - Universal Integrated Circuit Card	12
2.1.2. SE-Embedded	14
2.1.3. SMC - Secure Memory Card.....	15
2.1.4. Transacciones de pago con SE	15
2.1.5. Desarrollo de aplicaciones Android-SE	18
2.2. Basado en Servicios (Software)	19
2.2.1. HCE - Hosted based Card Emulation	20
CAPÍTULO III: HCE en Android	21
3.1. Requisitos en las transacciones de pago	21
3.2. Consideraciones de operación HCE.....	23
3.3. Coexistencia HCE y SE	24

3.4. Ejecución de Servicios HCE	26
3.5. Almacenamiento de credenciales.....	27
3.5.1. Utilización de Tokens - Tokenization	27
3.5.2. Cloud of Secure Elements	31
CAPÍTULO IV: Implementación de un caso de uso	34
4.1. Descripción de los componentes.....	35
4.2. Interacción detallada de los componentes	37
4.3. Seguridad en aplicaciones de pago HCE	48
5. Conclusiones	51
6. Bibliografía y Referencias	53
7. Bibliografía general	56

Introducción

La tecnología de pago mediante NFC, conocida en inglés como *contactless payment* está siendo utilizada desde el 2011 en Android mediante Google Wallet [4] sin embargo, el proceso requería un chip (*Secure Element*) incluido en el hardware [5]. Durante el 2014, se enfatiza el soporte de HCE, lo cual ha movido a muchas industrias de pagos a apostar por este servicio [3,6]. La adopción de servicios HCE en los dispositivos Android ocupa especiales cuidados por el malware y vulnerabilidades que este puede presentar. Por ese motivo, en la presente investigación se realizará un análisis de seguridad y los riesgos en el proceso de pago con Android y servicios HCE. Se abordará las características del sensor de proximidad NFC en los dispositivos Android, seguidamente se revisará los requisitos para el funcionamiento de Android como emulador de tarjetas de pago, así como un análisis de infraestructuras en la nube que proponen asegurar estas transacciones. Finalmente, se realizará una implementación del concepto en base a una solución utilizada en entornos reales para mostrar el caso de uso de una transacción segura. La metodología de la investigación consistirá en la descripción de la información relevante que permita plantear apropiadas condiciones de seguridad en el alcance del proyecto.

CAPÍTULO I: Dispositivos móviles Android y la interfaz NFC

1.1. Descripción del SO Android

Android es una plataforma de código abierto basado en Linux. La primera versión de Android (1.0) fue desarrollado en el 2008 y HTC Dream fue el primer equipo con este SO [7]. Las aplicaciones se desarrollan en Java y sobre Android Studio que es el IDE oficial (Integrated Development Environment). Por encima del kernel de Linux se encuentra Dalvik, que es la máquina virtual sobre la cual cada aplicación crea una instancia para ejecutarse.

1.2. Sensores incluidos

Las distintas versiones de Android y sus respectivas APIs permiten la interacción con los sensores que se implementa en cada teléfono móvil generalmente conocido como *smartphone*. Además de las capacidades de detección básicas en un smartphone como el audífono, micrófono y cámara se encuentran:

- Sensores de ubicación (GPS, Antenas WIFI, Bluetooth, identificadores de Celdas)
- Sensores de humedad y temperatura (ambiente y del procesador)
- Sensor de proximidad (infrarrojos)
- Sensores de luz (fotodiodos)
- Sensores de orientación y movimiento (Sistemas de coordenadas del dispositivo)
- Sensor de presión (barómetro)
- Sensores de aceleración y gravedad (movimiento, tiempo)
- Sensores de rotación (giroscopio)
- Sensores de campo magnético

Añadido a estas capacidades sensoriales, se encuentra la identificación de radiofrecuencia (RFID) y su subconjunto: la tecnología NFC, la cual fue inicialmente añadida en la versión Android 2.3 (Gingerbread) API v.9 que permitió la comunicación inalámbrica a distancias pequeñas menores de cinco centímetros.

1.3. Near Field Communication

NFC es una tecnología que deriva de RFID. Ésta fue mayormente utilizada en la industria de ventas para el inventario y prevención de robo mediante la utilización de etiquetas o tags-RFID que básicamente consisten en dos elementos: (a) bobinas para escuchar las señales de radiofrecuencia y (b) pequeños circuitos integrados que almacenan datos y lógica para retornar la información al lector-RFID. La diferencia esencial entre RFID y NFC es la proximidad, que en términos de seguridad está relacionado con la confianza entre los dispositivos NFC. Posee la ventaja de realizar la asociación y descubrimiento (*paring and discovering*) inmediato de dispositivos, evitando recargados mecanismos de seguridad. Esta facilidad ha sido aprovechada por numerosas aplicaciones para compartir archivos y aplicaciones, asociar accesorios, agregar contactos a las redes sociales, registrar pases de abordaje (*booking*), control de acceso e inclusive para mostrar contenido de las memorias portátiles o tarjetas SD sin necesidad de abrirlas a través de un equipo.

1.3.1. Descripción

NFC está diseñado para una frecuencia de corto alcance 13.56 MHz, velocidad hasta 424 kbps y una distancia muy pequeña de 1-4 cm o hasta 10 cm en perfectas condiciones. A diferencia de los tags-RFID que tienen un identificador único de 40 bits y son solamente de lectura; los tags-NFC pueden almacenar 48B y hasta 8KB de datos y adicionalmente pueden ser sobrescritos si el tag no tiene la protección de escritura [8].

1.3.2. Estándares de comunicación

Los estándares en NFC son regulados por varias entidades, entre ellas destaca la ISO/IEC y la organización NFC Forum¹ conformada por fabricantes, desarrolladores e instituciones financieras que promueven la tecnología NFC y su aprehensión.

Se definen principalmente los estándares ISO/IEC-14443 e ISO/IEC-7816. El estándar ISO/1443 define la operación de las tarjetas de proximidad, los tipos de tarjetas, modos de codificación y modulación. Está compuesto de

¹ www.nfc-forum.org

cuatro partes cuya descripción contiene la capa física (14443-1), la interfaz de señal de radiofrecuencia (14443-2), activación y colisiones (14443-3) y el protocolo de transmisión (14443-4).

El estándar ISO/IEC 7816-4 describe la organización y estructura de seguridad para el intercambio de la información por lo tanto se acentúa especialmente para las tarjetas de identificación y pagos. El formato estándar para los datos es NDEF (NFC Data Exchange Format) que contiene el *payload* y metadatos de longitud, tipo e identificador. Cabe resaltar que existen tags que no cumplen estas especificaciones ya que han implementado sus formatos propietarios.

1.3.3. NFC en dispositivos móviles

La incorporación de NFC varía en versiones y modelos, aunque la arquitectura generalmente implementada consiste en un controlador situado en la placa y una antena de transmisión como se presenta en la Figura 1.

La tecnología NFC en Android también satisface las características de bajo consumo de energía y en comparación con otras tecnologías como los códigos QR (Quick Response), NFC no requiere mantener posiciones fijas ni portar una cámara, similarmente en comparación con Bluetooth, no necesita un emparejamiento de los dispositivos para iniciar comunicación. Por otro lado, NFC está limitado a pequeñas transacciones de datos y a distancias cortas que proporcionan un nivel de seguridad ya que complica a los *sniffers* la exploración.



Figura 1: Antenas NFC en distintos modelos

1.3.4. Modos de operación

Los modos de operación de NFC son tres: (1) Lectura y escritura de datos desde el móvil NFC con un tag. (2) Peer-to-peer, que permite intercambiar datos entre dos móviles NFC y (3) Card Emulation, que permite interactuar el móvil con un lector NFC para transmitir datos cuya naturaleza de información es similar a las tarjetas transaccionales. Dentro de este último modo, se ha desplegado diferentes estructuras para el almacenamiento seguro de los datos, entre ellos Host Card Emulation HCE, el cual será desarrollado con más detalle por ser objetivo del presente trabajo.

1.3.4.1. Lectura/escritura de etiquetas

Existen muchos tipos de etiquetas o tags-RFID clasificadas entre activas y pasivas. Las pasivas no tienen energía y su uso es mayormente de lectura cuando son interrogadas por un escáner o lector-RFID. Mientras que

las activas tienen la ventaja de transmitir y recibir ondas a distancias más largas.

La comunicación de un tag pasivo hacia el lector-RFID se logra cuando las ondas de radio generadas por el lector-RFID causan a las bobinas del tag oscilar, lo cual se convierte en energía y por ende puede transmitir información. En este escenario y considerando a Android como un lector-RFID se resalta que no todos los dispositivos son compatibles con todos los tags. En la Figura 2, se observa un listado de combinaciones compatibles de los tags y modelos de dispositivos Android.

Device	Type 1	Type 2 (Mifare Ultralight)	Type 3 (Sony FeliCa)	Type 4 (Mifare DESFire)	Mifare Classic
Samsung Galaxy S	Yes	Yes	Yes	Yes	Yes
Google Nexus S	Yes	Yes	Yes	Yes	Yes
Google Galaxy Nexus	Yes	Yes	Yes	Yes	Yes
Google Nexus 7 version 1	Yes	Yes	Yes	Yes	Yes
Google Nexus 7 version 2	Yes	Yes	Yes	Yes	No
Google Nexus 4 (phone)	Yes	Yes	Yes	Yes	No
Google Nexus 10 (tablet)	Yes	Yes	Yes	Yes	No
Samsung Galaxy S4 (tablet)	Yes	Yes	Yes	Yes	No
Samsung Galaxy SIII	Yes	Yes	Yes	Yes	Yes

Figura 2: Modelos de SO Android compatibles. Imagen adaptada de [9]

Las aplicaciones más destacadas en este modo son escaneo de información en posters, lectura de mensajes, enlaces a perfiles de redes sociales, direccionamiento a URLs, reconocimiento de tarjetas identificadoras de personal, entre otros.

1.3.4.2. Comunicación peer-to-peer

En este modo, la energía en los dispositivos les permite realizar la comunicación activa y por lo tanto intercambiar datos. El formato del mensaje de intercambio es NDEF y requiere que ambos Android estén equipados con NFC.

La aplicación más referenciada es Android Beam cuya idea básica es compartir lo que se muestra en la pantalla. En Android 4.0 (Ice Cream Sandwich) se lograron transferencias de datos con tamaños pequeños como contactos, enlaces para descargar aplicaciones, enlaces para videos online, etc. Posteriormente, en Android 4.1 (Jelly Bean) la aplicación de Android

Beam incorporó el envío de archivos como fotos, videos y otros cuyos tamaños superan los megabytes [10]. El proceso subyacente consiste en hacer el emparejamiento con NFC y continuar la transferencia con Bluetooth. Otra aplicación que tiene un comportamiento similar es S-Beam de Samsung, en este caso difiere con la tecnología de transferencia ya que utiliza Wi-Fi Direct², aunque el emparejamiento lo realiza mediante NFC.

1.3.4.3. Emulación de tarjeta - Card Emulation

En este modo, el dispositivo móvil se comporta como una tarjeta transaccional en cumplimiento con los estándares de NFC que definen las interfaces de comunicación. La característica de este comportamiento hace que la mayor parte de aplicaciones prácticas estén involucradas en proyectos financieros y participen varias entidades como los fabricantes de microcontroladores, fabricantes de teléfonos, operadores de redes móviles, instituciones financieras y otras entidades reguladoras. Ante este variado ecosistema, a la fecha de esta investigación, no se ha estandarizado un determinado esquema de pago. Sin embargo, entidades con alta experiencia en el mercado como Visa y MasterCard implementan requerimientos funcionales específicos y de seguridad dentro de este modo de operación. Se comprende entonces que el elemento de interés es dónde y cómo resguardar la información crítica ya sea para autenticar usuarios y/o efectuar transacciones, evitando su exposición y alteración ante el malware e individuos con intenciones adversas, y mantener un sistema que no altere otros coexistentes.

² <http://www.wi-fi.org/discover-wi-fi/wi-fi-direct>

CAPÍTULO II: Implementación en modo emulación de tarjeta

Los casos de uso de emulación de tarjeta en un dispositivo móvil, además de proporcionar estrictamente niveles de seguridad para el procesamiento y guardado seguro de datos sensibles, también deben abordar requerimientos de transacciones óptimas en velocidad, usabilidad, interoperabilidad y disponibilidad. Para responder a estas necesidades, se conocen dos grandes mecanismos: uno de ellos basado en hardware o elementos seguros (Secure Element) y otro basado en software o servicios. Naturalmente, existen implementaciones híbridas de ambos mecanismos que incluyen servicios cloud, tokens, etc. para robustecer la seguridad y ampliar mercados de interés.

2.1. Basado en Secure Elements (Hardware)

Los Secure Elements (SE) básicamente son micro controladores que pueden venir en distintos formatos y puede albergar múltiples aplicaciones. Éstos presentan medidas contra los ataques sofisticados y como resultado no hay casos demostrados de acceso no autorizado o duplicación de los datos que se albergan. Se considera también un segundo factor de autenticación como "Algo que tienes" añadido a "Algo que sabes" [11].

El esquema de certificación de un SE está establecido de tal manera que varias entidades certificantes realizan una revisión independiente de los ataques y también del cumplimiento estricto de especificaciones funcionales. Por ejemplo, el SE en formato de tarjeta con chip integrado salió a operar tras haber pasado la certificación completa de EMVCo, Visa, MasterCard, American Express, etc.

La ilustración en la Figura 3, muestra los SE que han sido integrados a los dispositivos móviles para habilitar en el esquema de pago **NFC-payments-SE**. A continuación se hace una referencia de desarrollo en el tiempo sobre la integración de SE, que permitirá comprender la operación de NFC en Android.

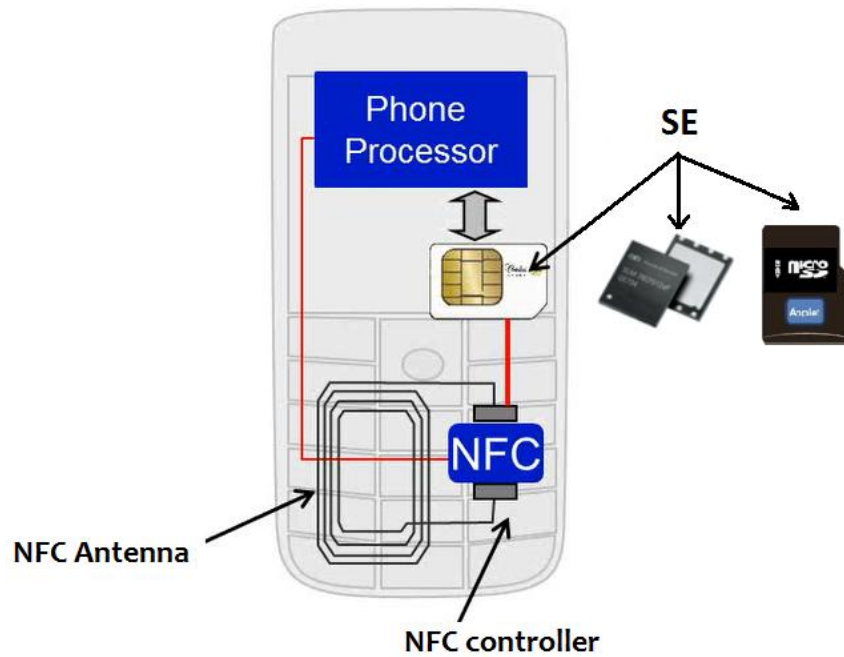


Figura 3: Esquema de comunicación con SE. Imagen adaptada de [3]

La conformación de NFC Forum inició en el 2004 con el objetivo de estandarizar NFC incluyendo distintos organismos de comunicación. La primera atención fue destinada a los esquemas de comunicación de lectura/escritura de tags. Seguidamente en el 2006, el primer dispositivo con NFC fue lanzado por Nokia. Consecuentemente, el modo de transmisión entre dos smartphones tomó mayor atención para la definición de transferencia de datos pequeños (contactos, URLs, e iniciar Bluetooth). Es así que a finales de 2008, GSMA como ente global para la industria móvil terminó por definir los requerimientos de los dispositivos móviles con NFC [12] y finalmente en el 2009 se publicaron los estándares. Un año más tarde, Google y Samsung lanzaron Nexus S, que es considerado el primer Android con NFC. Durante ese periodo de tiempo y en adelante, el interés fue incrementando para utilizar NFC en smartphones y efectuar pagos. Se definieron certificaciones para poder utilizar componentes seguros (por ejemplo UICC/SIM) y resguardar la información sensible de pagos, esto significó la unión de entidades financieras y operadores de redes móviles para estandarizar y diseñar modelos de colaboración para que las aplicaciones de pago vía NFC puedan trabajar en cualquier plataforma [13].

Estas entidades fueron GSMA³, EMVCo, Global Platform⁴ de tal manera que cualquier aplicación de pago certificada pueda trabajar con cualquier plataforma UICC/SIM certificada. En Setiembre de 2011, Google Wallet hace su aparición [4,14] en Nexus-S-4G (Android 2.3.4) en conjunción con Sprint como operador de la red móvil; Citi como la institución financiera, MasterCard como concesionario de pago y FirstData con el gestor de confianza para las transacciones con SEs. A partir de entonces, diversas compañías relacionadas a la industria avocaron sus actividades para asociar cuentas bancarias a la tarjeta SIM, incorporar microcontroladores NFC y lograr certificaciones para su implementación.

Los niveles de protección para las aplicaciones seguras en los SE (apps-SE) estaban implementados con permisos a nivel core del sistema operativo incluyendo firmas a nivel de plataforma [15], lo cual hacía imposible desarrollar y distribuir apps-SE sin que estuvieran firmados por los entes distribuidores. Es así que el esquema cambió a la dependencia de un certificado de confianza (en Android API v.15) que libera la dependencia a nivel de SO-plataforma. Paralelamente, SIMalliance⁵ publicó a mediados del 2011 la disponibilidad de Open Mobile API v1 que permite la utilización de SE (SIM) de un dispositivo móvil proporcionando una capa de transporte hacia la aplicación segura [16]. A la fecha, OMAPI 3.0 es la versión actual, que desde su aparición ha sido adoptada por la industria comercial y de proyectos dedicados a aplicaciones con SE en Android.

Estas aplicaciones se han implementado sobre tres tipos de SE: En el chip del celular, en tarjetas SD y en microchips embebidos en la placa del celular, los cuales son detallados a continuación.

2.1.1. UICC - Universal Integrated Circuit Card

Es un microchip usado en la telefonía celular suministrado por los operadores móviles MNO (Mobile Network Operator), es comúnmente llamado SIM (Subscriber Identity Module) y permite identificar a los usuarios suscriptores con la red móvil respectiva. Las tarjetas SIM cumplen con el

³ <http://www.gsma.com/>

⁴ <http://www.globalplatform.org/specificationcard.asp>

⁵ <http://www.simalliance.org/>

estándar ISO/IEC 7816 y además de albergar información personal y de contactos también incluyen múltiples aplicaciones, las cuales son generalmente conocidas como *applets* ya que emergió su desarrollo basado en la tecnología *Java Card*⁶. La instalación/actualización remota de estos *applets* se realiza a través de OTA (Over The Air), el cual es un método de transferencia de mensajes inalámbrico que consiste en utilizar la red celular para el envío de archivos binarios mediante SMS (Short Message Service). En muchas SIM esta es la única forma de cargar *applets* inclusive desde la personalización inicial [17]. La comunicación hacia la SIM es posible mediante un conjunto de comandos y procedimientos definidos en el estándar SIM ToolKit (STK), el cual es implementado por la MNO para gestionar los *applets* y actualizar sus servicios.

Específicamente la arquitectura de acceso a la SIM se realiza mediante Radio Interface Layer⁷ (RIL), esta capa permite el intercambio de paquetes del sistema operativo y la interfaz de radio, por ejemplo que Android comunique los servicios de telefonía. La librería de comandos RIL prohibía el intercambio de información entre una *applet* en el SIM y una aplicación móvil, por lo tanto existía una dependencia de interacción por las MNO. Los fabricantes de chips integraron una versión de RIL especial bajo reglas de control de acceso que haga factible esta comunicación, de esta manera una aplicación de pago podría implementarse bajo la segmentación de Dominios de Seguridad para cada entidad financiera.

Los casos de aplicaciones de pago utilizando el chip SIM como SE han sido los más implementados tras lograr alianzas entre industrias y cumplimientos de certificación. En la Figura 4, se observa la arquitectura básica del proceso de pago. A nivel del teléfono, se utiliza el protocolo de intercambio Single Wired Protocol (SWP) [18] que permite la comunicación full-duplex (transmisión y recepción al mismo tiempo) con el UICC/SIM para ejecutar el pago con el Point-of-sale (POS). Previamente este chip ha recibido el *applet* y las credenciales mediante OTA, las cuales son gestionadas por la entidad bancaria (Bank card). Un tercer rol corresponde a los Gestores de Confianza

⁶<http://www.oracle.com/technetwork/java/embedded/javacard/overview/default-1969996.html>

⁷<http://www.kandroid.org/online-pdk/guide/telephony.html>

(TSM) el cual habilita la integración independiente de entidades bancarias y está detallado en los siguientes numerales.

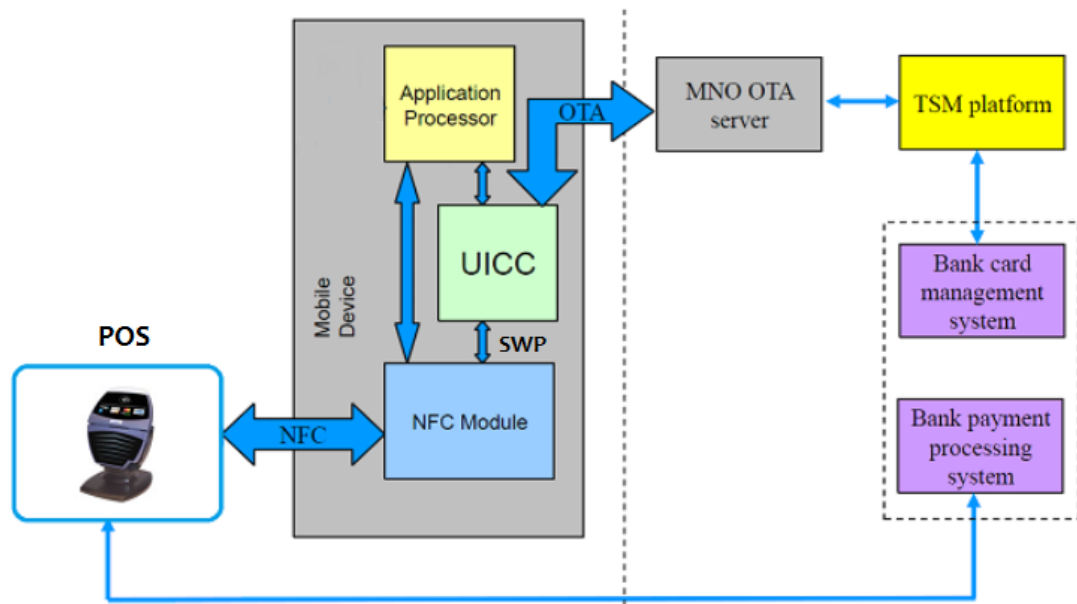


Figura 4. Arquitectura básica de pagos NFC-SIM. Imagen adaptada de [17]

2.1.2. SE-Embedded

El SE es un elemento adicional a la arquitectura del teléfono móvil, mediante un chip incrustado o dentro de un micro controlador en la placa madre. El fabricante que lidera el mercado de su fabricación es NXP, presentando dos tipos de módulos: NFC-transceptor y NFC-controlador cuya diferencia del segundo sobre el primero es la inclusión de un microprocesador con memorias RAM, ROM y EEPROM diseñados para traducir los comandos hacia el módulo NFC provenientes de aplicaciones programadas en lenguajes de alto nivel [18]. El modelo PN65N combina el controlador PN544 y un chip SMARTMX⁸ como SE. Las comunicaciones entre ambos chips se realizan mediante la interfaz NFC-WI (Wired Interface) también conocido como S2C (Signal In / Signal Out) cuyas señales utilizan dos canales dedicados a la transmisión y recepción.

⁸ <http://www.nxp.com/documents/leaflet/75017515.pdf>

2.1.3. SMC - Secure Memory Card

Las tarjetas Smart SD tienen la forma similar a las memorias microSD comunes pero internamente incluyen el soporte para la comunicación NFC aislando un SE, un Dominio de Seguridad, un controlador NFC y una micro antena. La información de pago es cifrada y firmada por la entidad financiera que proporciona el servicio. Su adopción fue atractiva por proporcionar portabilidad e insertarse en cualquier dispositivo que admitiera slots para la memoria microSD, sin embargo problemas técnicos se encontraron durante las implementaciones productivas. Entre ellos, la débil señal de radio debido al tamaño y la ubicación de la antena, interferencias causadas al situarse cerca el metal, dependencia de la ubicación del slot ya que en algunos modelos eran muy internos. Adicionalmente los mecanismos de comunicación segura entre la interfaz de usuario y los slots de la memoria aún no están definidos.

2.1.4. Transacciones de pago con SE

Para los servicios móviles, los parámetros de seguridad de un SE permiten considerarlo como un componente idóneo para las transacciones de pago. Sin embargo, para lograr la seguridad de principio a fin (end-to-end security) es necesario incluir otros componentes que permitan asegurar la confidencialidad en la ejecución de las instrucciones del aplicativo y en la comunicación de los datos procesados. El componente a nivel del dispositivo móvil se refiere a un espacio de procesamiento aislado conocido como TEE (Trusted Execution Environment) y el componente a nivel externo se refiere al intercambio seguro de mensajes contemplado por un tercero de confianza conocido como Trusted Service Managers (TSM) que principalmente realiza el mantenimiento de los esquemas de seguridad y la calidad del servicio al cliente [19].

En los escenarios donde un SE alberga múltiples cuentas por ejemplo: tarjetas de crédito, débito, pases de tránsito, tarjetas para clientes de tiendas, bonos, etc. Los componentes TEE y TSM complementan la seguridad del SE ya que cada tarjeta contiene su propia información de seguridad que debe ser aislada para cada proveedor de servicio. Por otro lado, se asegura la obtención de información y actualizaciones de manera

independiente al método (generalmente OTA) y a través de la red de los distintos operadores móviles.

a) Gestores de confianza - Trusted Service Managers (TSM)

El concepto fue introducido en el 2007 para facilitar la adopción de los servicios NFC, descrito como una entidad independiente cuyo rol es de intermediario entre las instituciones financieras y de telefonía/propietarios de los SE dentro del ecosistema de pagos mediante NFC.

La Figura 5 ilustra la interrelación de los tipos de TSM: (1) SE-TSM que es responsable del acceso y modificación de los datos en el SE y (2) SP-TSM que proporciona la aplicación dentro del SE responsable de efectuar los pagos [20]. Ambas funciones pueden ser manejadas por un ente TSM, cuya neutralidad se considera fundamental para que los distintos dispositivos móviles puedan interactuar con varias cuentas de los proveedores de servicio a través de redes de pago.

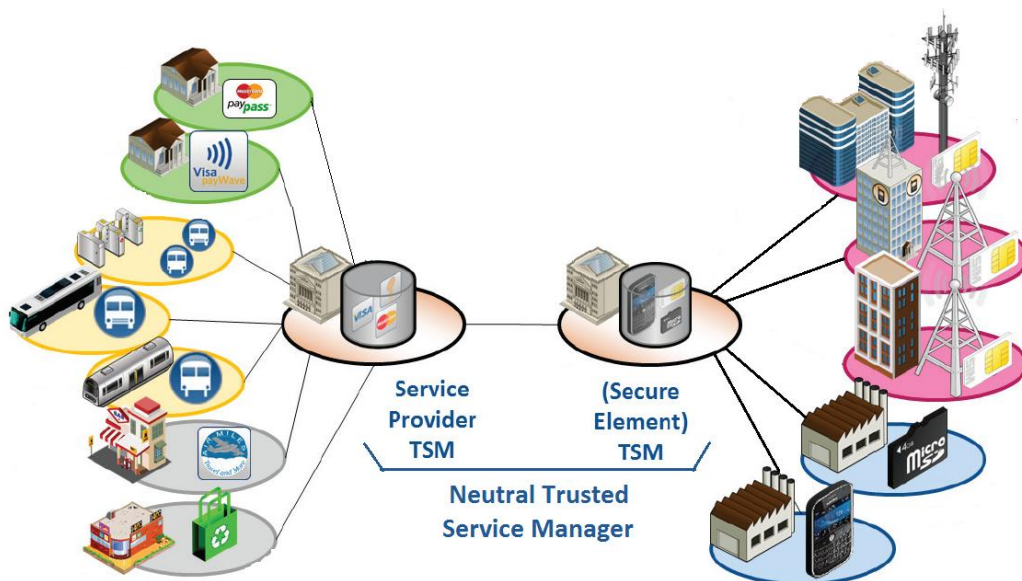


Figura 5: Rol de TSM - Imagen adaptada de [20]

El rol de TSM requiere manejar relaciones contractuales con todas estas entidades incluyendo soporte de servicio al cliente, centros de datos y calidad de servicio. Entre las capacidades clave [19] se encuentran: (a) la descarga de aplicaciones proporcionadas por los proveedores de servicio, considerando que requiere autenticación y enrolamiento previo, (b) la restricción de acceso a la información del SE a otros procesos, (c) asegurar

los protocolos de seguridad inicio-fin del tratamiento de la información validando la identidad del usuario móvil para realizar la transacción.

b) Ambientes de ejecución - Trusted Execution Environment (TEE)

Aunque los datos permanezcan protegidos en el ambiente de un SE, su procesamiento en la arquitectura del sistema operativo del dispositivo denominado también Rich Operative System (RichOS) puede ser vulnerado por el malware existente. Un TEE es un sistema de ejecución aislado que refuerza a los SE (no los reemplaza) proporcionando un mayor nivel de seguridad que el RichOS. Se presenta una interfaz de usuario (UI) creando una canal seguro entre el SE y el usuario final para el cifrado de sus credenciales e información de la tarjeta de pago a ser almacenada en el SE. Las aplicaciones ejecutadas en este entorno aislado se llaman Trusted Applications (TA). Inicialmente estas TA nacieron para otros mercados con necesidades de protección de su contenido (gestión de *copyright* de contenido digital), *BYOD* para el manejo de la información confidencial y otras tecnologías de servicios de almacenamiento seguro, operaciones criptográficas, sincronización segura, etc.

El entorno aislado que ofrece TEE permite la protección ante ataques de software ya que encapsula la toma de controles de la pantalla y el teclado garantizando “What You See Is What You Sign” (esta propiedad permite al usuario tener la certeza de la información a ser procesada, se aplica especialmente para las transacciones de pago).

Global Platform es la entidad que estandariza la interoperabilidad de TEE [21] definiendo los objetivos y los requisitos funcionales de seguridad en ciclo de vida de la aplicación (end-to-end). El perfil de operación de TEE está basado en la certificación EAL2+ (Evaluation Assurance Level⁹) que corresponde a los distintos niveles para evaluar entornos seguros, específicamente el segundo nivel está centralizado en las vulnerabilidades de explotación por software.

En la Figura 6, se presenta la arquitectura TEE. A la izquierda se muestra la separación entre el RichOS y el TEE en el dispositivo. A la derecha se

⁹ Niveles definidos en el estándar ISO/IEC 15408: Common Criteria for Information Technology Security Evaluation

muestra las tres alternativas de implementación del TEE: procesador paralelo, procesamiento dentro del subsistema del chip y en cada componente del procesador.

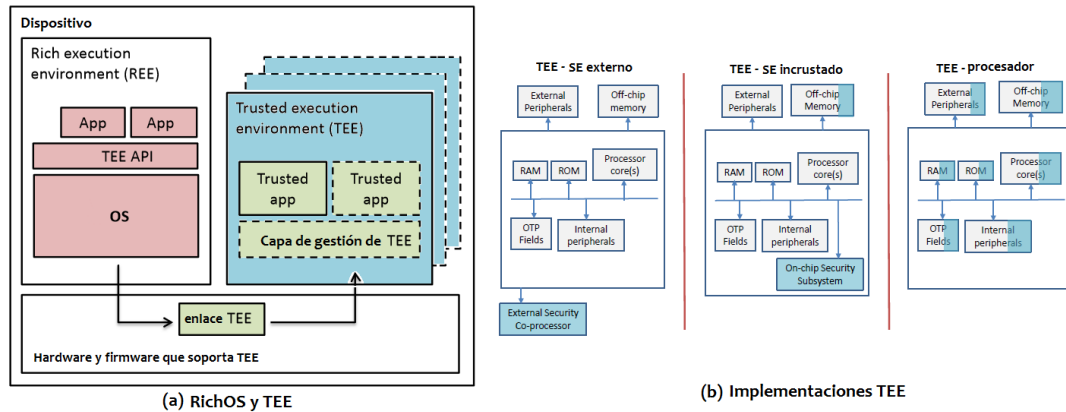


Figura 6: Arquitectura de TEE. Imagen adaptada de [22]

La interfaz de interacción TEE API está desarrollada a nivel propietario con mecanismos de firmas que forman sistemas cerrados basados en credenciales dependientes del fabricante. On Board Credentials (ObC) es una plataforma que promueve la creación flexible de credenciales para permitir a proveedores de servicio implementar instancias de firmas a nivel de hardware TEE [22]. Esto permite aprovechar los entornos de TEE que están implementados en varios dispositivos móviles de propósito general; sin embargo, aún persiste la dependencia de implementar este diseño.

2.1.5. Desarrollo de aplicaciones Android-SE

En la comunidad de código abierto se encuentra el proyecto Seek-for-Android [23] el cual proporciona un conjunto de herramientas de evaluación para elementos seguros en la plataforma de Android. El objetivo del proyecto es que Android se convierta en una plataforma segura para desarrollar aplicaciones compatibles en software y hardware que pueda ser implementado en cada teléfono Android que salga al mercado. La idea básicamente consistió en procesar información con los SE desde Android. La referencia de esta interacción estaba definido en una API conocida con el nombre de "*Secure Application And Trusted Services API (SATSA, JSR 177)*" disponible desde el 2004 para la plataforma Java Micro Edition - J2ME. La utilización no se permitía a nivel de desarrolladores en ninguna de

las plataformas tales como Android, Windows e IOs, que precisamente tomaron mayor foco de atención para el desarrollo.

En caso de Android el proyecto SEEK se inició en el 2010 con el lanzamiento de SmartCard API v1. Esta era una interfaz propietaria que permitía acceder a los SE e inicialmente estaba limitado a teléfonos con propósito de desarrollo ya que era necesario reinstalar (flashear) el sistema o rootear¹⁰ el teléfono.

Por su parte SIMalliance, como ente promotor de implementaciones con SE, desarrolló en el 2011 las especificaciones de Open Mobile API (OMAPI v1), las cuales definen el cómo las aplicaciones móviles pueden acceder a los SE mediante las interfaces o librerías. Esta API además proporciona diagramas UML que orientan su utilización por servicios desde distintos lenguajes de programación y plataformas de dispositivos móviles.

A partir de la versión SmartCard API v.2.2.0 incluyó OMAPI en sus versiones y posteriormente se eliminó la interfaz propietaria para dar soporte únicamente a OMAPI proporcionando el servicio de implementación específico para Android.

A la fecha, la versión disponible de Smartcard API v3.2.1 fue lanzada en Julio de 2014. Incluye la versión OMAPI v2.05 y está basado en Android 4.4.4 (KitKat). Por su parte la versión de OMAPI v3.0 fue lanzada en Noviembre de 2014 que probablemente sea incluida en las siguientes versiones del proyecto SEEK.

Probablemente el caso esperado para los desarrolladores de Android, cuyas aplicaciones están condicionadas al uso de SE, es la integración de una arquitectura segura como el prototipo de SEEK dentro de los nuevos móviles, mientras tanto se avance con las distintas versiones, protocolos y esquemas de servicios conocidos.

2.2. Basado en Servicios (Software)

¹⁰Rootear un teléfono Android significa reemplazar el sistema original para escalar permisos sobre las configuraciones y hardware que son inaccesibles a un usuario normal.

A diferencia del mecanismo basado en SE, se presenta una solución basada en servicios. La motivación de este mecanismo evita el vínculo complejo y costoso que tiene la presencia de un SE en la interacción de las entidades dentro del ecosistema de NFC. Similarmente, se aprovecha la adaptabilidad de los servicios cloud para entornos móviles. La tecnología implementada recibe el nombre de HCE (Hosted based Card Emulation), que será detallado en los siguientes numerales.

2.2.1. HCE - Hosted based Card Emulation

Este mecanismo permite a una aplicación de un dispositivo móvil con NFC emular a una tarjeta y comunicarse con el lector-NFC sin la dependencia de un SE. En este sentido, el SE 'físico' se traslada a un entorno 'virtual' en la nube y se emplean librerías y APIS utilizadas por la aplicación.

En [24] se presenta la historia de HCE. La cual inició en el 2012 por SimplyTapp; y luego fue públicamente soportado por Blackberry. Por otra parte, en Android se añadió parches en la versión CyanogenMod¹¹ y seguidamente la versión Kitkat 4.4 soporta oficialmente HCE.

¹¹Es un firmware basado en Android que ofrece opciones no encontradas en la versión oficial <http://www.cyanogenmod.org/>

CAPÍTULO III: HCE en Android

Desde el soporte de HCE en las versiones oficiales de Android, Google Wallet que es la aplicación más representativa, solamente puede ser usada con dispositivos que soportan HCE ya que mediante éste se realizarán las transacciones de pago conocidas como '*tap and pay*'.

La API de Android que soporta HCE permite a los desarrolladores controlar la interfaz NFC, su aplicación es guiada mediante un repositorio publicado en el sitio oficial de Android [25] y la emulación de tarjetas prosigue los estándares mencionados en el numeral 2.2 de este trabajo.

HCE en Android 4.4 no está completamente aceptado como un esquema de gestión seguro de acuerdo a las especificaciones de Global Platform, como por ejemplo: respuestas de error al comando de selección, envío de comandos obligatorios y soporte de canales lógicos.

Sin embargo, existen industrias dentro de las redes pagos que soportan HCE; por ejemplo Visa y MasterCard habilitan este servicio proporcionando sus lineamientos de implementación, requerimientos para el proceso de aprobación y cumplimiento de estándares.

Para comprender la operación de HCE, primero se hará una referencia a los requisitos del método de pago basado en SE del tipo *smartcard* generalmente conocido como tarjeta con chip integrado. Este método está operativo internacionalmente en base al estándar ISO/IEC 7816 y subyace en la idea de colocar múltiples aplicaciones financieras en un solo chip. Seguidamente se mencionarán las consideraciones de operación de HCE, ya que además de emular el método mencionado, debe cumplir con otros requisitos característicos de un teléfono móvil.

3.1. Requisitos en las transacciones de pago

EMVCo (Europay/MasterCard/Visa) es un esquema estandarizado de pagos que usa la tecnología de tarjetas integradas con un chip. La descripción de la interoperabilidad entre las tarjetas y los dispositivos de procesamiento (ATM, POS, etc.) está publicada en una serie de libros¹² en base a las especificaciones de los estándares ISO/IEC 7816 e ISO/IEC 7810.

¹² <http://www.emvco.com/specifications.aspx?id=21>

El compendio de Arquitectura y Requerimientos Generales describe las características del chip insertado en las tarjetas de crédito y débito. Mediante este chip se autentica al usuario cuando ingresa su clave secreta, permitiendo el control de aprobación de transacciones (incluyendo aquellas sin conexión) a través de algoritmos de cifrado y claves públicas.

Con respecto a la selección de cuenta y servicio, este chip está diseñado para clasificar aplicaciones de acuerdo a una lista de identificadores únicos: AID (Application Identifier) que son asignados por una autoridad de registro definida en la ISO/IEC 7816-5. Cada AID contiene dos partes RID (Registered Application Provider ID) que es un identificador único para cada proveedor y PIX (Proprietary Application Identifier Extension) que permite diferenciar los variados productos ofrecidos por cada proveedor. Por ejemplo:

Aplicación	RID	PIX	AID
Visa Electron	A000000003	2010	A0000000032010
Visa PLUS	A000000003	8010	A0000000038010
MasterCard Maestro	A000000004	3060	A0000000043060

Para el procesamiento, los terminales de pago compatibles solicitan un AID a la tarjeta, ésta selecciona la aplicación que responde al AID e inicia los parámetros que serán intercambiados con el terminal hasta finalizar el proceso. Los terminales se comunican con la tarjeta usando comandos definidos por EMV (consistentes con ISO/IEC 7816-4), cuyas tareas principales son recibir datos, validar el PIN y firmar las transacciones de autorización de pago.

Para lograr la selección de la aplicación, mencionado anteriormente, se utilizan los comandos SELECT y READ. Ambos comandos tienen una estructura regular de una cabecera y datos. La instrucción SELECT AID enviada por el terminal permite seleccionar la aplicación para operar. En caso que la tarjeta contenga la aplicación, se envía la instrucción READ RECORD para leer los datos y consecutivamente operar con estos.

En cumplimiento de los estándares, los dispositivos móviles en modo de emulación de tarjeta deben cubrir con este proceso, y además contemplar los siguientes atributos, ya que son multitarea, poseen energía y tienen mayor capacidad de almacenamiento:

- a. Si el móvil contiene uno (o varios) SE, debe ser posible instalar aplicaciones en diferentes SE al mismo tiempo, pero solamente una de las aplicaciones con el mismo AID en diferentes SE debe estar activada mediante una billetera (*card manager*).
- b. Debe ser posible seleccionar una aplicación predeterminada para las transacciones.
- c. Similar a las *smartcards*, los pagos deben operar cuando el móvil esté apagado mediante el modo *powered-by-the-field* el cual permite activar al SE con la energía de la inducción generada por el lector. Sin embargo, se deshabilita este modo debido a que: (1) se puede asumir el consentimiento del usuario en distintos contextos cuando la configuración no requiere ingresar PIN previo a la transacción, (2) con la pequeña energía proporcionada no es posible controlar las ventanas de tiempo de activación del PIN por lo que su usabilidad para varias transacciones no es controlada.
- d. Un móvil debe incorporar notificaciones de presencia o alejamiento de lectores NFC y retener la transacción.
- e. Se debe notificar el tráfico entrante para la selección de aplicación mediante los comandos SELECT.
- f. Luego de seleccionar la aplicación mediante su AID, todos los siguientes paquetes APDU (Application Protocol Data Unit) deben ser entregados a esa aplicación hasta que otro SELECT cambie a otra *applet*.
- g. Al recibir un comando SELECT vacío se debe retornar la descripción de las funcionalidades de la billetera, evitando mensajes de error.

3.2. Consideraciones de operación HCE

El funcionamiento de HCE en Android está dotado de particularidades limitantes y otras facilitadoras que permiten un mejor control de la transacción, por ejemplo: Requiere que el móvil esté encendido, el ingreso

del PIN tiene ventanas de duración de tiempo, se muestran mensajes de confirmación del monto y moneda, entre otros. Esto permite una transacción más transparente para el usuario a comparación de hacerlo sosteniendo su tarjeta cerca al lector NFC para un intercambio directo entre ambos.

Entre las particularidades limitantes se puede mencionar a la semántica del comando SELECT en Android para el enrutamiento de las AIDs, el cual no es totalmente compatible a las especificaciones estandarizadas. La selección de algún AID desconocido termina en mensajes de error, limitando el proceso a la aplicación en el intercambio de paquetes.

Otra diferencia es que Android requiere un AID para iniciar una aplicación y no contempla el caso de activar alguna aplicación predeterminada por lo tanto, cuando el lector NFC está configurado para iniciar con el envío de otros comandos propios de la aplicación, Android termina en error. El estado de error también sucede ante un SELECT vacío, aunque lo esperado es un mensaje descriptivo de las funcionalidades. Si bien estas validaciones no representan un inconveniente principal, son derivadas del modelo operativo de HCE.

Haciendo una comparación con *applets* dentro del SE, dos *applets* no pueden tener el mismo ID en el entorno del SE; en cambio, nada impide a un usuario instalar múltiples billeteras que pueden contener las mismas tarjetas virtuales y por lo tanto guardan el mismo identificador AID de aplicación. Por ejemplo, un usuario puede instalar SwipeYours y Tapp (ambas billeteras instaladas en Android) y a cada una de ellas se puede configurar las tarjetas de pago. En este caso, si no se elige una billetera predeterminada, internamente la selección de AID se dirige hacia la billetera normalmente utilizada y las colisiones no detectadas pueden producir alteraciones y finalizar en una transacción inconsistente o estado de error al no identificar a la aplicación.

3.3. Coexistencia HCE y SE

La interacción entre el controlador NFC y la selección de aplicaciones o *applets* se observa en la Figura 7. La idea de registrar todos los AIDs en la tabla de enrutamiento del controlador NFC crearía un desbordamiento de sus capacidades. En Android se utiliza la ruta por defecto del controlador

para el enrutamiento a los servicios HCE y se registra cada AID de los *applets* dirigidos al SE.

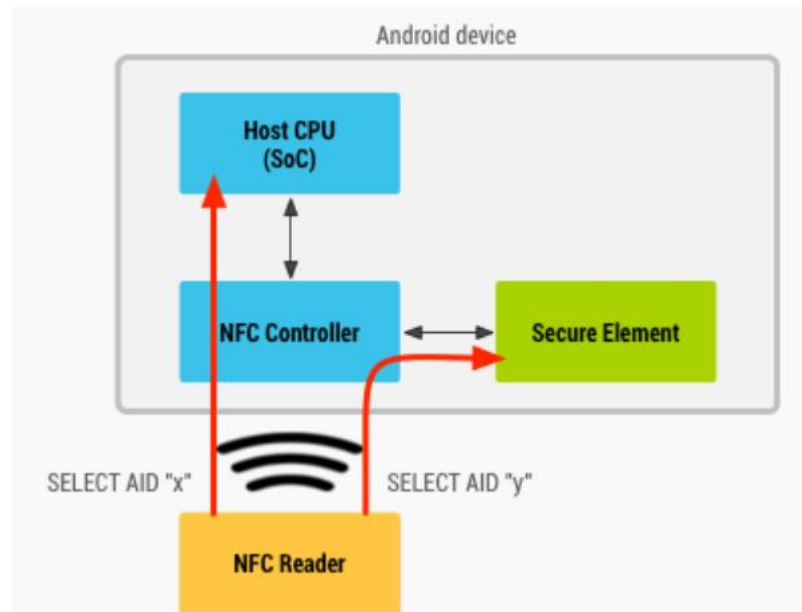


Figura 7: Operación de Android con SE y HCE. Fuente [25]

La tabla de enrutamiento, también conocido como Contactless Frontend (*CLF*) *routing table*, requiere soportar diferentes tipos de enrutamiento, de acuerdo a las especificaciones de GSMA TSG Handset [26]. Estos son:

- a. Enrutamiento por un AID: Se deberá revisar si la aplicación existe en la tabla y en caso no esté presente, la transacción se envía a la aplicación por defecto.
- b. Enrutamiento por tecnología de radiofrecuencia: Permite configurar al CLF con una ruta por defecto para las transacciones, por ejemplo el caso de lectores MIFARE DESFire y MIFARE Classic.

La tecnología MIFARE no usa AIDs en las implementaciones clásicas ni en algunas de las implementaciones DESFire. La implementación de aplicaciones HCE, no permite gestionar los servicios de MIFARE Classic, similarmente con MIFARE DESFire. Es así que HCE presenta limitantes en la emulación de estas tecnologías.

Por otro lado, ya que HCE requiere conocer los AID para el enrutamiento, el registro de los AID de los *applets* deben ser accesibles al CLF tan pronto son descargados al SE. Lo cual no era necesario en el proceso usual.

Es importante mencionar que el firmware del controlador es independiente del entorno del sistema operativo y, por lo tanto Android no puede acceder a las comunicaciones entre el controlador y el SE. De otro lado, es posible que el SO Host pueda intercambiar APDU con el SE mediante una entrada dedicada. Por este canal un malware intentaría enviar comandos hacia SE con los parámetros propios de la interfaz externa NFC. Aunque hasta el momento no se ha vulnerado por la robusta lógica de detección canal-SE implementado de acuerdo a los estándares.

3.4. Ejecución de Servicios HCE

La arquitectura de las aplicaciones HCE están basadas en servicios de Android que pueden ejecutarse sin la necesidad de una interfaz de usuario, aunque es recomendable para las aplicaciones de pago notificar la transacción y registrar los AIDs para evitar colisiones entre aplicaciones.

HCE permite el agrupamiento de AIDs para que puedan enrutarse hacia un servicio. Este grupo se considera una entidad distintiva para Android, es decir, se garantiza que todos los AIDs en el grupo son enrutados al servicio designado y no pueden enrutarse a otro servicio, similarmente un AID externo no puede enrutarse al servicio designado.

Existen también dos categorías; CATEGORY_PAYMENT para las aplicaciones en la industria de pagos y otra para los demás servicios denominada CATEGORY_OTHER. Cada grupo AID puede ser asociado con una categoría y esto permite definir servicios predeterminados a nivel de categorías. Denotando que en la categoría PAYMENT sólo un grupo AID puede ser habilitado en el sistema en el momento de su uso y también se puede definir un servicio predeterminado por lo cual se recomienda que sea la aplicación más adoptada por los comerciantes.

Como toda aplicación en Android, se definen atributos en su manifiesto *AndroidManifest.xml* que permitirán definir los servicios, éstos son:

- a. Dentro de la definición del servicio `<service>` debe solicitarse el permiso para NFC, como: *android.permission.BIND_NFC_SERVICE*.
- b. Dentro `<intent-filter>` se declara la interfaz del servicio como: *android.nfc.cardemulation.action.HOST_APDU_SERVICE*.

- c. Para el caso de los AIDs de las aplicaciones enrutadas al SE, se debe declarar bajo `<intent-filter>` la interfaz como: *android.nfc.cardemulation.action.OFF_HOST_APDU_SERVICE*.
- d. Dentro de `<meta-data>` debe declararse el nombre del recurso XML de donde se definen los grupos AID que son solicitados por el servicio. Este recurso indica el detalle que será mostrado al usuario bajo la etiqueta `<host-apdu-service>`, incluyendo la descripción de los grupos AID, la categoría que pertenece y una lista de AIDs.
- e. Dentro de `<host-apdu-service>` también se precisa si los servicios HCE podrán ejecutarse cuando el teléfono está bloqueado mediante *android:requireDeviceUnlock="false"* de esta manera el dispositivo necesitará estar desbloqueado antes de enlazar los datos.
- f. Especialmente para los servicios HCE que definen la categoría PAYMENT, se debe proporcionar un identificador en imagen en: *android:apduServiceBanner="@drawable/my_banner"*. El motivo es porque Android 4.4 incluye un menú *'tap&pay'* que enumera las aplicaciones de pago y el usuario debe seleccionar la aplicación predeterminada.

3.5. Almacenamiento de credenciales

Las consideraciones anteriores permiten el enrutamiento selectivo e identificado de las aplicaciones de pago. Es así que la única protección para la aplicación es el sistema operativo y por lo tanto no está exento de malware. Básicamente se presume que la información puede ser espiada y debido a ello se evita el resguardo de los datos originales y se opta por la utilización de credenciales que permitirán acceder a estos datos. Entre las propuestas más adoptadas se encuentra *Tokenization* y la implementación de *Cloud of Secure Elements (CoSE)*.

3.5.1. Utilización de Tokens - Tokenization

Es una de las soluciones más adoptadas que permiten pagos seguros en la nube. La utilización de un identificador único *'Payment Token'* (en adelante token) en tiempo limitado reemplaza la necesidad de almacenamiento de las credenciales de pago originales. El token es una

representación del Primary Account Number (PAN), que es el número de 16 (hasta 19) dígitos impreso en las tarjetas el cual permite realizar la transacción en los puntos de venta. El proveedor de los tokens denominado *Token Service Provider* (TSP) debe relacionarlos unívocamente con el PAN y actualizarlos al finalizar la transacción.

3.5.1.1. Token Service Provider

El TSP es el ente responsable de proporcionar los tokens de para cada solicitante que requiera el mantenimiento de las credenciales de pago. El TSP conforma un rol clave en el ecosistema de pagos, puede ser una entidad independiente o parte de la red de pago. Además de cumplir con las especificaciones globales de compatibilidad debe mantener una protección esmerada ante ataques que intenten vulnerar sus bases de claves y funciones criptográficas.

De acuerdo al framework técnico '*Payment Tokenization Specification*' [27] de EMVCo, las responsabilidades incluyen:

- + Generación y aseguramiento del token.
- + Mantenimiento del repositorio de tokens (*Token Vault*).
- + Implementación de medidas de seguridad desde el proceso de registro de la entidad solicitante (*Token Requestor*) y manejo de controles.
- + Desarrollo de las APIs y plataformas de comunicación del Token Requestor.

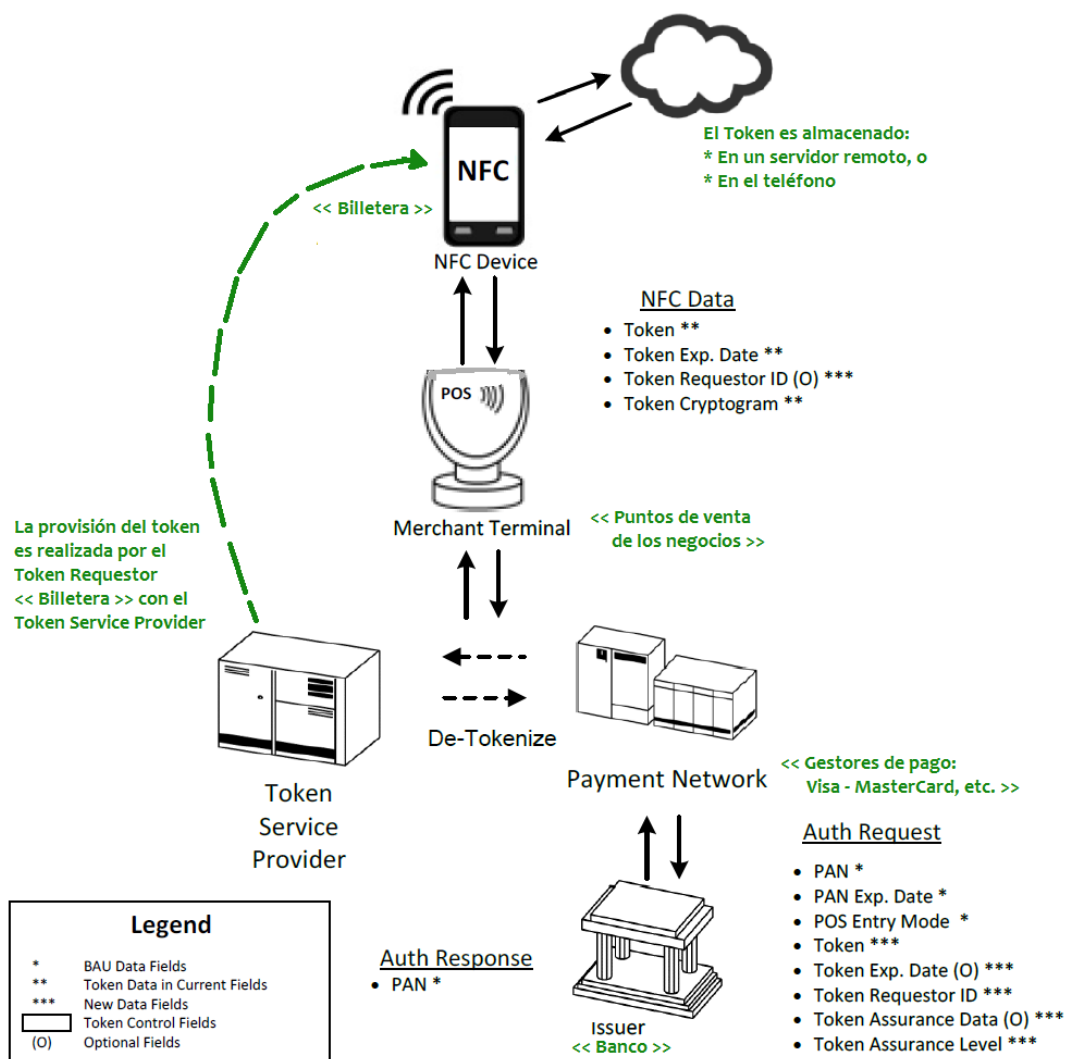


Figura 8: Transacción con tokens de pago. Imagen adaptada de [27]

Previamente a la transacción con tokens, se hace un proceso de enrolamiento del teléfono del Cliente con el banco o Emisor de la tarjeta, que a su vez se interrelaciona con el TSP. La Figura 8, permite identificar los roles en el ecosistema de pagos. El Cliente (Card Holder) transmite el token mediante la Billetera digital (Token Requestor) al Solicitante (Merchant), éste lo transmite al Gestionador (Token Service Provider) que valida la información en la red de pago correspondiente (Payment Network) con el Emisor de la tarjeta (Card Issuer).

El rol del TSP puede corresponder a una misma entidad, por ejemplo, la red de pago que incluya este rol debe cumplir con la definición de los mensajes de autorización, excepciones e impactos del servicio de tokens y soportar las funciones para la *De-tokenization*. Este proceso consiste en obtener el PAN

original a partir del token. Para ello se realiza el mapeo del token en los repositorios y se retorna el PAN identificado.

Por su lado, los solicitantes o TR, que inician los requerimientos de tokens al TSP, pueden ser: Negocios de Card-on-file (entidades que guardan información de la tarjeta en sus sitios), Adquirir Processors (entidades que proporcionan soluciones de Puntos de Venta), gestores de billeteras digitales y emisores de tarjetas.

3.5.1.2. Procesamiento de Tokens

Los tokens presentan atributos que determinan el proceso de la transacción y son definidos en la interacción entre el TR y el TSP.

La generación de tokens no debe entrar en conflicto con el PAN. La identificación de un token con su respectivo PAN debe ser unívoca en el repositorio de tokens. En similitud con las tarjetas, los tokens deben incluir una fecha de expiración y solamente pueden ser usados en el Dominio definido para el TR (por ejemplo una billetera digital) a través de canales o sitios seguros. Durante el registro del TR, el TSP debe asociar cada token con el respectivo identificador del TR y debe validar su emisión directamente con el emisor de la tarjeta. En la solicitud de autorización de la transacción, el TR incluye información denominada *Token Cryptogram* que es únicamente generado por el TR para validar el uso autorizado del token con el TSP y/o el emisor de la tarjeta.

Dependiendo de la aplicación, otros atributos del token indican:

- Validez solamente para montos definidos.
- Limitarse a un único uso o perdurar almacenado en el dispositivo.
- Permanecer activados en determinados horarios y en límites de tiempo.
- Activación solamente para negocios específicos.
- Validez solamente para transacciones vía NFC, entre otros.

Con los atributos mencionados, el token habilita una transacción de pago y limita sus alcances. Es posible incluir una acción para validar la transacción por el usuario o iniciador, aunque en las especificaciones de [27] esta operación puede delegarse al TR.

En el momento que el token es solicitado, debe asegurarse que el PAN es legítimamente usado por el TR, este proceso se conoce como *Identification and Verification (ID&V)* y se ejecuta en cada solicitud para validar la autorización de la transacción. Los métodos generalmente usados son: Verificación de cuenta mediante un código de autenticación (PIN), utilización de sistemas de registro de sesiones, inclusión de APIs para validación del TR, uso de One-Time-Passwords validado por el Emisor de la tarjeta, confirmación de emails en dos vías, etc. Las características de estos medios de validación deben ser por distintos canales (*out-of-band channel*) y no deben ser predecibles.

Es importante implementar un enfoque para el manejo de temporalidad del token. En caso se manejen distintos tokens, debe limitarse tanto su uso en otros canales de pago como su validez temporal en fracciones de tiempo y en horarios, de manera que si un intruso accede a las credenciales del móvil no pueda cometer fraude. En caso se pretenda mantener un único token activo en el móvil, debe permanecer aislado del sistema (caso de Apple Pay que implementa *applets* de pago certificadas en el SE).

Otras soluciones como Google Wallet utilizan tokens de claves de sesión debido a que los datos reales de la tarjeta están almacenados en la nube de Google [28] con lo cual permite registrar las transacciones incluyendo su localización. La provisión de parámetros en el token y las credenciales de la tarjeta, requiere mantener la conexión en la nube y en algunos casos activar otros sensores de ubicación del dispositivo. Por ejemplo, se puede aumentar/reducir el monto transaccional cuando la geolocalización del dispositivo está a una distancia alejada de la usual.

3.5.2. Cloud of Secure Elements

Otra alternativa sujeta al cumplimiento de los estándares de pagos se conoce como Cloud of Secure Elements (CoSE). Ésta es una plataforma dedicada para los servicios NFC que propone utilizar ambientes virtualizados en aplicaciones móviles. La idea básica que se presenta en [29] es acceder a los SE desde servidores remotos y mediante canales TLS.

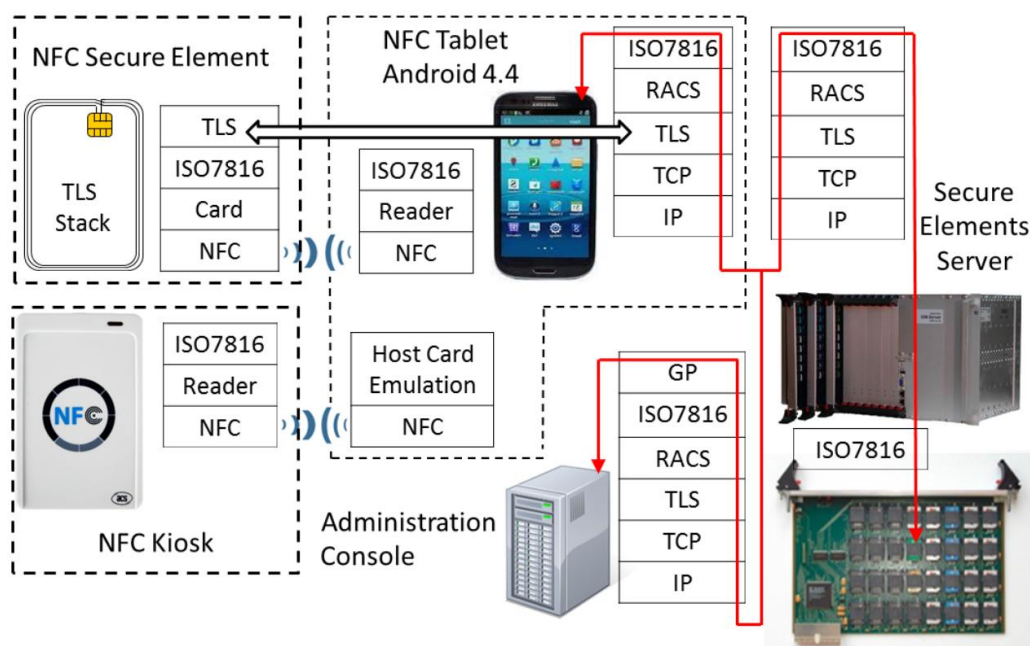


Figura 9: Plataforma de CoSE. Fuente [29]

La Figura 9, presenta la plataforma de CoSE, que está compuesta por cinco elementos: El lector NFC, una tarjeta que almacena la pila TLS y los credenciales, un móvil Android en modo HCE y el servidor de SE.

La comunicación se realiza mediante el protocolo RACS (Remote APDU Call Secure), el cual trabaja en la capa de aplicación con medidas de seguridad del protocolo TLS mediante un certificado de autenticación.

El punto clave en esta infraestructura es la apertura de sesión para iniciar un canal seguro entre el teléfono móvil y el servidor que resguarda los SE. Para ello, ya que las credenciales no pueden estar albergadas en el móvil, la interacción se maneja mediante una tarjeta NFC que iniciará la sesión con el servidor de SE.

El conjunto de claves incluidas en esta tarjeta NFC corresponden a dos pares de claves para el cifrado y la integridad. Tanto la tarjeta NFC como el servidor son mutuamente autenticados con sus certificados y claves privadas. En esta comunicación, una API (NFC TLS API) permite que el móvil Android enlace mensajes (paquetes RACS), habilite el socket, descifre los paquetes y responda al servidor remoto. Una vez que el canal se haya abierto, la sesión es transferida al móvil junto a la suite de cifrado y las credenciales que harán posible proseguir la transacción.

Por su parte, el servidor remoto mantiene el inventario de SE y los identifica con un SEID. Desde la consola administrativa ISD (Issuer Security Domain) se permite acceder, modificar y actualizar los *applets*. Es importante mencionar que el administrador remoto de esta consola debe pasar por mecanismos de autenticación y autorización previos para cada acción.

CAPÍTULO IV: Implementación de un caso de uso

El presente caso de uso corresponde a un pago seguro en un entorno HCE. El alcance responde a escenarios de transacciones rápidas y montos pequeños, donde no se requiere incluir otros medios de validación adicionales a la presentación de las credenciales del teléfono y donde las expectativas del usuario son seguridad, control y registro de sus pagos digitales al alcance de su teléfono.

En este sentido, se presentarán los elementos participantes y los aspectos de seguridad de SimplyTapp [30]. Esta es una organización que provee servicios a las entidades emisoras de tarjetas para procesar pagos en la nube. Se ha escogido esta solución porque a la fecha del presente trabajo está operativa e implementada en entornos reales [31] [32]. La arquitectura HCE de SimplyTapp es aplicable a las industrias de ventas, bancos, tickets, tránsito, etc. y proporciona una interfaz web y archivos de instalación para que un desarrollador pueda experimentar con las fuentes de la aplicación móvil. Adicionalmente, proporciona herramientas que se adecuan a la comprensión del presente caso de uso. Cabe indicar que para poder realizar estas acciones se requiere el registro en el sitio web de SimplyTapp.

Esencialmente, en el presente caso de uso se revisará la generación de la tarjeta virtual por parte de la entidad emisora, la autorización de esta tarjeta al proveedor de tokens, la incorporación de esta tarjeta a la aplicación móvil (billetera virtual) que utilizará el usuario, el proceso de envío de tokens hacia el dispositivo móvil y finalmente, el procesamiento de los tokens para la generación de criptogramas que son comunicados al terminal POS.

Es importante mencionar que SimplyTapp está definida para trabajar con terminales POS reales, por lo tanto la implementación práctica para la comunicación con el POS experimental (es decir, lector NFC y computador) fue resuelta con la aplicación de librerías y fuentes obtenidas de [33], de tal manera que pueda lograrse el paso de tokens.

A continuación se enlistan los componentes que interactúan en el proceso de una transacción y seguidamente se presentará el detalle de las acciones.

4.1. Descripción de los componentes

En la Figura 10 se muestra un esquema de interacción de los componentes para el caso de uso de un pago seguro en HCE.

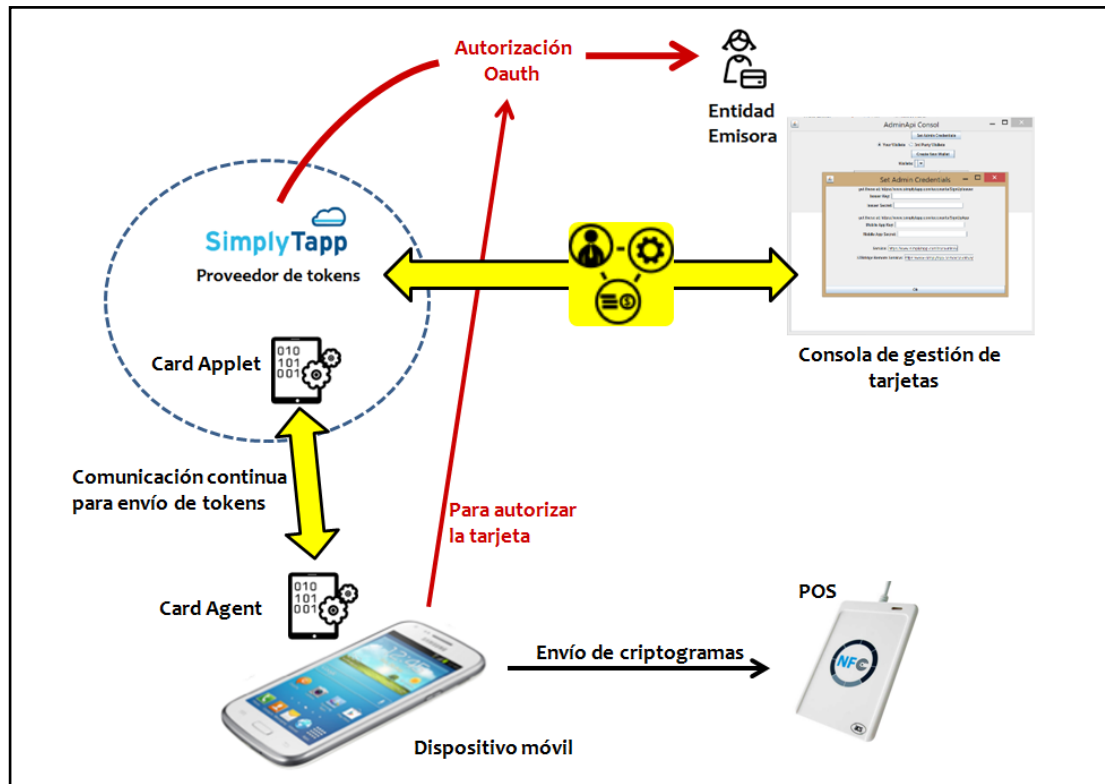


Figura 10: Interacción de los componentes

(A) Kit de desarrollo

Dentro de este Kit [34] se incluye el Issuer SDK y el Mobile SDK. El Issuer SDK está definido para una Entidad Emisora, permite crear credenciales (Card Applet y Card Agent) que en este caso conforman la tarjeta de pago. Incluye una consola de gestión que permite controlar el ciclo de vida de las tarjetas.

El Mobile SDK está definido para crear la billetera o aplicación móvil que después de las autorizaciones pertinentes, almacenará las tarjetas del usuario (específicamente al Card Agent porque el Card Applet estará en la nube).

(B) Card Agent

Se ejecuta en cada dispositivo móvil y representa a un Card Applet específico. Es responsable de responder a todas las solicitudes realizadas, es la primera interfaz de respuesta cuando el lector interroga al dispositivo, comprende la lógica del Applet remoto correspondiente manteniendo una

comunicación continua. Este Agent puede responder directamente al POS, aunque en otros casos necesitará interactuar con el Applet remoto para obtener respuesta. Este Agent realiza consultas en tiempo no-real al Applet remoto, es decir, en tiempos previos al inicio del contacto con el POS. Estas consultas previas son importantes para resolver inconvenientes de rendimiento y latencia de comunicaciones en tiempo real con un Applet remoto. Esencialmente, este Agent actúa como intermediario del Applet remoto.

(C) Card Applet

Para efectos de pago, estos Applets cumplen con especificaciones de tarjetas como: MasterCard, Visa, EMV, American Express, etc. Este Applet es cargado dentro de las tarjetas con chip. Para este entorno, el comportamiento es el mismo con la diferencia que el Card Applet reside en la nube y es creado unívocamente con un Card Agent para conectar individualmente a cada dispositivo.

(D) gpjNG Shell

Global Platform Java Card Next Generation Shell (gpjNG) es una shell usada como herramienta para la comunicación con los Applets, por comandos de solicitud y respuesta APDU. Esta shell es instanciada desde la consola AdminApi Console proporcionado en el SDK de SimplyTapp. Soporta un compendio de comandos para tareas como: Autenticación, selección del Applet, ejecución de scripts hacia el Applet, etc.

(E) Protocolo OAuth 2.0

Es un protocolo para permitir autorizaciones seguras para aplicaciones web, móviles y de escritorio [35]. Para los proveedores de servicio que manejan credenciales, OAuth permite que los usuarios proporcionen acceso a sus datos mientras se mantienen protegidas sus credenciales. Para el presente caso, OAuth es utilizado para autorizar a la aplicación móvil la capacidad de contener tarjetas virtuales y también permite la autorización de las tarjetas con cada entidad emisora cada vez que se intente incluir tarjetas en la aplicación.

(F) NFC Tools

Es un conjunto de librerías implementadas en Java para la comunicación NFC. Es compatible con el Kit de desarrollo de Android SDK-19 (y superiores) que corresponde a la versión del Smartphone Samsung.

(G) Lector ACR-122U

Se encuentra en [36]. Representa al terminal POS junto al computador y soporta tarjetas del estándar ISO 14443.

(H) Smartphone Android

Samsung S4 mini modelo GT-I9195. No rooteado. Versión KitKat 4.4, el controlador NFC está incluido en la placa y la antena NFC en la batería.

4.2. Interacción detallada de los componentes

A continuación se presentan las actividades necesarias en orden de ejecución para lograr la interacción de los componentes que compromete el logro de una transacción segura:

1. Registro en el sitio Web de SimplyTapp como:
 - 1.1. Entidad Emisora
 - 1.2. Desarrollador de la aplicación móvil
2. Compilación de la aplicación móvil y autorización con la Entidad Emisora
3. Creación de la Tarjeta (Card Applet y Card Agent)
4. Autorización de incorporación de la tarjeta a la aplicación móvil
5. Mantenimiento de la seguridad en el ciclo de vida de la tarjeta
6. Construcción de criptogramas y seguridad de la transacción

A continuación se presenta una descripción a detalle:

1. Registro en el sitio Web de SimplyTapp

1.1. Entidad Emisora

Este proceso inicia con la creación de una cuenta en SimplyTapp. Seguidamente se debe crear una Entidad Emisora, como se muestra en la Figura 11, es necesario indicar el nombre y una imagen que será el ícono representativo de la Entidad. Al momento de guardar la Entidad, se crea automáticamente un identificador y las respectivas claves: *Issuer Consumer Key & Issuer Consumer Secret*. En este caso, no fue necesario cambiar el

servidor de Tarjetas para posteriormente autenticarse a esta Entidad Emisora. La sección *Add Brand* permite añadir tarjetas y será detallada más adelante para no perder el flujo del proceso.

Hasta este punto se resalta la importancia de las claves generadas, debido a que en el momento de solicitar autorización, se hará una invocación a la Entidad Emisora únicamente a través de estas claves.

The screenshot shows the 'Your Issuing Entities' page in the SimplyTapp application. The page has a sidebar with navigation links: Home, Developer, Issuer, Usage, and Account. The main content area shows the details for an entity named 'Edith-UBA'. The details include:

- Issuer Name: Edith-UBA
- Issuer Logo: A placeholder image with a button labeled 'Examinar...' and a message 'No se ha seleccionado ningún archivo.'
- Access Token Callback: <https://www.simplybank.us/demo/getAccessToken.php>
- Issuer ID: 1287
- Issuer Consumer Key: a9rUeMpgR53nlrbDAD7CPw8whmFfrxjYPI7ZliZ7
- Issuer Consumer Secret: 6mQR5RhHXHI42vms8Mql6iCojPlbiKoJA3FlwMDc

At the bottom of the details section, there is a button labeled 'Configure Your Card Server' and a green button labeled 'Save Entity Changes'. Below the details section, there is a section titled 'Your Card Brands' with a button labeled 'Add Brand'.

Figura 11. Creación de la Entidad Emisora

1.2. Desarrollador de la aplicación móvil

Similar a lo anterior, debe crearse una cuenta de Desarrollador. Como se muestra en la Figura 12, el desarrollador registra una aplicación móvil, dando un nombre y un ícono distintivo. Seguidamente, se crean las claves: *App Consumer Key* & *App Consumer Secret*, las cuales identificarán a la aplicación móvil.

The screenshot shows the 'Your Mobile Applications' page in the SimplyTapp interface. The sidebar on the left contains navigation links: Home, Developer, Issuer, Usage, and Account. The main content area displays the details for an application named 'Tapp-dev'. The details include:

- App Name:** Tapp-dev
- App Logo:** A placeholder image with a 'DEV' badge and a message: 'Examinar... No se ha seleccionado ningún archivo.'
- x-oauthflow Callback:** x-oauthflow://st_oauth_callback_tapp_dev
- App Consumer Key:** 9uuELvhBlcrZmhop9TuEi3ZQTGuEguVo8rJWCEct
- App Consumer Secret:** p2mKuo2t3UhBAN1WKI9cbsUnYQF3CBsWculHlpNm

A 'Save Changes' button is located at the bottom of the application details section.

Figura 12. Registro de la aplicación móvil

2. Compilación de la aplicación móvil y autorización con la Entidad Emisora

Se utilizó el Mobile SDK para compilar la aplicación. Fue necesario activar las opciones de desarrollador y habilitar la instalación de aplicaciones de fuentes terceras. Se logró la instalación de la aplicación “*Tapp-dev.sdk*”. En la Figura 13, se observan las pantallas de la aplicación móvil desde su apertura.

En la primera pantalla, la aplicación solicita al usuario determinar el método predeterminado de pago para las transacciones vía NFC. Esta consideración de seguridad se notificó porque previamente se tenía instalado otra aplicación billetera de prueba “*SwipeYours*¹³” y en consecuencia es necesario elegir una aplicación predeterminada.

La segunda pantalla, es lanzada posterior a la aceptación. La aplicación pregunta al usuario si quiere autorizar a la aplicación para ser contenedora de tarjetas y realizar pagos en un terminal POS. Internamente, el proceso está basado en los estándares de OAuth. Es así que, desde la página de

¹³ SwipeYours es una versión demo que también pertenece a SimplyTapp.

SimplyTapp se verifica que la aplicación es auténtica. Una vez que el usuario la haya aprobado, la aplicación terminará con las claves otorgadas por OAuth.

Hasta el momento la aplicación solamente está permitida de acceder a una tarjeta remota, pero es necesario agregarla mediante otro proceso. Esto se puede observar en la tercera pantalla que solicita la adición de tarjetas.

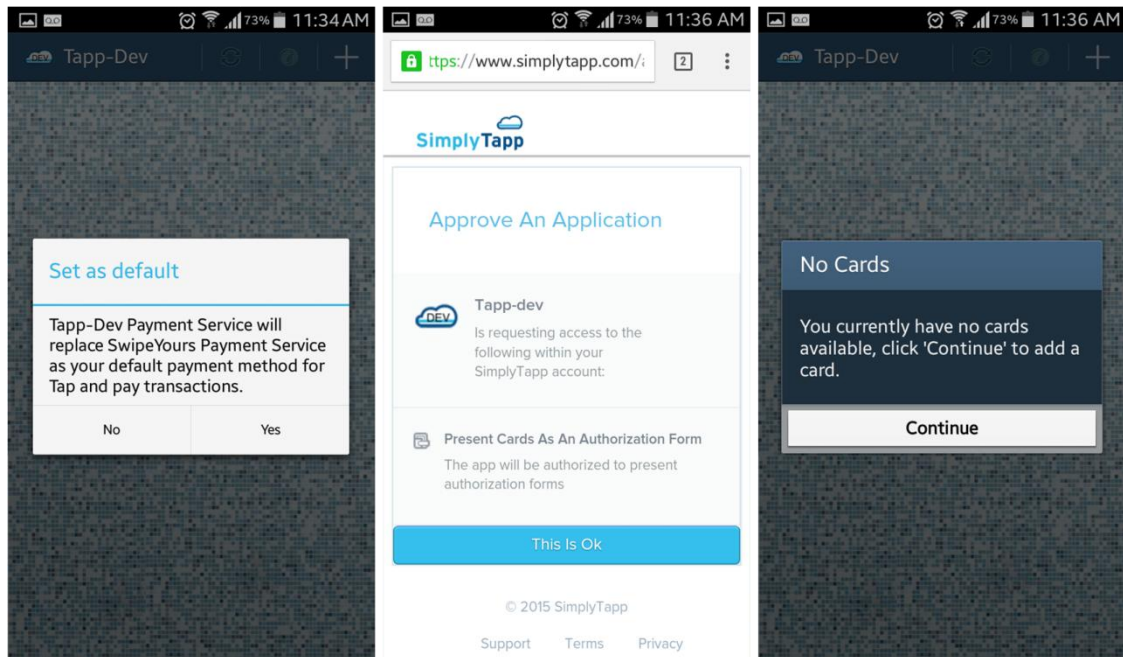


Figura 13. Aplicación móvil HCE

En este punto la aplicación móvil (billetera) está aprobada por el usuario para realizar pagos en los POS con el conjunto de tarjetas que pueda contener. El cliente debe aplicar a cada Entidad Emisora para obtener una tarjeta digital.

3. Creación de la Tarjeta (Card Applet y Card Agent)

El Issuer SDK trae consigo los recursos para compilar tanto el CardAgent.jar como el CardApplet.jar, que representan a la tarjeta de algún usuario. La Entidad Emisora es responsable de generar la tarjeta y posteriormente sube estos archivos a la nube. En este caso a la nube del proveedor de tokens como se muestra en la Figura 14.

El Issuer SDK ofrece distintos tipos de ejemplos para compilar las tarjetas, de los cuales se escogió CardVisaMSD-SwipeYours. Similarmente, se crea

un hash de la versión subida y un par de credenciales de acceso (Token Access y Token Secret) para ser validados cuando el usuario solicite añadir esta tarjeta a la aplicación (no se refieren a los tokens utilizados para una transacción).


Tarjeta Edith

Brand Details

Card Brand

Tarjeta Edith

Card Brand Logo



Examinar...

No se ha seleccionado ningún archivo.

Card Brand ID

1066

Acquire Card Url

none

Update Card Brand

Applet Version

1.0
(Agent MD5: e8d40094d9357dde05368a4bdf8ee627)

Card Applet Jar

CardApplet-VisaMSD-SwipeYours.jar

Card Agent Jar

CardAgent-VisaMSD-SwipeYours.jar

Test Card Access Data:

Reset Test Card

Test Card Access Token

dvcR8xe9i80b4aVIBymvzUtShlx6THsAMwuNX2Px

Test Card Token Secret

hRxWXduh67dkJLOeIEiEKqJuaZBDowBoB1kd4eAe

Test Utility

STBridge (java -jar STBridge.jar --help)

Utility Connection:

java -jar STBridge.jar -ck
a9rUeMpgR53nlrbDAD7CPw8whmFfrxjYPI7ZliZ7 -cs
6mQR5RhHXHI42vms8MqI6ICojPIbiKoJA3FlwMDc -at
dvcR8xe9i80b4aVIBymvzUtShlx6THsAMwuNX2Px -ts
hRxWXduh67dkJLOeIEiEKqJuaZBDowBoB1kd4eAe -service
https://www.simplytapp.com/accounts/Api

Deprecate Applet

Figura 14. Creación de una Tarjeta de Pago en la Entidad Emisora

4. Autorización de incorporación de la tarjeta a la aplicación móvil

El proceso de aprobación es similar a la anterior. Como se muestra en la Figura 15. La aplicación móvil mediante OAuth enlaza al sitio web de la Entidad y la primera pantalla muestra a Simply Bank como la Entidad inicial de prueba. Sin embargo, para acceder y autenticar a la Entidad que previamente se creó, es necesario completar la información de las credenciales obtenidas en la Figura 11 bajo la sección "Issuer OAuth Configuration".

The figure consists of two side-by-side screenshots of a mobile application interface.

Left Screenshot: The browser address bar shows `https://www.simplybank.us/de`. The page title is "SimplyBank". Below the title is the heading "SwipeYours Mobile Card Application". The form is divided into two sections:

- GENERAL INFORMATION** (Non-editable Demo-only fields):
 - Full Name: John Smith
 - Email: john.smith@example.com
 - Phone Number: +1-512-555-1234
 - Billing Address: 678 NFC Payments Lane
 - City: Austin
 - State: TX
 - Zip Code: 78701
- Issuer OAuth Configuration** (Copy the values from the Issuer section of your SimplyTapp account):
 - Issuer Consumer Key: a9rUeMpgR53nirbDAD7CPw8whmFrxjYPI7ZiIz7
 - Issuer Consumer Secret: 6mQR5RhHXHl42vms8MqI6iCojPibiKoJA3FlwMDc
 - Card Brand ID: 1066
 - Hide and use default SwipeYours Issuer: [checkbox]

A "Send Application" button is at the bottom.

Right Screenshot: The browser address bar shows `https://www.simplytapp.com`. The page title is "SimplyTapp". The heading is "Approve An Application".

- Edith-UBA** (Is requesting access to the following within your SimplyTapp account):
 - Create A Card: The app will be authorized to create a
 - Tarjeta Edith** (card inside one of your mobile apps)
- Please Pick One Of Your Apps:**
 - Tapp-dev (DEV)
 - This One** (highlighted)

Figura 15. Proceso de aprobación para añadir una tarjeta

Seguidamente el proveedor de los tokens, en este caso SimplyTapp, solicita al usuario aprobar el adiconamiento de una tarjeta a su aplicación móvil mostrando al usuario la tarjeta virtual que previamente fue creada. Solamente después de la autorización del usuario, la aplicación cuenta con las credenciales de acceso para conectarse a la tarjeta.

Finalmente, el sitio retorna a la aplicación móvil, que ahora cuenta con una tarjeta virtual remota para conectarse. Internamente, se ha instalado el Card Agent en el dispositivo móvil que durante las transacciones estará continuamente comunicado con sus respectivo Card Applet en la nube.

5. Mantenimiento de la seguridad en el ciclo de vida de la tarjeta

El manejo del ciclo de vida de las tarjetas virtuales en el sistema de SimplyTapp se realiza desde una consola de AdminApi Console mostrada en la Figura 16. A través de ella, la Entidad Emisora puede controlar la tarjeta virtual cuando el dispositivo móvil es susceptible a algún robo o vulnerabilidad. Esta herramienta utiliza comandos de paquetes APDU para identificar y acceder a una tarjeta virtual conociendo las credenciales de acceso.

Se definen cuatro estados:

- **Creación:** Se crea la tarjeta.
- **Desactivación:** El Card Agent estará desactivado para realizar transacciones.
- **Activación:** Inverso a la desactivación, se activa nuevamente el Card Agent.
- **Finalización:** En este estado, el Card Agent queda deshabilitado para responder a las transacciones de pagos. No se ejecuta ni aún cuando se refresque o reinicie la aplicación. Este estado no puede cambiarse una vez alcanzado.

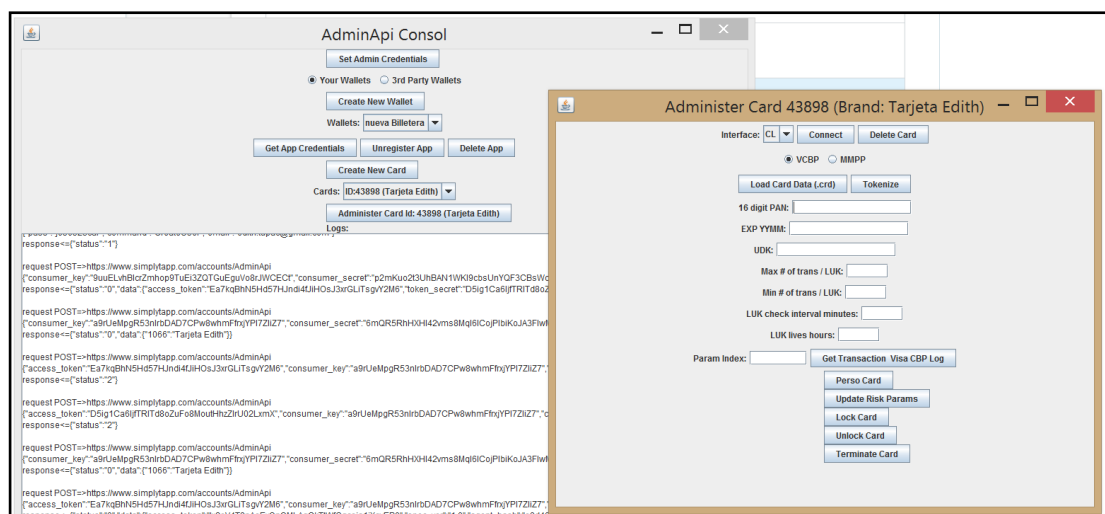


Figura 16. Consola de administración de tarjetas

6. Construcción de criptogramas y seguridad de la transacción

Es necesario tener en cuenta los siguientes puntos:

Hasta el momento, la Entidad Emisora y el Proveedor de Tokens están conjuntamente autenticados y por lo tanto ha sido posible realizar la creación y autorización de las credenciales OAuth (de clave y acceso). El resguardo y generación de tokens para las transacciones en HCE procederán en este caso de la nube de SimplyTapp aunque puede implementarse para que sea exclusivamente desde la nube de la Entidad Emisora.

Las transacciones que usan tokens tienen el mismo formato de datos que una transacción "no tokenizada" (con tarjetas físicas) con el objeto que sea comprendida por el POS. Estos datos son: PAN, Fechas, código del servicio, datos de la Entidad y finalmente la generación del criptograma.

El criptograma es un valor calculado de un sólo uso para cada transacción y no puede ser usada más de una vez.

En un entorno común (no HCE) este criptograma es calculado en un algoritmo que conjuga: una clave única derivada de una clave maestra propia de la entidad bancaria junto al UN (Unpredictable Number), este dato es proporcionado por el POS para la creación final del criptograma, el cual es retornado al POS.

La obtención final del criptograma en HCE es similar al descrito anteriormente, con la única diferencia que la clave única derivada es en realidad un token tLUK (Tokenized Limit Use Key) proporcionado por el proveedor de tokens y aceptado por la Entidad Emisora por los mecanismos de autorización previamente mencionados.

Para el presente caso de uso, es necesario describir la seguridad de cómo el token llega al dispositivo móvil y cómo se garantiza la integridad en su utilización.

La propuesta de SimplyTapp para los dispositivos Android es hacer uso de Google Cloud Messaging (GCM) como se muestra en la Figura 17. De tal manera que se el paso del mensaje para el cálculo del token tLUK se realice por dos canales distintos.

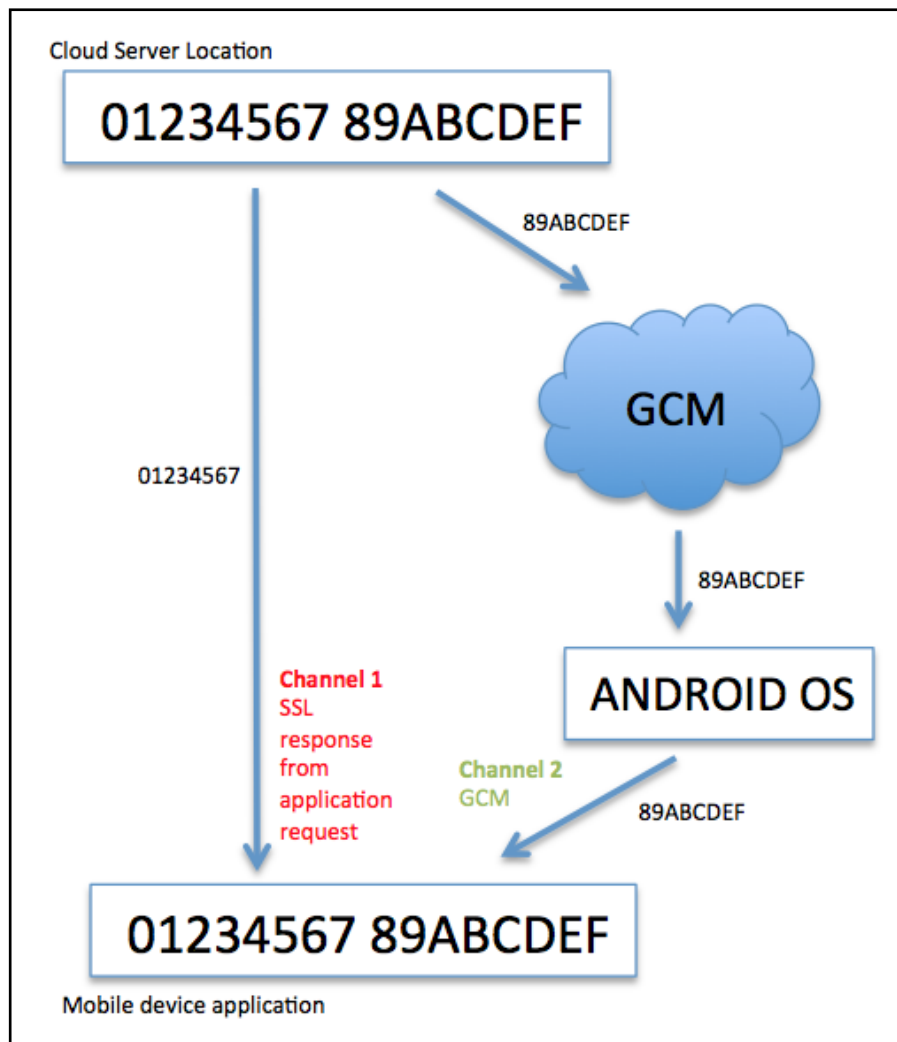


Figura 17. Esquema de GCM para aplicaciones HCE. Fuente [37]

Inicialmente es necesario que la aplicación móvil (Tapp-Dev) esté registrada en Google para utilizar las APIs y obtener un identificador único en GCM. Seguidamente, cuando la aplicación es descargada e instalada por el usuario, el dispositivo Android envía un SENDER ID y APPLICATION ID hacia los servidores de Google, quien devuelve un REGISTRATION ID al dispositivo. Este último ID es resguardado en la nube de SimplyTapp. En este sentido, cada vez que se necesite enviar una notificación, desde la nube envía el mensaje junto al REGISTRATION ID y finalmente el servidor de Google entrega ese mensaje al respectivo dispositivo.

El servidor Cloud Server Location corresponde a la nube de SimplyTapp. Este servidor mediante el Card Applet genera un mensaje que es dividido en dos. La mitad es enviada hacia el dispositivo desde el servidor en la nube mediante un canal seguro Transport Layer Security (TLS) y la otra mitad es

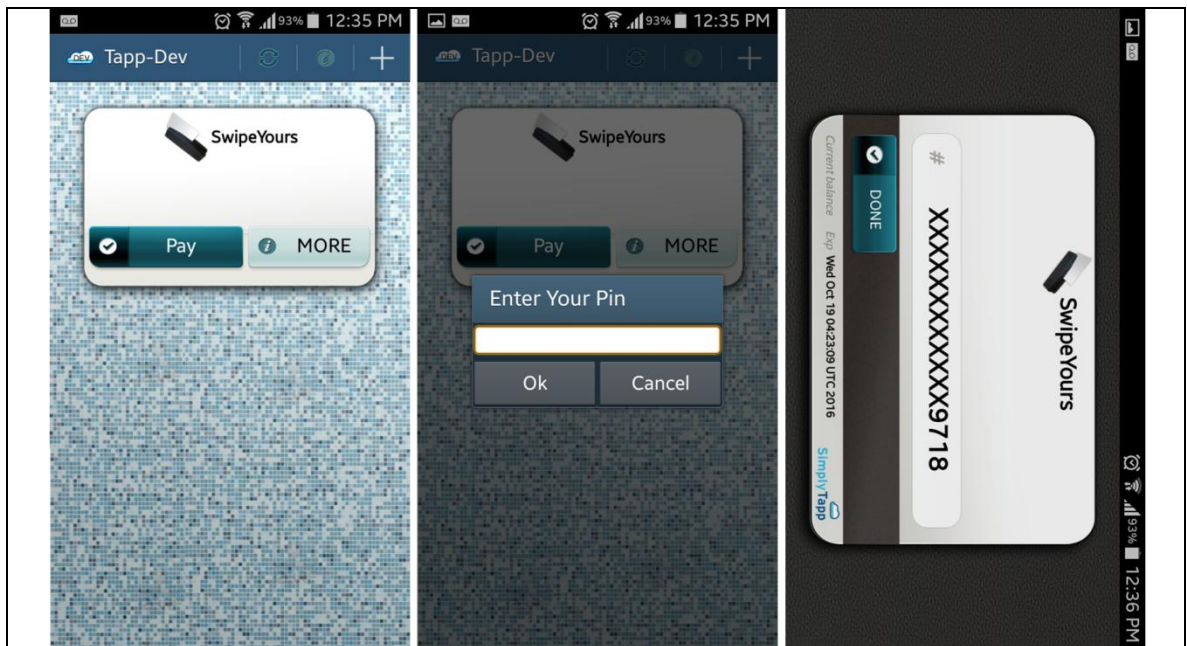
enviada mediante una notificación por la red GCM hacia su respectivo Card Agent. Éste recibe la notificación y junto al primer mensaje construye Tokenized Limit Use Keys (tLUK) que permite el cálculo del criptograma en el momento del pago.

Para evitar problemas de latencia de red en el momento de la transacción, es posible que el Agent se comuniquen con el Applet en momentos previos a efectuar un pago para solicitar uno o más tokens (tLUK) que serán usados en la construcción de los criptogramas. El tiempo de vida de cada token y el intervalo de tiempo entre solicitudes dependerá de las configuraciones de la Entidad Emisora y el proveedor de tokens.

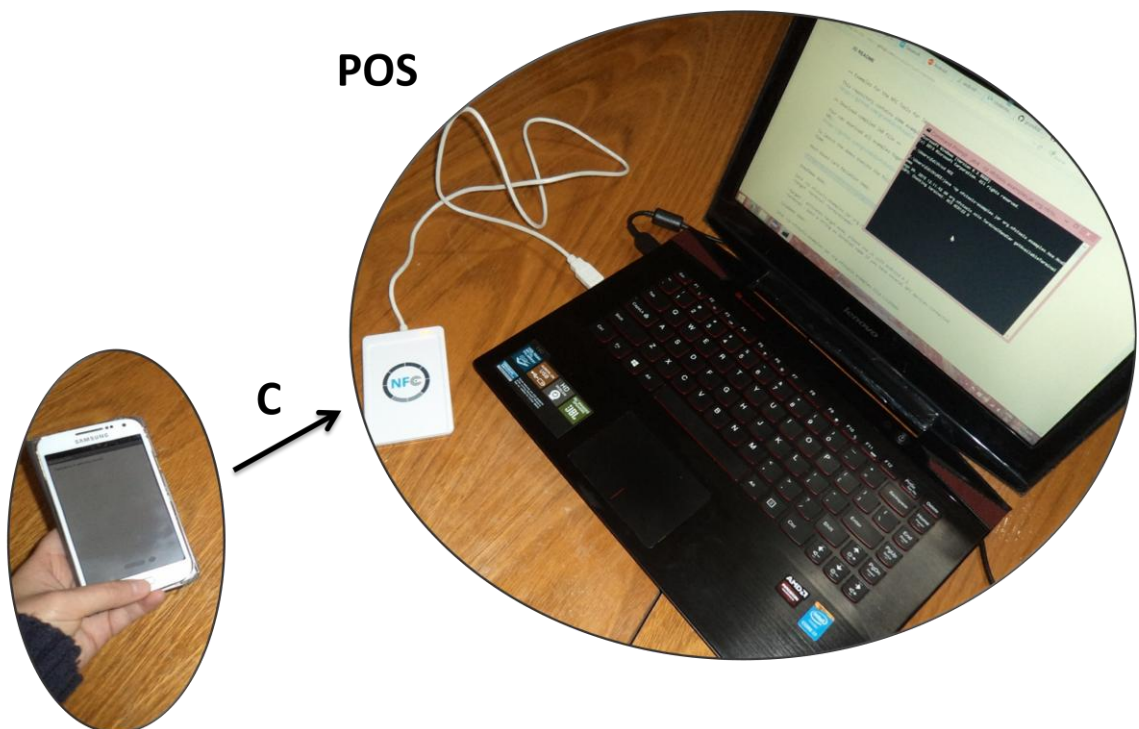
En la Figura 18, se observa una reproducción del momento de una transacción, la aplicación está configurada con una tarjeta añadida de SimplyTapp que tiene datos de prueba. En la parte (A), se solicita la entrada de un pin para activar la tarjeta. Este pin es parte de la seguridad de la aplicación y no está relacionado con la clave de la tarjeta que comúnmente se maneja. La parte (B) corresponde al momento que el teléfono es aproximado al terminal. El computador y el lector NFC corresponden al terminal de pago (POS), el computador ejecuta un programa que reconoce la interfaz del Lector en el puerto USB y envía un comando de selección de la aplicación proporcionando el identificador AID y permanece en espera de respuesta de la aplicación.

Cuando Android es acercado al terminal, se reconoce la aplicación y se inicia el envío del token, como es mostrado en la parte (C).

En este tramo final y con el propósito de hacer interferencia para entorpecer la señalización del lector NFC durante la lectura del token, se colocaron tarjetas (compatibles con el lector), sin embargo esto no causaba la interrupción en la selección del identificador AID.



(A) Activación por PIN de la tarjeta de prueba



(B) Interacción Android y lector NFC prueba de transmisión de tokens

```
INFO : org.nfctools.examples.hce.IsoDepTamaCommunicator - TargetData: 0100042004
082CB4110578B37002
INFO : org.nfctools.examples.hce.IsoDepTamaCommunicator - IsoDep Supported
INFO : org.nfctools.examples.hce.IsoDepTamaCommunicator - Received: Hello Desktop!
INFO : org.nfctools.examples.hce.IsoDepTamaCommunicator - Received: Message from
android: 4vu9m3nbcs3g8n3rpc4gult21b
INFO : org.nfctools.examples.hce.IsoDepTamaCommunicator - Received: Message from
android: tjgttl1hjqtn9gggo0to0n7gdq5c
```

(C) Envío de tokens hacia el POS

Figura 18. Prueba de transacción HCE

4.3. Seguridad en aplicaciones de pago HCE

Un entorno HCE debe proporcionar seguridad en sus tres componentes básicos: en el dispositivo Android, comunicación en el tramo aplicación-POS y la infraestructura en la nube.

En la nube son críticos tanto el almacenamiento de credenciales como el paso de mensajes de la transacción. Al tratarse de medios de pago, es indispensable implementar protocolos seguros para todo enlace. A esto se añaden otros requisitos de compatibilidad plasmados en estándares internacionalmente conocidos. Por otro lado, además de una relación de confianza directa con la entidad bancaria, se hace presente una nueva relación con un Proveedor de Confianza quien es responsable de la gestión de credenciales, tokens y resguardo de información para las transacciones bancarias. Estas relaciones y acuerdos deben ser evaluados cuidadosamente por las entidades bancarias ya que la información comprometida puede producir problemas transaccionales a gran escala.

En el tramo físico entre el lector NFC y el teléfono móvil, se hace presente una vulnerabilidad cuando un atacante incorpora mecanismos para escuchar o corromper el pase de datos entre ambos. Este escenario puede ser evitado asegurando que los POS eviten la proximidad con otros objetos o componentes, especialmente metales o conductores que interfieran con ondas de radiofrecuencia.

El aseguramiento de una aplicación HCE debe cubrir la protección de datos críticos, sus funciones criptográficas y su aislamiento controlado. Especialmente en Android las funciones de seguridad que vienen predeterminadas en cada versión pueden ser vulneradas si se obtiene acceso de superusuario mediante el rooteo del móvil. Esto afectaría a los servicios HCE, especialmente en el resguardo de llaves y clonación de credenciales. A continuación se presentan estas funcionalidades y otras reportadas en [38] que sirven de guía para la implementación de aplicaciones NFC:

- La aplicación debe integrar técnicas de tamper-proofing y ofuscación de código para ser resistente al análisis, manipulación de código e ingeniería reversa. Asumiendo la desconfianza de la instalación y ejecución en el SO del dispositivo.

- Android proporciona un repositorio seguro (*Android Keystore*) que permite a cada aplicación encapsular sus propias credenciales y protegerlas del resto. Sin embargo, un teléfono rooteado puede vulnerar esa seguridad y por lo tanto afectar el resguardo de los tokens un entorno de tokenización.
- Desde el dominio del SO, un malware en un teléfono rooteado puede saturar la tabla de enrutamiento que discrimina las AIDs de los *applets*. Esto puede causar la denegación del servicio de otra aplicación legítima de pago.
- La seguridad en el envío de mensajes que servirán para las transacciones debe realizarse en lo posible por canales distintos. De tal manera que una vulnerabilidad en uno de ellos, no corrompa todo el proceso transaccional.
- Las aplicaciones HCE pueden ser afectadas por la clonación. Por ello, es crítico saber en qué dispositivo está siendo usado. Algunas aplicaciones incluyen soluciones de IDs, entonces cuando se activa la inscripción, se genera un ID del dispositivo asegurando que no se pueda replicar el estado del dispositivo e invalidar una clonación. El identificador permitirá ejecutar la aplicación solamente para ese dispositivo en particular. Otra solución para evitar la utilización del token clonado es recurrir la interacción del usuario adicionando un factor de autenticación.
- Si las credenciales de pago quedan vigentes en corto tiempo, se volverán inservibles cuando la transacción se complete. Si un intruso alcanza esa información no podrá volver a usarla. De ahí que la determinación de la temporalidad dependerá de la tolerancia al riesgo.
- El suministro de datos en internet (Over the Internet-OTI) hacia la aplicación como PAN temporales, fecha de expiración, claves criptográficas de sesión, etc. deben realizarse sobre canales seguros (TLS).
- Otro aspecto importante es que las claves privadas y algoritmos de cifrado pueden ser dinámicamente actualizadas, individualmente en

cada aplicación sin que se requiera la participación del usuario o actualizaciones desde el repositorio.

- La robustez de la seguridad en la nube no debe afectar los requisitos de velocidad de una transacción. Si los mecanismos de autenticación que procesa o recibe la aplicación superan el intervalo de tiempo (generalmente 400ms) no podrá concretarse la transacción.

5. Conclusiones

1. El beneficio de seguridad de NFC es la proximidad, es decir la estrecha distancia entre los dispositivos representa un nivel de confianza que no requiere descubrimiento ni asociación explícita a comparación de Bluetooth.
2. La realización de pagos vía NFC-HCE es una alternativa de mucho interés especialmente para las transacciones de montos bajos y en establecimientos que requieren rapidez en la transacción. Los riesgos son admisibles y se incrementa la experiencia del usuario.
3. Es importante mantener cada aplicación aislada (*sandboxing*) de otra, especialmente limitar los accesos y privilegios para procesos y objetos desde su instalación. Esta característica está incorporada desde la versión 4.3. de Android (Security Enhancements for Android), sin embargo los dispositivos rooteados y no controlados son vulnerables al escalamiento de privilegios.
4. La implementación de la tokenización de credenciales es un mecanismo más aceptado y recomendado para la implementación de aplicaciones de pago. A esto, se puede incluir un pin o clave como un factor de autenticación que habilite el token al momento de una transacción.
5. La prestación de servicios en la nube para una transacción de pago debe implementar fundamentalmente mecanismos de autenticación fuertes y conducir transacciones seguras que no limiten el rendimiento.
6. La red de conexión hacia la nube debe ser altamente disponible y protegida por canales seguros, de lo contrario el almacenamiento de credenciales tanto en HCE como CoSE es afectado durante una transacción de pago y las oportunidades de fraude son admisibles.
7. La localización del usuario es un tema sensible con respecto a la privacidad ya que la transacción y la actualización de tokens desde la nube permite también registrar los movimientos del usuario.
8. A diferencia de las transacciones de pago online, existe un reto especial para los pagos móviles vía NFC porque pueden realizarse

sin interacción del usuario, y deben incrementar la seguridad a comparación de los sitios web que son respaldados por la entidad bancaria.

9. Aunque la interacción del usuario en HCE no sea un requisito estandarizado, es importante su participación para la autorización y permite evitar transacciones fraudulentas.
10. La adopción de HCE en Android es admisible para la emulación de tarjetas bajo el estándar ISO-78416. Lo opuesto ocurre para las tecnologías propietarias y sus aplicaciones, por ejemplo las tarjetas MIFARE y los sistemas de transporte con esta tecnología son irremplazables con soluciones HCE.

6. Bibliografía y Referencias

- [1] Near Field Communication <http://www.androidauthority.com/what-is-nfc-270730/> (Consultado el 20/08/2014)
- [2] What's the difference between RFID and NFC?
<http://electronics.howstuffworks.com/difference-between-rfid-and-nfc1.htm>
(Consultado el 02/09/2014)
- [3] Host Card Emulation (HCE) 101 <http://www.smartcardalliance.org/wp-content/uploads/HCE-101-WP-FINAL-081114-clean.pdf> (Consultado el 02/09/2014)
- [4] Launching Google Wallet
<http://googleblog.blogspot.com.ar/2011/09/launching-google-wallet-on-sprint-and.html> (Consultado el 28/08/2014)
- [5] NFC Enables Secure Mobile Transactions for Google Wallet
<http://www.nxp.com/news/press-releases/2011/05/nfc-enables-secure-mobile-transactions-for-google-wallet.html> (Consultado el 28/08/2014)
- [6] Google Wallet ends support for physical secure elements
<http://www.nfcworld.com/2014/03/17/328326/google-wallet-ends-support-physical-secure-elements/> (Consultado el 01/09/2014)
- [7] Announcing the Android 1.0 SDK, release 1 <http://android-developers.blogspot.com.ar/2008/09/announcing-android-10-sdk-release-1.html> (Consultado 15/02/2015)
- [8] Greg Milette and Adam Stroud, Professional Android Sensor Programming, John Wiley USA, 2012, ISBN: 978-1-118-18348-9 (Consultado 16/02/2015)
- [9] Beginning NFC - Near Field Communication with Arduino, Android, and Phone Gap Tom Igoe, Don Coleman, Brian Jepson - 2013 ISBN 9781449372064 (Consultado 15/03/2015)
- [10] Google I/O 2012 - Up Close and Personal: NFC and Android Beam
<https://www.youtube.com/watch?v=HkzPc8ZvCco> (Consultado 20/03/2015)
- [11] Secure Element Deployment and Host Card Emulation
<http://www.simalliance.org/en?t=/documentManager/sfdoc.file.supply&fileID=1398960008176> (Consultado el 28/08/2014)
- [12] GSMA calls for mass market NFC handsets by mid-2009
<http://www.nfcworld.com/2008/11/19/3235/gsma-calls-for-mass-market-nfc-handsets-by-mid-2009/> (Consultado el 20/08/2014)
- [13] GSMA, EMVCo and Global Platform to develop cross-industry NFC certification
<http://www.nfcworld.com/2009/12/16/32483/gsma-emvco-and-globalplatform-to-develop-cross-industry-nfc-certification-process/>
(Consultado el 30/08/2014)

- [14] Make your phone your wallet
<http://newsroom.mastercard.com/press-releases/google-citi-mastercard-first-data-and-sprint-team-up-to-make-your-phone-your-wallet/> (Consultado el 28/08/2014)
- [15] Accessing the embedded secure element in Android 4.x
<http://nelenkov.blogspot.com.ar/2012/08/accessing-embedded-secure-element-in.html> (Consultado el 25/02/2014)
- [16] SIMalliance Open Mobile API boosts Smartphone Application Security
<http://www.simalliance.org/en?t=/documentManager/sfdoc.file.supply&fileID=1302080109425> (Consultado el 25/02/2014)
- [17] Using the SIM card as secure element
<http://nelenkov.blogspot.com.ar/2013/09/using-sim-card-as-secure-element.html> (Consultado el 08/09/2014)
- [18] Implementation of NFC on an ARM Cortex-M3 Based Platform
<http://www.diva-portal.org/smash/get/diva2:460440/fulltext01.pdf> (Consultado el 25/02/2014)
- [19] Trusted Service Manager: The Key to Accelerating Mobile Commerce
http://www.firstdata.com/downloads/thought-leadership/fd_mobileism_whitepaper.pdf (Consultado el 21/03/2015)
- [20] The NFC Revolution – The role of the Trusted Service Manager
<http://www.bellid.com/resources/downloads/nfc-revolution/> (Consultado el 20/01/2014)
- [21] Global Platform TEE Guide
<https://www.globalplatform.org/mediaguidetee.asp> (Consultado el 25/04/2015)
- [22] Trusted Execution Environments on Mobile Devices
<http://www.sigsac.org/wisec/WiSec2014/sites/default/files/wisec-tutorial2.pdf> (Consultado el 25/04/2015)
- [23] Seek for Android Project
<https://code.google.com/p/seek-for-android/> (Consultado el 19/06/2015)
- [24] Alattar, M.; Achemlal, M., "Host-Based Card Emulation: Development, Security, and Ecosystem Impact Analysis," *High Performance Computing and Communications, 2014*, pp.506,509, 20-22 Aug. 2014 (Consultado el 20/06/2015)
- [25] Host-based Card Emulation
<https://developer.android.com/guide/topics/connectivity/nfc/hce.html> (Consultado el 24/11/2014)

- [26] GSMA MIFARE4Mobile Implementation Guidelines - April 2014
<http://www.gsma.com/digitalcommerce/wp-content/uploads/2015/04/GSMAM4MIG-15042015.pdf> (Consultado el 10/05/2015)
- [27] Payment Tokenization Specification - Technical Framework
<https://www.emvco.com/specifications.aspx?id=263> (Consultado el 17/05/2015)
- [28] Apple Pay Vs. Google Wallet : The Secure Element
<http://www.gmarwaha.com/blog/2014/10/02/apple-pay-vs-google-wallet-the-secure-element/> (Consultado el 17/05/2015)
- [29] Urien, P., "Cloud of secure elements: An infrastructure for the trust of mobile NFC services," *Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2014 IEEE Oct. 2014 (Consultado el 20/05/2015)
- [30] SimplyTapp <https://www.simplytapp.com/> (Consultado el 12/07/2015)
- [31] Inversión de Verizon Ventures en SimplyTapp
http://www.theregister.co.uk/2015/06/08/verizon_splits_with_carrier_led_bonk_tech_and_invests_in_simplytapp/ (Consultado el 20/07/2015)
- [32] Inversiones en SimplyTapp
<http://www.pymnts.com/news/2015/simplytapp-receives-mozido-investment-of-2-5m/> (Consultado el 15/02/2015)
- [33] Repositorio de Github del proyecto HCE <https://github.com/grundid/host-card-emulation-sample> (Consultado el 12/12/2014)
- [34] Kit de desarrollo de SimplyTapp <http://wiki.simplytapp.com/software-dev-kits> (Consultado el 03/07/2015)
- [35] The OAuth 2.0 Authorization Protocol <http://oauth.net/> (Consultado el 03/07/2015)
- [36] Lector NFC ACR-122U <http://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader> (Consultado el 12/01/2014)
- [37] Google Cloud Messaging and HCE
<http://blog.simplytapp.com/2014/04/host-card-emulation-series-google-cloud.html> (Consultado el 15/06/2015)
- [38] The Host Card Emulation in Payments: Options for Financial Institutions
http://www.mahindracomviva.com/wp-content/uploads/2015/02/HCE_Report-MobeyForum_MahindraComviva.pdf (Consultado el 17/05/2015)

7. Bibliografía general

- Technology and Security Considerations for Mobile Contactless Payments at the Point-of-Sale in the U.S.
<http://www.bostonfed.org/bankinfo/payment-strategies/publications/2013/summary-of-mpiw-meeting-june-2013.pdf>
- Routing secure element payment requests to an alternate application
<http://www.google.com/patents/US8807440>
- Blog of a security professional
<https://randomoracle.wordpress.com/>
- Utilización de HCE
http://noticias.lainformacion.com/economia-negocios-y-finanzas/servicios-bancarios/mastercard-utilizara-host-card-emulation-hce-para-pagos-con-moviles-nfc_CKaA9pFXzjQ79Dv4rNXqD1/
- Empresas que adoptan Host Card Emulation (HCE)
<http://contactlessintelligence.com/2014/02/20/mastercard-visa-adopt-host-card-emulation-hce-for-nfc-based-mobile-payments/>
- Blog de SimplyTapp
<http://blog.simplytapp.com/2013/12/host-card-emulation-series-user.html>
- Referencias y glosario de términos
<http://www3.sympatico.ca/bdreid/professional/glossary.html>