UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

ANÁLISIS DE SEGURIDAD DE LA TARJETA BIP! CHILENA COMO MEDIO DE PAGO

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

BORIS AMARO YAZIM ROMERO QUEZADA

PROFESOR GUÍA:
ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
JAVIER BUSTOS JIMÉNEZ
PATRICIO INOSTROZA FAJARDIN
JAIME NAVÓN COHEN

SANTIAGO DE CHILE
2016

Resumen

ANÁLISIS DE SEGURIDAD DE LA TARJETA BIP! CHILENA COMO MEDIO DE PAGO

A fines del año 2007 se descubrieron vulnerabilidades en el cifrado de las tarjetas modelo MIFARE Classic, populares en el mundo por su masivo uso en sistemas de transporte público. En Chile son usadas como medio de pago en el *Transantiago*, sistema de transporte público iniciado el 2007, bajo el nombre de *tarjetas bip!*. A raíz de estas vulnerabilidades, las *tarjetas bip!* fueron blanco de masivos ataques el año 2014.

El objetivo de este trabajo es analizar la seguridad de la utilización de la *tarjeta bip!* como sistema de pago, y las consecuencias que las vulnerabilidades de la tarjeta pueden llegar a tener sobre la población chilena. Para esto, este trabajo mide no sólo la factibilidad técnica de varios ataques (tanto conocidos como nuevos) sino también el eventual impacto de los mismos. Partiendo de una revisión cuidadosa de los estándares, especificaciones y diversos ataques de sistemas similares, el análisis utiliza diversas técnicas, destacando entre ellas la ingeniería reversa efectuada sobre el sistema y las distintas *tarjetas bip!* utilizadas, la ingeniería reversa de aplicaciones móviles asociadas al sistema, y la experimentación directa sobre el sistema distribuido real y en funcionamiento del *Transantiago*, utilizando sólo información públicamente disponible (no información privilegiada).

Fruto de este trabajo, en esta tesis se presentan y discuten los resultados obtenidos del análisis mencionado anteriormente, incluidos los éxitos y fracasos cosechados tras intentar vulnerar las medidas de seguridad encontradas. También se analizan las posibles mitigaciones para dichos ataques y se presentan propuestas de mejora. Entre sus resultados más relevantes, destacan varios ataques exitosos que permiten, por ejemplo, inhabilitar tarjetas de otros usuarios mediante el contacto, obtener información personal del dueño de una tarjeta, e incluso obtener beneficios económicos directos. Ejemplo de estos últimos están los ataques donde se simula un transbordo en vez de un viaje nuevo, se modifica el saldo almacenado en una tarjeta o se vuelve a estados previos de la misma utilizando información almacenada previamente en la tarjeta. Junto con esto, el trabajo también clarifica la distribución de la información almacenada en cada tarjeta y su función en el contexto del sistema, aprendida gracias a los distintos experimentos realizados.

Finalmente, desde una perspectiva más general, este trabajo entrega un estudio comprensivo de los efectos del uso de una tecnología vulnerable en la operación de un sistema de pago asociado a un sistema de transporte masivo como el Transantiago.

A las dos grandes mujeres que ayudaron a forjarme tal cual soy.
Mamá, abuela, las amo.

Agradecimientos

En este largo y tedioso proceso, no tengo más que agradecer a un montón de personas que me han ayudado, de distintas maneras, a sortearlo finalmente de forma exitosa.

Primero, agradecer a mi familia, siempre presente y constante, que mantuvieron la preocupación desde los primeros pasos en mi vida académica, de que mi formación fuera la mejor posible, y que por sobre todas las cosas, me motivaron a buscar mi felicidad. Los amo.

A mi pareja, que me acompañó incondicionalmente muchas de las tardes de trabajo y me apoyó cada vez que necesitaba su cercanía, aun cuando la desesperación de fracasos temporales me volvía insoportable. Bonita, eres la mejor.

A todos mis amigos, viejos y nuevos, que alguna vez se preocuparon por saber como iba con estos asuntos y que con atención, escuchaban mi descripción del trabajo a realizar, o incluso los distintos problemas a los que me estaba enfrentando. Con mi mayor aprecio, agradecer a esos que además discutieron conmigo y opinaron para mejorar mi trabajo. Por su tiempo y dedicación, muchas gracias.

Al profesor guía, los profesores correctores, las secretarias del DCC, y todas las personas que cooperaron en que este documento viera la luz, y que este finalmente proceso se completara.

A la gente del *Transantiago* que se dará el tiempo de revisar lo que hizo este joven en su Tesis de Magister. Espero sinceramente, que les sea útil mi trabajo.

Y por último, a todos aquellos que por alguna razón llegaron a leer esto, y gastaron tiempo de su vida en saber en que trabajé durante varios meses de mi vida.

Para todos ustedes, gracias totales.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto

Al instaurarse *Transantiago* como el sistema de transporte público de Santiago de Chile, la tarjeta *bip!* se posicionó como el emblema del radical cambio experimentado en la locomoción colectiva chilena. El uso de dinero físico en las *micros*[1] dejó de ser necesario, traspasando toda la responsabilidad de las transacciones monetarias a la interacción de los *validadores*, dispositivos lectores de tarjetas *bip!* que efectúan el cobro de pasaje, con la información contenida en estas tarjetas.

Para la implementación de las tarjetas *bip!* se ocuparon como base tarjetas modelo *MIFARE Classic*[2] de tamaño $1K$, a las que se les unió su versión de $4K$ el año 2015 en el lanzamiento de la nueva Tarjeta Nacional Estudiantil [38]. Dichas tarjetas son conocidas en el mundo por su uso, el cual consiste en dar acceso a zonas restringidas o efectuar pagos de pasajes en diversos sistemas de transporte, tal como ocurrió con las tarjetas *Oyster* de Londres o con las *OV-Chipkaart* en el caso de los Países Bajos.

Pero la popularidad de este modelo de tarjeta no se limita únicamente a su uso. Su algoritmo de cifrado, conocido como *CRYPTO1*, fue lanzado al mercado de forma *privativa*, es decir, como propiedad de su empresa productora *NXP*[3] sin entregar mayores detalles de su implementación al público. Debido a su amplio uso, fue objeto de múltiples estudios de seguridad y criptografía, ya que el hecho de ocultar la implementación de un algoritmo pretendiendo alcanzar seguridad (concepto conocido como *seguridad por oscuridad*) atenta contra el *Principio de Kerckhoffs*[4], uno de los pilares teóricos tanto en el área de estudio

[1] *Micros*: nombre popular que se le entrega a los buses del transporte público en Chile.

[2] Información corroborada a través de la página *www.tarjetabip.cl*, respuesta a sugerencia número 10002964, fecha 5 de Septiembre del 2014.

[3] `http://www.nxp.com/`

[4] *Principio de Kerckhoffs*: Un sistema de seguridad no debe depender de que su diseño permanezca en secreto, y este no debe fallar si cae en manos enemigas. [17]

de la criptografía como de la seguridad. Dadas estas características, los estudios sobre estas tarjetas no se hicieron esperar, y junto a estos, los resultados.

En diciembre del año 2007, un grupo de investigadores alemanes presentaron en el congreso número 24 del *Chaos Computer Club* [25] sus avances respecto a ingeniería reversa efectuada sobre el algoritmo criptográfico que utiliza la tarjeta MIFARE Classic, publicándolo en mayor extensión finalmente el año 2008 en [23, 24]. En paralelo y de forma independiente, investigadores de la *Universidad de Roadboud* en los Países Bajos trabajan con la descripción lógica del cifrador y su protocolo de comunicación, presentando sus resultados el mismo año en [13, 8]. Así, con sus estudios respectivos, estos grupos lograron demostrar de manera independiente lo inseguro del cifrado que ocupa el modelo de las tarjetas mencionadas, dejando en particular abierta la posibilidad de ataques sobre estas tarjetas. Los políticos y encargados de los sistemas de transporte que utilizan este tipo de tarjetas aseguran que al caer este acto en la ilegalidad se desincentiva completamente la posibilidad de ataques[5], pero ¿hasta que punto este desincentivo es efectivo?

A modo de ejemplo, el descubrimiento de estos problemas repercutió en los Países Bajos, a tal punto que el *ministerio holandés de transporte, trabajos públicos y administración del agua* pidió a *Trans Link System* o TLS asesoría respecto al tema. A su vez, este solicitó al centro de investigación *TNO*[6] un reporte de la veracidad de los eventuales problemas que podrían traer estas debilidades. Dicho reporte corroboró que el ataque planteado era realista, recomendando en sus conclusiones un futuro cambio del medio de pago, de forma paulatina y tranquila, procurando aplacar los ánimos respecto a la gravedad del asunto [39]. Ante esta respuesta, se solicitó un contrareporte a otro medio independiente, en este caso la *Royal Holloway, University of London*[7], donde finalmente se corroboró la información previamente entregada sobre la validez del ataque, pero además se aseguró que el sistema debía ser cambiado de forma rápida, dado que los ataques planteados no presentaban mayor dificultad de ejecución [5].

Corriendo el año 2013 en Chile, el estudiante universitario Javier Pérez logró consultar saldos y recargar dinero en su tarjeta *bip!* sin problemas desde una aplicación creada para su *smartphone* [2, 32, 35]. Javier Pérez presentó sus resultados en la conferencia de seguridad del mismo país conocida como *8.8 computer security conference*. Ya a mediados del año 2014, un sujeto identificado como Elesier Rascovsky fue detenido por vender tarjetas clonadas y adulteradas, mediante las cuales entregaba un saldo de CLP 25.000 ante un pago de CLP 8.000 [34]. Asimismo, información relevante del sistema podía encontrarse disponible en la web en diversos foros del país, entregando inclusive software y/o instrucciones para realizar ataques, planteándolos como una alternativa para evitar el pago de este servicio [33].

Todos estos parecían ser casos aislados, hasta que el 17 de octubre del 2014, el portal tecnológico *FayerWayer* informó sobre la existencia de una aplicación desarrollada para teléfonos *Android*, la cual permitía, utilizando el sistema de transmisión de datos NFC incorporado en

[5]Fuentes: [32] y [35]

[6]`http://www.tno.nl/`

[7]`https://www.royalholloway.ac.uk/`

ciertos *smartphones*, efectuar una recarga automática de CLP 10.000 a cualquier tarjeta *bip!* [11]. La distribución de esta información y el masivo uso de esta aplicación, trajo una alta incertidumbre y controversia a nivel país, y en lo concreto, más de 30.000 tarjetas comprometidas en menos de dos semanas [20]. Las autoridades encargadas aseguraron la existencia de diversos mecanismos para detectar y bloquear las tarjetas adulteradas, comprometiéndose a aplicar todo el rigor de la ley sobre aquellos que participen de este fraude. Mientras en los foros chilenos, las versiones de la aplicación, cada vez más avanzadas y con múltiples mejoras, siguen distribuyéndose abiertamente entre los interesados. Con ésto, el tema pasó de ser absolutamente ignorado por las autoridades y la población, a posicionarse en un constante estado de alarma. Las críticas y discusiones continúan hasta hoy.

1.2. Objetivos del trabajo

El presente trabajo posee el siguiente objetivo general y los siguientes objetivos específicos:

1.2.1. Objetivo general

Analizar vulnerabilidades de la tarjeta *bip!* como sistema de pago y las consecuencias que estas pueden llegar a tener sobre la población chilena.

1.2.2. Objetivos específicos

1. Analizar los problemas de seguridad presentes en el cifrado utilizado en la tarjeta *bip!*, dejando en evidencia cuáles son sus principales vulnerabilidades.

2. Mostrar las consecuencias directas que las vulnerabilidades analizadas pueden traer sobre la vida de los ciudadanos chilenos con ejemplos prácticos de ataques y sus consecuencias.

3. Determinar la criticidad de las brechas de seguridad estudiadas en términos de privacidad de datos y de disponibilidad del servicio. Para ello se evaluarán ataques tanto de lectura de información de las tarjetas como de escritura, para así orientar la búsqueda de mejoras o plantear un eventual reemplazo al sistema utilizado.

4. Analizar cada una de las mitigaciones planteadas para reparar la efectividad del sistema de pago evaluando su desempeño y posibles puntos débiles. Además, proponer nuevas estrategias tanto para las amenazas existentes, como para las planteadas en este estudio, analizando también su posible desempeño.

5. Entregar estrategias de acción que permitan a implementadores de políticas públicas aprender de este caso, concientizando a la sociedad chilena acerca de que requieren

las tecnologías que ocupa diariamente para transportarse y las implicancias que éstas pueden tener sobre su privacidad.

1.3. Metodología de trabajo

El desarrollo de la presente tesis siguió la siguiente metodología de trabajo. Ésta se divide en etapas, las cuales a su vez se subdividen en tareas.

Primera Etapa: Obtención y análisis de datos

1. Verificar el estado del arte respecto a las documentaciones, debilidades e implementaciones de ataques disponibles para las *MIFARE Classic* 1K y 4K, y para la tarjeta *bip!* en particular.

2. Obtener una amplia gamma de tarjetas *bip!* a disposición, distribuidas entre los cuatro tipos especificados, a fin de poseer una cantidad apropiada de datos sobre los cuales efectuar análisis y poder generar conclusiones. Para esto se usan tarjetas ofrecidas anónimamente para la investigación o compradas con el mismo fin.

3. Probar la eficacia de los ataques mencionados sobre el cifrado *CRYPTO1* en su aplicación particular sobre las tarjetas *bip!*, considerando sus diversos modelos de manera suficientemente representativa.

4. Comprender la información obtenida de las muestras en el punto anterior y concluir características relevantes respecto al cifrado utilizado.

5. Entender la distribución de la información almacenada dentro de los múltiples tipos de tarjetas analizadas y cómo ésta interactúa con el sistema.

Segunda Etapa: Análisis y desarrollo de ataques

6. Analizar la información almacenada en los diversos tipos de tarjetas *bip!*, y qué tan sensible es para el usuario que ésta caiga en entidades maliciosas. Los ataques aquí propuestos serán conocidos como *ataques de lectura*.

7. Analizar la posibilidad de escribir en los datos de la tarjeta y los mecanismos de defensa que posee el sistema ante este tipo de fraudes. Los posibles ataques aquí encontrados serán conocidos como *ataques de escritura*.

8. Evaluar y probar la escalabilidad de estos ataques y las consecuencias que pueden traer eventualmente sus aplicaciones sobre la población.

Tercera Etapa: Análisis y desarrollo de mitigaciones

9. Evaluar las mitigaciones propuestas para los ataques previamente descritos, considerando su posible eficiencia y sus eventuales debilidades.

10. Proponer nuevas mitigaciones para las situaciones problemáticas anteriormente encontradas, evaluando a fondo su aplicación y viabilidad, a fin de presentar un avance respecto a la seguridad del sistema y las opciones de reparar las inseguridades presentadas.

Cuarta Etapa: Conclusiones

11. Concluir respecto a la seguridad general del sistema, dada la información recopilada y generada en el estudio. Entregando además, una opinión respecto a cómo abordar y proceder ante situaciones similares a futuro.

1.4. Descripción del trabajo realizado

Tal como se planteó en los párrafos anteriores, muchas dudas quedan respecto de la utilización de estándares suficientes de seguridad para proteger los datos del usuario y las transacciones mismas que ocurren en el sistema de transporte chileno. Así, dada esta problemática, el presente trabajo se centra en analizar la seguridad del sistema de pago del *Transantiago*, en particular las implicancias del ataque sobre el sistema de cifrado de la tarjeta *bip!*, midiendo tanto la factibilidad de posibles ataques conocidos como la eventual magnitud de los mismos, analizando a su vez posibles reparos o mejoras.

De esta manera, la motivación principal de esta tesis es lograr analizar el uso de una tecnología insegura de forma masiva, considerando las consecuencias de su efectividad y las posibles mitigaciones para los problemas que esto presenta, en un caso de estudio particular e interesante, como lo es el uso de tarjetas *bip!* en el *Transantiago*.

Para concretar todos los objetivos de manera completa, fue necesario realizar un análisis detallado de la forma en que se almacena la información en las tarjetas de los usuarios, diferenciando los diversos tipos disponibles en el mercado y cómo la información se encuentra distribuida en ellas. Así, fue vital contar con un universo amplio de muestras de tarjetas, representativo para todos los tipos disponibles, a fin de entregar una mirada sólida y correcta del sistema en general.

Para evaluar la efectividad de los ataques sobre la tarjeta *MIFARE Classic*, se trabajó con implementaciones conocidas de los ataques planteados que mostraron en el pasado, éxito en su aplicación. Con esto en marcha, se concluye información relevante respecto a los mecanismos de seguridad que se utilizan actualmente en las diversas tarjetas del *Transantiago* y la efectividad real que éstos presentan como vector de ataques masivos.

Desde aquí, existen múltiples preguntas relevantes respecto al funcionamiento del sistema,

debido principalmente a la baja información disponible de manera pública de esta implementación. Estas preguntas se inspiran en especulaciones de diversas fuentes, generadas por ejemplo, a partir de un análisis simple de la arquitectura misma que posee este sistema. De estos cuestionamientos, fue importante dilucidar, por ejemplo, qué información actualmente se almacena en las distintas tarjetas disponibles, siendo especialmente relevante la presencia de información sensible de los usuarios en los mismos dispositivos. Respecto al cifrado, fue importante analizar si se utiliza íntegramente el sistema provisto por *MIFARE* o existen modificaciones; y si ataques conocidos se aplican así sobre el sistema. Fue necesario investigar además, especificaciones generales tales como si las claves utilizadas en el cifrado son las mismas para todo tipo de tarjeta o si existen diversas combinaciones posibles dependiendo de alguna característica extra.

Por lo tanto, es vital para este trabajo dilucidar estos cuestionamientos y entregar claridad respecto a los reales niveles de seguridad existentes. De esta forma, este trabajo busca responder estas preguntas a cabalidad, evidenciando en caso de ser necesario, las falencias ignoradas, quizás tanto por los encargados del sistema como por la ciudadanía chilena en general.

Por último, y tal como se mencionó en el contexto histórico, las recientes revelaciones de ataques utilizados de forma masiva y sus correspondientes propuestas de mitigación, constituyen objetos importantes de análisis considerados en este estudio. Éstos a su vez, son acompañados de una búsqueda exhaustiva de otras posibles formas de ataque y de múltiples mecanismos de mitigación para los diversos escenarios planteados, en los que se analiza a fondo su efectividad y viabilidad. Así, con estos últimos objetivos, se le entrega aún mayor importancia, urgencia y responsabilidad al desarrollo de esta tesis.

Capítulo 2

Tarjeta Mifare Classic

2.1. Introducción

Para la implementación de la tarjeta *bip!* chilena, se utilizó como base *hardware* especializado en sistemas de transporte público y acceso restingido. Así, la tarjeta *MIFARE Classic* entró al juego, siendo la elegida para esta función[1], principalmente, por sus similares aplicaciones en el extranjero. Entre diversos casos, podemos citar a la tarjeta *Oyster* [30] en Londres; la tarjeta *Charlie* [3] en Boston; la *SmartRider* [40] en Australia; la *EasyCard* [7] en Taiwan; la tarjeta *Octopus* [29] en Hong Kong; y, la *OV-chipkaart* [31] en los Países Bajos.

La tarjeta *MIFARE Classic* fue introducida al mercado en 1994 por *Phillips*, actual empresa *Semiconductores NXP* [2], y en su propia definición en [26] señala que "*es una tarjeta inteligente que funciona sin contacto, en un rango de frecuencia de 13.56 MHZ con capacidad de lectura y escritura bajo el protocolo ISO 14443 [1]*". Está disponible en dos versiones dependiendo del tamaño de su memoria; de 1 kB y de 4 kB. En este estudio, nos centraremos en las de 1 kB, dado que son las que el sistema de transporte chileno utiliza, exceptuando el caso de las TNE instauradas desde el año 2015, ya que utilizan la versión de 4 kB.

En este capítulo, se analizan las principales características de esta tarjeta, la forma en la que trabaja, sus principales vulnerabilidades y los más famosos ataques planteados y/o probados para la misma. Así, es posible relacionar esta información con nuestro caso de estudio y determinar, a futuro, cómo influye el hecho de que su base sea insegura en el servicio final.

[1] Información corroborada a través de la página *www.tarjetabip.cl*, respuesta a sugerencia número 10002964, fecha 5 de Septiembre del 2014.

[2] `http://www.nxp.com/`

2.2. Descripción General

2.2.1. Capacidad computacional

La tecnología *RFID* (identificación por radiofrecuencia) y las *tarjetas inteligentes que funcionan sin contacto* han encontrado cabida en los mercados del mundo de forma amplia en las últimas décadas, reemplazando recursos como códigos de barras, tarjetas de cintas magnéticas y hasta boletos de papel en diversas aplicaciones. Las características que provee su interacción de modo inalámbrico ofrece múltiples ventajas en los campos de la comodidad y rendimiento, transformándoles en productos deseables en el desarrollo de diversos rubros.

Las *tarjetas inteligentes que funcionan sin contacto*, entre las cuales destaca la *MIFARE Classic*, se diferencian de las tarjetas *RFID*, dado que poseen pequeñas capacidades de computación. Éstas capacidades se ocupan en general, para el uso de algún tipo de criptografía de llave simétrica simple, resultando aptas por ejemplo, para aplicaciones en donde se requiere control de acceso o protección de cierta información. En el caso de la *MIFARE Classic*, se utilizan estos recursos en la ejecución del cifrador de flujo *CRYPTO1*, gracias al cual se establece una comunicación cifrada (y no en texto plano) entre el lector y la tarjeta misma en sus interacciones. Para mayor información respecto a las diferencias entre *RFID* y *tarjetas inteligentes que funcionan sin contacto*, y las medidas de seguridad de la comunicación de *MIFARE Classic* dirigirse a [18].

Tal como se describió en la introducción, el cifrador *CRYPTO1* fue producido de forma *privativa*, es decir, pertenecía a la empresa *NXP* y se mantuvo en secreto, ocultando su lógica al análisis a todos los que quisieran conocerla. En consecuencia, esta práctica entró en conflicto con una de las bases de la criptografía conocida como el *Principio de Kerckhoffs*[3], el cual requiere, en resumen, que la seguridad sólo recaiga en ocultar alguna clave con la que trabaja el sistema, y no parte del funcionamiento del mismo. Este hecho motivó a varios grupos universitarios a estudiar sus características y procurar efectuar algún tipo de ingeniería reversa, a fin de entender cómo este cifrador funcionaba y cáles características de seguridad poseía. Lamentablemente, posterior a estos análisis y entendido el comportamiento del cifrador, los resultados no fueron positivos, y tal como muchos supusieron en su momento (debido a la dificultad de producir sistemas criptográficos seguros de la nada), el cifrador poseía múltiples vulnerabilidades. Trabajos detallados de este proceso de estudio son [13, 10, 24]; entre otros.

Mayores características del cifrado, y cómo interactúa en la comunicación tarjeta-lector, se encuentra en la próxima sección.

[3] *Principio de Kerckhoffs*: Un sistema de seguridad no debe depender de que su diseño permanezca en secreto, y este no debe fallar si cae en manos enemigas. [17]

2.2.2. Almacenamiento

La tarjeta *MIFARE Classic* es esencialmente un chip de memoria EEPROM con características de comunicación inalámbrica que pretenden ser seguras. Posee una memoria estructurada; en el caso de la tarjetas de 1 kB, está dividida en 16 *sectores*, los cuales a su vez, se subdividen en 4 *bloques* de 16 bytes cada uno*(figura 2.1)*. Para el caso de la versión de 4 kB, la distribución no es homogénea, ya que tiene 40 sectores: los primeros 32 sectores de 4 bloques cada uno, y los últimos 8 sectores de 16 bloques cada uno. Los bloques tienen a su vez el mismo tamaño que en la versión de 1 kB, es decir, 16 bytes cada uno. En general, a lo largo de este documento, nos centraremos en el análisis de las tarjetas de 1 kB, dado que las únicas tarjetas de 4 kB que se ocupan en el sistema, es decir, las TNE emitidas a partir del año 2015 [38], emulan el comportamiento de las tarjetas normales de 1 kB (se detalla más de esto en la sección 4.4.2).

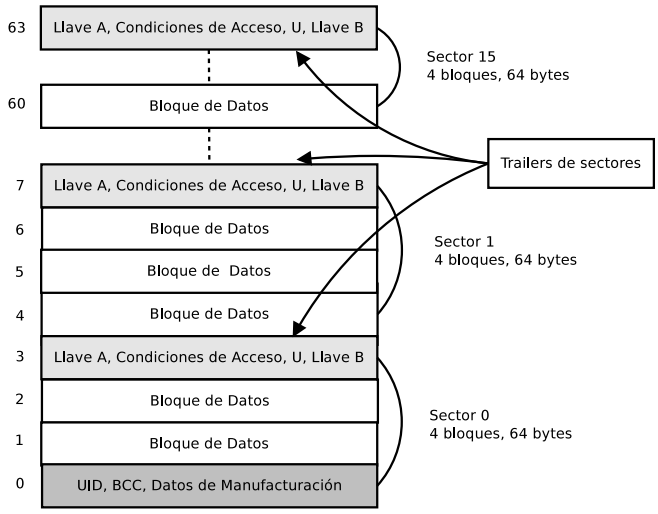

Figura 2.1: Diagrama de la memoria de una tarjeta MIFARE Classic de 1kB.

En todas las tarjetas *MIFARE Classic*, el bloque 0 del sector 0 *(figura 2.2)* contiene datos especiales, los cuales poseen exclusivamente permisos de lectura establecidos por *hardware*. Los primeros 4 bytes son el *identificador único de la tarjeta*, conocido como UID por sus siglas en inglés. El siguiente byte, conocido como BCC, equivale a la aplicación sucesiva de *XOR* sobre todos los bytes del UID. Por último, los demás 11 bytes contienen datos de la empresa manufacturera de la tarjeta [10].

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UID				BCC	Datos del fabricante										

Figura 2.2: Representación de los bytes del bloque 0 del sector 0

Por otro lado, cada sector tiene en su último bloque, conocido como el *trailer* del sector, dos llaves secretas de 48 bits (conocidas como *llave A* y *llave B*), más 24 bits con las *condiciones*

de acceso correspondientes al mismo sector *(figura 2.3)*. Estas llaves y condiciones de acceso ocupan un rol importante en el proceso de cifrado de la comunicación *tarjeta - lector*. Por ejemplo, para efectuar operaciones sobre algún bloque es necesario primero autentificarse con el sector al cual pertenece dicho bloque. Las condiciones de acceso en el sector, que se analizarán en detalle en futuros capítulos, determinan si la llave A o la llave B debe ser usada en cada una de las operaciones a realizar. Además, al lado de los bytes que determinan las condiciones de acceso, hay un byte U sobrante el cual no tiene propósito definido. Un ejemplo de representación de un sector 0 completo, puede encontrarse en el anexo 7.1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Llave A							Bits Acceso			U	Llave B				

Figura 2.3: Representación trailer de un sector cualquiera

Los bloques restantes pueden dividirse en dos tipos dependiendo de la configuración explicitada en los bits con las condiciones de acceso de su sector: *bloques de datos* y *bloques de valor*. Un bloque se denomina *bloque de datos* cuando sus 16 bytes están disponibles para ser utilizados para almacenar datos, sin mayor limitación.

Por el otro lado, cuando se usa un bloque como *bloque de valor* este sólo tiene 4 bytes disponibles para guardar un valor numérico con signo, sobre el cual es posible realizar operaciones como *incremento, decremento, restaurar* y *transferir*, además de las básicas operaciones de *lectura* y *escritura*. El espacio restante se utiliza para respaldar este valor una vez más y almacenar también su inverso, en el cual cada bit b' equivale a $b' = 1 \oplus b$, siendo b el valor del bit originalmente. Los últimos cuatro bytes, se utilizan para almacenar una dirección del bloque de tamaño 1 byte, valor también duplicado junto a su inverso dos veces *(figura 2.4)*. Para almacenar los valores negativos, se utiliza el estandar *two's complement*[4].

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
valor				$\overline{\text{valor}}$				valor				dir	$\overline{\text{dir}}$	dir	$\overline{\text{dir}}$

Figura 2.4: Representación de los bytes de un bloque de valor

Cabe destacar además, que toda la información respaldada en bloques se guarda siguiendo la representación de palabras *little-endian*[5], es decir, almacenando desde el byte menos significativo hacia el más significativo de los valores, en la dirección más pequeña de la memoria hacia la más grande.

[4] `http://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html`
[5] `http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Data/endian.html`

Para obtener mayor información de las especificaciones de las *MIFARE Classic* tanto de 1 kB como de 4 kB, pueden consultarse las hojas de características de productos [27] y [28] respectivamente y [37] para ambas.

2.3. Comunicación con dispositivos

La comunicación de las tarjetas *MIFARE Classic* con los diferentes dispositivos habilitados para su interacción, respeta el protocolo establecido en la norma ISO/IEC 14443A [1], implementando sus partes 1 a la 3 de forma completa, variando con una implementación propia en la parte 4, con lo que diverge del estándar [10, 16]. Esta parte 4 del protocolo ISO mencionado es la que estipula cómo los mensajes son enviados entre los diversos dispositivos participantes, y es donde entra en juego en la versión de NXP su cifrador propietario conocido como CRYPTO1.

2.3.1. Cifrador CRYPTO1

CRYPTO1 es un *cifrador de flujo*, esto es, un cifrador de clave simétrica, donde dígitos de texto plano se combinan con una corriente de dígitos pseudoaleatorio, conocida como *keystream*. En este cifrado, cada dígito de texto plano es operado uno a la vez con el dígito correspondiente del *keystream*, produciendo un nuevo dígito de la corriente de texto cifrado.

El algoritmo de cifrado de CRYPTO1 es un *registro de desplazamiento con retroalimentación lineal*[6] (conocidos como LFSR por su sigla en inglés) de 48 bits. Su polinomio generador del registro es:

$$P(x) = x^{48}+x^{43}+x^{39}+x^{38}+x^{36}+x^{34}+x^{33}+x^{31}+x^{29}+x^{24}+x^{23}+x^{21}+x^{19}+x^{13}+x^{9}+x^{7}+x^{6}+x^{5}+1 \tag{2.1}$$

El trabajo de ingeniería reversa para entender su funcionamiento fue principalmente realizado por Karsten Nohl et al. [24]. Este no sólo sirvió de inspiración para múltiples trabajos a futuro sino justificó y dió sustento para las dudas que ya se manejaban respecto a este cifrador. Para mayor información respecto a CRYPTO1 ir a [13, 12, 15, 24].

2.3.2. Protocolos de comunicación

Para comenzar las interacciones tarjeta-lector primero necesitamos poder alimentar energéticamente nuestra tarjeta entrando al campo magnético del lector. Una vez energizada comienza inmediatamente el proceso conocido como *protocolo anticolisión*, seguido por el *protocolo de autenticación* para luego realizar la *comunicación encriptada*.

[6] `http://www.yikes.com/~ptolemy/lfsr_web/`

El *protocolo anticolisión* tiene como idea el poder realizar transacciones de modo seguro con una única tarjeta, a pesar de que existieran más de ellas en el campo de lectura. Este comienza con el envío del UID por parte de la tarjeta al lector, quien luego selecciona a la tarjeta respondiendo con el mismo UID, estableciendo así la comunicación.

El *protocolo de autenticación* tiene cabida al intentar realizarse alguna operación sobre los datos de la tarjeta. Las operaciones modulares, como escritura o lectura, se efectúan sobre bloques completos, por lo cual el lector envía una solicitud a la tarjeta para acceder al sector al cual pertenece el bloque en cuestión. En este momento la tarjeta responde con un *desafío aleatorio* de 32 bits al lector. Desde este paso, toda la comunicación efectuada es encriptada, esto quiere decir, que la información en cuestión es enviada luego de ser operada, mediante XOR ($\oplus$) con el *keystream* producido por CRYPTO1. Así, el lector responderá cifrando su propio desafío de 32 bits, más una respuesta al desafío enviado por la tarjeta de 32 bits y también cifrada. Si todo sigue bien, la tarjeta responderá el desafío del lector cifrando esta respuesta y enviándola, completándose así la autenticación.

Hecho esto, la *comunicación encriptada* puede realizarse sin problemas con el sector autenticado con lo que pueden efectuarse las operaciones pertinentes, respetando siempre los bits con las condiciones de acceso del sector.

Para mayor información de estos protocolos, revisar [10, 12, 13, 15] y el anexo 7.1.

2.4. Vulnerabilidades

A pesar de las buenas intenciones del protocolo en cuanto a su seguridad, la MIFARE Classic tiene múltiples vulnerabilidades, que principalmente recaen en la implementación del protocolo. A continuación se describen algunas de las más importantes.

2.4.1. Tamaño de las claves

Sin conocer mucho de la tarjeta MIFARE Classic, el simple hecho de que posee claves de 48 bits de largo para resguardar sus datos inmediatamente levanta dudas de su real seguridad. Basta compararla con la primitiva criptográfica DES[7], desarrollada en los años 70 y que utiliza claves de 56 bits, es decir, con un costo 2^8 veces mayor frente a un ataque de fuerza bruta. Pese a eso, ya a principios del año 1999 fue quebrada por ataques de fuerza bruta que demoraban aproximadamente unas 22 horas. Tal y como se menciona en [18], la elección de tamaños simplemente fue una mala decisión de diseño, justificada probablemente en el hecho de que el uso de estas tarjetas nunca se pensó para sistemas de alta seguridad.

[7] `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`

2.4.2. Bits de paridad

La norma ISO/IEC 14443A [1] establece que por cada byte enviado en la comunicación tarjeta/lector debe enviarse un bit de paridad, siguiendo el modelo de *paridad impar*[8], a modo de verificación de integridad. Este bit debería ser calculado sobre el texto cifrado, pero la implementación de la tarjeta MIFARE Classic computa este bit sobre el texto plano. El problema de esto es que en cada transferencia este bit de paridad está entregando información del texto enviado innecesariamente, la cual puede ser utilizada posteriormente para ataques tal como se mostrará en la próxima sección. A este problema, se le suma el hecho de que el bit del *keystream* usado para encriptar el bit de paridad de un byte dado, es reusado para encriptar el siguiente bit de texto plano, rompiéndose nuevamente el esquema de la norma ISO mencionada. Para mayor información de estas vulnerabilidades revisar [12, 13].

2.4.3. Mensajes de error

En el proceso de autenticación entre una tarjeta y un lector, luego de que el lector recibe el desafío aleatorio de la tarjeta envía la respuesta a éste designada como r_l, junto a su propio desafío denominado d_l, además del byte de paridad correspondiente a todo este mensaje. Si al menos uno de estos bits en el byte de paridad no corresponde, la tarjeta no entrega respuesta. No obstante, si la paridad está correcta pero la respuesta r_l no, la tarjeta responde con un código de error $0x5$ encriptado, a pesar de que el proceso no haya sido completado y no se pueda asumir que el lector puede desencriptar. Esto, que puede parecer simple, fue descubierto y explotado en [10, 12] de la forma descrita en la próxima sección.

2.4.4. Generador de Valores pseudo-aleatorios

Los números aleatorios utilizados en la tarjeta en cuestión son generados por un LFSR de un máximo de 16 bits de la forma:

$$P(x) = x^{16} + x^{14} + x^{13} + x^{11} + 1 \tag{2.2}$$

Este LFSR posee un estado inicial constante completamente innecesario, que vuelve a establecerse cada vez que la tarjeta inicia una operación, perdiendo la aleatoriedad de los usos previos. La generación de aleatoriedad sólo depende del número de ciclos de reloj que ocurren entre que la tarjeta es energizada y el momento en que el valor es solicitado. Además, el LFSR tiene un periodo de 65535 ciclos, y dado que cambia cada $9{,}44\mu s$, un ciclo dura $618ms$. Conocida esta información y utilizando los implementos necesarios, la aleatoriedad de los valores generados prácticamente deja de existir. Para más detalles, revisar [24, 12].

[8] `https://www.techopedia.com/definition/1801/odd-parity`

2.4.5. Autenticaciones de sectores anidadas

La primera autenticación desde que una tarjeta se conecta con otro dispositivo es distinta a las que le siguen. Esto se debe a que los primeros pasos de la primera autenticación no se realizan encriptados, a diferencia de los posteriores, donde toda la comunicación ya es encriptada. Luego del comando de autenticación, el estado inicial del cifrador cambia a la clave del nuevo sector explorar, y el protocolo de autenticación vuelve a comenzar. Esto permite, como describiremos en la próxima sección, obtener todas las claves de una tarjeta a partir de una sola clave. Para mayor información de esta vulnerabilidad, dirigirse a [12].

2.5. Ataques Importantes

Los ataques sobre la tarjeta MIFARE Classic evolucionaron durante el año 2008 rápidamente a sofisticados mecanismos para obtener toda la información vinculada a una tarjeta. A continuación, se describen importantes hitos en esta cadena de sucesos, que permitieron finalmente diseñar e implementar los distintos ataques hasta ahora efectuados en el sistema de transporte chileno, junto a los descritos en el presente trabajo. Una buena compilación y descripción de varios ataques se encuentra en [15].

2.5.1. Fuerza bruta *online*

Sin mayor conocimiento de criptografía ni de las vulnerabilidades previamente descritas, un ataque *online* de fuerza bruta podría ser efectuado intentando con todas las posibles combinaciones de claves, de largo de 48 bits, para cierto sector. Estas son 2^{48} opciones, las cuales tomarían, dado los $5ms$ de demora de cada intento, unos 44627 años de espera. Claramente esta no es una buena opción, pero es el primer análisis pertinente dada la debilidad presentada en 2.4.1.

2.5.2. Recuperación de *keystream*

En [10] se utilizan las características del generador de números pseudo aleatorios, descritas en 2.4.4, para recuperar parte del *keystream* utilizado en alguna transacción genuina entre una tarjeta y un lector. Para analizar estos datos, la transacción debe ser escuchada con hardware especializado (como Proxmark[9]). Con esto, incluso sin saber alguna clave, se pueden leer y modificar bloques de la tarjeta conociendo solamente el texto plano que es necesario enviar.

[9] `http://www.proxmark.org/`

2.5.3. Estados del cifrador

Usando hardware especializado para controlar variables del proceso de comunicación (como los mensajes enviados, tiempo de conexión, etc.) Garcia et al. en [13] logran determinar que el estado inicial del LFSR presente en el cifrador es la misma clave de 48 bits a utilizar. Así, entendiendo además como varían los estados internos del LFSR y cómo trabaja su función de flujo f, desarrollan una técnica para volver al estado inicial del generador, es decir, la clave, usando partes del *keystream* una vez recuperado con ataques como 2.5.2.

2.5.4. Fuerza bruta *offline*

Explotando las vulnerabilidades 2.4.3 y 2.4.2, utilizando hardware especializado, es posible, enviando mensajes al azar, utilizar los mensajes de error de la tarjeta para obtener información del estado interno del cifrador, y de esta forma, aprovechando el ataque 2.5.3, recuperar la clave en cuestión. El ataque se plantea en [12], y se asegura que usando hardware apto para encontrar la clave que cumple con las condiciones, podría reducirse un ataque de fuerza bruta a 36 minutos. Esto es clara mejora del ataque 2.5.1.

2.5.5. Ataque *darkside*

El nombre de este ataque se debe principalmente a la publicación [9] en la cual fue presentado. Plantea una evolución a una serie de ataques que requerían pre computación, un alto número de consultas a una tarjeta, e incluso tablas de datos para llegar a buen puerto. Este ataque en comparación, no requiere pre computación y realiza entre 1500 y 4000 consultas. Se basa en un caso particular de la vulnerabilidad 2.4.3, y realizando álgebra con las respuestas obtenidas y varios intentos sobre la tarjeta, finalmente logra entregar una clave de la tarjeta. Este es uno de los ataques más populares conocidos a la tarjeta, y de los cuales se utilizó su implementación en esta tesis. Para mayor detalle, revisar el artículo en cuestión.

2.5.6. Ataque de las autenticaciones anidadas

Nuevamente utilizando dispositivos especializados para mantener constante los tiempos de conexión y distintos parámetros de forma conveniente, en este ataque se explota la vulnerabilidad 2.4.5 para lograr obtener, a partir de una clave de la tarjeta, todas las demás. La ventaja ocurre ya que la autenticación desde un sector a otro, nos entrega una ventaja de 32 bits del *keystream*, quedando simplemente 2^{16} posibles claves para el nuevo sector. Repitiendo este proceso unas cuantas veces (dos o tres en la práctica) la intersección de los candidatos se reduce drásticamente a 1 o 2 opciones, obteniéndose así la clave del nuevo sector sin mayor dificultad. Esto se puede extender a todos los demás sectores, y finalmente así, obtener todas las claves de la tarjeta analizada. El ataque es planteado en detalle en [12] y se utilizará una implementación del mismo en este trabajo.

Capítulo 3

Sistema de Transporte Chileno

3.1. Introducción

El *Transantiago*[1], nombre mediante el cual se denomina al sistema de transporte de Santiago de Chile, ve la luz operativamente el año 2007 en el mes de febrero, existiendo un plazo hasta el año 2011 para su implementación completa en el área de buses, recorridos e infraestructura.

Entre sus principales apuestas, planteó un mapeo completamente nuevo de los recorridos que se desplazarían por la capital, centrándose en un cambio de paradigma de viaje. La nueva idea era evitar la existencia de un único viaje entre un punto y otro de la ciudad (lo cual provocaba viajes extensos y buses repletos, al ser opciones casi únicas de transporte), reemplazándolo por buses *troncales* distribuidos en las diferentes grandes rutas que comunican la ciudad, desplazándose el usuario hacia y desde ellas mediante buses *alimentadores*, los que recorren en general no más que un par de comunas. Así, se esperaba reducir el tiempo del viaje final a costa de agregar *transbordos*, es decir, cambios de un recorrido a otro como parte del viaje.

Además, el cambio planteaba avances tecnológicos importantes, tales como la eliminación de toda transacción de dinero físico dentro de un bus para hacer efectivo el pago de un pasaje, la unificación de las opciones de viaje bajo un único sistema de pago tanto para *micros*[2] como para el metro, la implementación de computadores en las mismas máquinas para un mejor monitoreo del bus por parte de los ahora *operadores*, hasta el seguimiento mismo de cada maquina vía GPS, dándole grandes oportunidades al sistema de relacionarse de mejor forma con sus usuarios, esperando poder ofrecer así, mejoras en la calidad del servicio mismo.

Indiferente al éxito o fracaso que finalmente hayan podido tener estas medidas en el ámbito político, los cambios tecnológicos se concretaron, y con ellos, nuevos problemas entraron

[1] `http://www.transantiago.cl/`

[2] *Micros*: Nombre popular que reciben los buses del sistema de transporte público chileno.

a la escena. Por el lado del primer cambio tecnológico importante mencionado, la responsabilidad de llevar a cabo las transacciones monetarias se pasó a las ya detalladas tarjetas MIFARE Classic de 1 kB primeramente, sumándose la versión de 4 kB desde el 2015, para algunos tipos de tarjetas. Dados los problemas ya mencionados de estas tarjetas en 2.5, este estudio logra importancia crítica al analizar los riesgos de ataque en un sistema vigente y en funcionamiento.

La responsabilidad de la puesta en marcha de toda la parte tecnológica del *Transantiago* fue finalmente aceptada por la empresa *Sonda*[3], la cual no ha sido llevada libre de conflictos o controversias [22]. El detalle aprendido de cómo distintas piezas se unen en este sistema y cuál es la lógica con la que operan, se detallará en las siguientes secciones de este capítulo.

3.2. Tipos de Tarjetas

Existen dos grandes grupos de tarjetas para poder realizar viajes en el *Transantiago*. Estos grupos los denominaremos *tarjetas anónimas* y *tarjetas vinculadas a un usuario*, las cuales pueden ser utilizadas bajo distintos contextos de usuario.

3.2.1. Tarjetas anónimas

Las únicas tarjetas anónimas del sistema son las *tarjetas bip! al portador*[4]. Estas están disponibles para cualquier usuario, son transferibles y no tienen mayor vinculación al usuario que las usa (aunque en la práctica, generalmente los usuarios manejan una única tarjeta, creando un vínculo uno a uno). Dado este anonimato en el uso, no pueden ser bloqueadas por robo.

Para su implementación se utilizan las tarjetas MIFARE Classic de 1 kB. El costo de estas tarjetas es de CLP 1.500, con carga mínima de CLP 1.000, y permite realizar hasta dos *transbordos* pagando un único pasaje, en un plazo de dos horas desde efectuado el pago. No gozan de reducción en el precio del pasaje.

3.2.2. Tarjetas vinculadas a un usuario

Tal como su nombre lo indica, las tarjetas de este tipo están vinculadas a un usuario en particular y son entregadas exclusivamente bajo ciertos contextos por entidades específicas. Una tarjeta de este grupo puede ser de tres tipos: *tarjeta personalizada*, *tarjeta bancaria* o *tarjeta nacional estudiantil*.

[3] `http://www.sonda.com/cl/`
[4] `http://tarjetabip.cl/bip-portador.php/`

Figura 3.1: Distintos tipos de tarjetas disponibles en el *Transantiago*.

- **Tarjeta personalizada.**[5] En su uso, una *tarjeta personalizada* es idéntica a la *tarjeta bip! al portador*, cobrando los mismos valores por pasajes y soportando los mismos transbordos para las distintas plataformas del sistema. Su particularidad recae en que al estar vinculada a un usuario desde su creación, en caso de robo o perdida, el propietario puede solicitar un bloqueo de la tarjeta actual y una recuperación del saldo de la tarjeta extraviada en una nueva *tarjeta personalizada*. Estas tarjetas tienen un costo mayor a las *tarjetas bip! al portador*, y sólo pueden obtenerse en las oficinas del servicio al cliente. En su diseño visual posee una fotografía del usuario vinculado, su nombre y su RUN.

- **Tarjeta bancaria.**[6] Ciertas entidades bancarias pueden ofrecer en algunas de sus tarjetas de débito la funcionalidad extra de *tarjeta bip!*, la cual trabaja prácticamente de forma idéntica a una *tarjeta personalizada*. Su saldo puede ser recuperado de igual forma en caso de robo o perdida, y está vinculada al propietario de la cuenta en cuestión. Cabe destacar que los saldos de la tarjeta respecto a la cuenta bancaria son absolutamente independientes, incluso, sus sistemas están generalmente separados de forma física (uno por chip y el otro por banda magnética), por lo cual, sólo comparten el plástico que los contiene. Los bancos capaces de entregar este tipo de tarjeta son *BancoEstado*, *Banco de Chile*, *Banco de Créditos e Inversiones (BCI)* y *Banco Santander*.

- **Tarjeta nacional estudiantil.**[7] La *TNE* es un tipo de *tarjeta bip! personalizada* entregada por la *Junta Nacional de Becas* o *JUNAEB*[8] a todos los estudiantes del país, dada la reducción de tarifa de la que gozan. Este beneficio es personal e intransferible (con lo cual la tarjeta también debería serlo) y es accesible las 24 horas del día a lo largo de todo el año. En su diseño visual, contienen una foto del estudiante, su nombre y RUN, además del grado de estudio que está cursando. Dada su menor tarifa, la carga mínima de este tipo de tarjetas también es menor.

[5] `http://tarjetabip.cl/bip-personalizada.php/`
[6] `http://tarjetabip.cl/bip-bancaria.php/`
[7] `http://tarjetabip.cl/bip-tne-escolar.php/`
[8] `http://www.junaeb.cl/`

3.3. Viajes

3.3.1. Validación

Tal como se describió en un comienzo, uno de los grandes aportes del *Transantiago* fue la unificación del sistema de pago de los dos tipos de transportes públicos más importantes dentro de la capital, el Metro de Santiago[9] y los Buses del transporte público. Dado el nuevo modelo de viaje descrito en 3.1, en el cual los transbordos de una máquina a otra son una parte importante de cada viaje a realizar en el sistema, no todos los viajes son efectivamente cobrados de la misma manera. Lo que sí sucede, es que todos se hacen válidos de la misma forma, es decir, para poder realizar cada viaje, siempre se debe seguir un pequeño protocolo.

Para hacer válido un viaje, al ingresar tanto a un metro como a un bus, el usuario debe acercar su tarjeta a alguno de los *validadores*[10] disponibles en el transporte en cuestión. Al encontrarse en el radio de acción su tarjeta con el validador, este último se encarga de cobrar el pasaje adecuado, dependiendo del tipo de tarjeta en uso y los viajes efectuados previamente. Este proceso actualiza la tarjeta utilizada con los datos necesarios para describir el proceso recién efectuado, guardando parte de esta información para un posterior balance del sistema.

Figura 3.2: Validadores presentes en el transantiago.

Acompañado de una transacción exitosa, viene una luz verde junto a un aviso sonoro, el que se puede interpretar como un "bip!", he allí la relación con su nombre. A su vez, en la pantalla de los validadores, aparece tanto el cobro del viaje, como el saldo restante en la tarjeta, a modo de aviso. Para el caso contrario, se obtiene un sonido distinto de negación junto a una notificación de color rojo. Estas últimas pueden presentarse comúnmente con tres respuestas: *Saldo Insuficiente*, *Tarjeta Bloqueada* o *Tarjeta No Habilitada*.

El primer mensaje aparece en los casos en los cuales la tarjeta no tiene el dinero suficiente para cancelar el viaje en cuestión, por lo cual, debe ser recargada para seguir siendo utilizada.

[9] `http://www.metrosantiago.cl/`

[10] *Validadores*: dispositivos lectores encargados de cobrar el pasaje al usuario, para así acceder al servicio en los distintos medios de transportes disponibles. Ejemplos de ellos son los mostrados en la figura 3.2.

El segundo tipo de mensaje se despliega cuando la tarjeta ha sido modificada en sus saldos y pasa a ser bloqueada en el sistema a fin de no permitirse nuevamente su uso, tal como se explica de forma más detallada en las secciones 3.4.2 y 5.4.1.

Por último, el mensaje *Tarjeta No Habilitada* ocurre cuando la información de la tarjeta no es válida para el sistema. Como ya ha sido expuesto, no existe una comparación con datos históricos almacenados en el sistema en cada validación, por lo cual, los mecanismos de validación utilizados a fin de comprobar la integridad de su información están dentro de las mismas tarjetas. De esta forma, cuando existe algún tipo de problema con el cómo están los datos en la tarjeta, los validadores impiden el uso de ella pero sin bloquearla. Esto sucede debido a que el error de integridad de los datos no necesariamente es culpa del usuario, sino que puede ser del sistema, tal como sucedió en los casos expuestos en [6].

3.3.2. Precios y cobros

Los pasajes por viaje tienen un precio general en los buses, diferenciados dependiendo el horario en el Metro[11], y tarifa reducida para estudiantes en ambos medios de transporte[12].

Como se mencionó en 3.1, los *transbordos* están contemplados como parte importante de cada viaje en el *Transantiago*. Es por esto que pueden realizarse dos cambios de medio de transporte en un periodo de dos horas desde la validación del primer viaje, sin que se efectúe nuevamente un cobro de pasaje. Para combinaciones en las cuales se mezcle un viaje de metro con viajes en micro de distintos costos, se cobra el servicio de mayor tarifa en el momento de efectuar la validación en el transporte en cuestión. Por ejemplo, si una persona cambia de un bus troncal a un metro en horario punta, al validar su viaje en el metro el *validador* cobrará solamente la diferencia entre el pasaje del metro en horario punta -que es más costoso- y el pasaje estándar de los buses. Cabe destacar que en este periodo de dos horas sólo se puede realizar un viaje en metro, y cada nombre de recorrido para los buses debe ser distinto; en caso contrario, volverá a cobrarse un nuevo pasaje. Más información oficial respecto al tema de los cobros, puede encontrarse en [19].

Además de las medidas ya descritas, existe un sistema de préstamo de dinero en el *Transantiago* conocido como *Viaje de Emergencia*[13], en virtud del cual un usuario puede viajar una vez, sin dinero suficiente para pagar su pasaje, quedando así con saldo negativo en su tarjeta. Esto sólo funciona en horarios establecidos, lo cual se funda en el hecho de que en dicho horario los centros de carga están cerrados, y no se puede culpar necesariamente al usuario de no tener dinero en la tarjeta. El pago de este saldo en contra se efectúa una vez que se vuelve a recargar dinero en la tarjeta, por ende se puede volver a acceder al *viaje de*

[11]Con fecha 15 de Octubre del 2015, el precio del pasaje en los buses es de CLP 640, mientras que en el metro es de CLP 610 en Hora Baja (de 6:00 a 6:29 horas y de 20:45 a 23.00 horas, de lunes a viernes), de CLP 660 en Hora Valle (de 6.30 a 6:59 horas, de 9.00 a 17.59 horas y de 20.00 a 20.44 horas de lunes a viernes, más todo sábado, domingo y festivos) y de CLP 720 en Hora Punta (de 7.00 a 8.59 horas y de 18.00 a 19.59 horas, de lunes a viernes).

[12]Con fecha 15 de Octubre del 2015, la tarifa reducida de estudiantes es de CLP 210.

[13]`http://www.tarjetabip.cl/viaje-emergencia.php/`

emergencia, en caso de ser necesario.

3.4. Funcionamiento del sistema de pagos

Poca información pública existe de los múltiples detalles que probablemente tienen las implementaciones tecnológicas necesarias para hacer funcionar un sistema tan grande como el *Transantiago*. Incluso, existen pocas referencias a métricas que puedan definir cuándo el sistema está efectivamente funcionando bien, y cuándo no. Ante esta dificultad, la mayoría de los datos que se obtienen para caracterizarlo provienen de la simple observación, el análisis de tecnologías ocupadas de forma visible, y finalmente, fuentes circunstanciales como discursos o entrevistas de personalidades relacionadas. Dicho esto, se asume que parte de la información entregada a continuación puede no ser del todo precisa, pero se cree que con alta probabilidad, es correcta. Esta formará una base fundamental para los análisis experimentales planteados en los próximos capítulos.

3.4.1. Manejo de Datos

El funcionamiento del *Transantiago* siempre se pensó de manera *offline* en sus validadores [22], es decir, que en el flujo normal de operaciones no se contemplan modificaciones directas en algún registro central o base de datos que administre la información del sistema. Esto tiene sus beneficios, dado que el sistema no se arriesga a producir inconsistencias por problemas en la verificación de datos o a recibir ataques directos a sus registros almacenados. Esta particular implementación *offline* tampoco considera consultas en tiempo real, lo cual tiene inconvenientes importantes, ya que no permite la verificación de los datos de una tarjeta en el mismo momento de su uso. Por esta razón, a lo largo del presente estudio se considerará que el sistema confía en la información de las tarjetas utilizadas, ya que de cierta manera, siempre supone que la información almacenada en la tarjeta es correcta, o que al menos, ésta no provocará daños importantes en caso de ser corrupta.

Tal como describió Andrés Navarro en su rol de presidente de la empresa *Sonda*, en su discurso *Nuestra verdad sobre el Transantiago: una lección de resiliencia* [22], el diseño original del nuevo sistema contemplaba la descarga de los datos acumulados en un recorrido completo, de forma inalámbrica y automática al llegar a la estación terminal. Posteriormente, estos datos serían revisados y agregados al registro oficial, en un proceso denominado *clearing*. Este proceso es importante para el sistema ya que más allá de poder generar un registro robusto y verificar eventuales problemas de consistencia en los flujos de las tarjetas, puede determinarse el cómo han de repartirse los gastos, responsabilidades y ganancias entre las distintas empresas participantes del *Administrador Financiero del Transantiago*[14]. Debido a diversos problemas de planificación e infraestructura, esta modalidad de recolección de la información no logró instaurarse en todos los terminales de buses, teniendo que realizarse

[14] `https://es.wikipedia.org/wiki/Administrador_Financiero_de_Transantiago/`

incluso recolecciones de forma manual en algunos terminales a cargo de personal de *Sonda*, al final de los horarios de trabajo de los operadores.

Junto a estos problemas, la necesidad de agregar más máquinas debido a la insuficiente oferta de transporte frente a la alta demanda de los usuarios al principio de la implementación del sistema, requirió medidas rápidas, las que también afectaron el funcionamiento mismo del sistema. Como solución, se implementaron e improvisaron nuevos buses, en los que se prescindió del uso de computadores que almacenaran y transmitieran información. En estos buses se utilizaron sólo validadores, de forma independiente, los cuales son capaces de resguardar toda la información recopilada en sus recorridos previos al *clearing* respectivo. Para esto, fue necesario incluso sacar algunos validadores de buses ya preparados con dos validadores, en pos de una mayor cobertura del sistema.

El proceso de *clearing*, se realiza en periodos de cada 3 o 4 días, existiendo así un pequeño desfase de la información almacenada con respecto a lo que está sucediendo en el sistema. Esta información es visible en el portal *bip! en línea*[15], en el cual pueden consultarse las últimas transacciones de determinada tarjeta en un periodo máximo de 90 días. Es por ello, que las tarjetas son los dispositivos más actualizados en todo el sistema. Ellas son quienes deben tener la información para determinar cuánto dinero tienen disponibles efectivamente, cuándo no se debe cobrar, cuándo se puede acceder a tarifas disminuidas, cuándo se hizo transbordo e incluso en cuáles buses se viajó últimamente. Esta característica de las tarjetas es un antecedente importante, el cual será explotado a futuro en este estudio.

3.4.2. Bloqueo de tarjetas

Luego de la distribución masiva de la aplicación de celular *Carga Bip!*, la cual utilizando tecnología NFC sobrescribía el saldo de alguna tarjeta con CLP 10.000, se enfrentó un serio problema como sistema ante la gran cantidad de tarjetas modificadas utilizando dinero inexistente para viajar [20]. Dado el mecanismo de *clearing* de datos ya mencionado en el sistema y las características mismas de la arquitectura del *transantiago*, es prácticamente imposible poder detectar un fraude de este estilo en el mismo momento en que se efectúa. Es por esto que las medidas de prevención para este tipo de ataques se toman de forma posterior al proceso de análisis de datos recopilados.

Por consiguiente, detectada una tarjeta adulterada en la cual su saldo no sigue un flujo lógico en sus datos almacenados, ésta pasa a formar parte de una *lista negra* de tarjetas modificadas, las cuales al primer intento de uso normal son bloqueadas para su posterior uso en el sistema. Estas *listas negras* deben, por ende, estar cargadas en los validadores de cada medio de transporte del sistema, y probablemente tienen un máximo de registros aceptables para no aumentar los costos en memoria y procesamiento en cualquier transacción. Más información respecto a estas listas negras y al bloqueo de tarjetas, se encontrará en las conclusiones del análisis experimental realizado en este trabajo.

[15] `http://pocae.tstgo.cl/PortalCAE-WAR-MODULE/`

Capítulo 4

Atacando la tarjeta *bip!*

4.1. Introducción

En el presente capítulo se estudia la efectividad de ataques conocidos sobre la tarjeta *bip!*, pero en particular sobre su base tecnológica, la tarjeta *MIFARE Classic* de 1 kB (para mayor información sobre esta última, sus especificaciones técnicas, vulnerabilidades y posibles ataques, dirigirse al capítulo 2). Conocidos sus ataques y explotables vulnerabilidades, esta parte del estudio prueba implementaciones de ataques conocidos que logran quebrar la seguridad entregada por el cifrador CRYPTO1. Para esto, se sigue la guía de ataque presentada tanto en [14] como en [15], las que implementan los ataques 2.5.6 y 2.5.5 previamente estudiados, a fin de obtener todas las claves de acceso de alguna tarjeta. Finalmente, se procede a presentar los resultados obtenidos de esta prueba práctica, y analizar sus implicancias sobre el *Transantiago* mismo.

4.2. Preparación previa

A continuación se muestran los distintos recursos utilizados en la implementación del análisis.

4.2.1. Hardware utilizado

1. **Lector de tarjetas.** Para esta labor se utiliza un lector NFC modelo SCL3711, muy útil dada su movilidad e independencia. Permite leer todo tipo de tarjetas inteligentes que sigan la norma ISO 14443 y dispositivos que utilicen NFC, en particular las tarjetas MIFARE a analizar. Trabaja conectado vía USB a la estación de trabajo. Para mayores especificaciones técnicas se puede revisar el anexo 7.3.

Figura 4.1: Lector de tarjetas utilizado junto a la estación de trabajo.

2. **Estación de trabajo.** Para poder operar el lector de tarjetas y realizar cualquier análisis y procesamiento de los datos obtenidos a lo largo de este estudio, se utilizó un Notebook *Dell Inspiron 14 N4030*, provisto con el sistema operativo *Ubuntu*[1] *12.04*, el cual es una popular distribución de *Linux*. Para mayores especificaciones técnicas del equipo, ver anexo 7.4.

3. **Segunda estación de trabajo.** Se agregó una segunda estación de trabajo en el proceso en remplazo a la primera, cumpliendo así las mismas funciones detalladas en 2. En este caso, el equipo utilizado fue un Notebook *Lenovo Think Pad Edge E420*, provisto con el sistema operativo *Ubuntu*[2] *14.04*. Dado que para efectos prácticos de este estudio las diferencias entre esta estación de trabajo y la previa son prácticamente nulos, se les denotará de forma indistinguible a ambos como *estación de trabajo* a lo largo de este documento. Para mayores especificaciones técnicas del equipo, visitar el anexo 7.4.

4.2.2. Software utilizado

1. **Libnfc**[3]**.** Biblioteca de código libre que permite establecer comunicación de bajo nivel con distintos tipos de lectores NFC. Soporta múltiples sistemas operativos, y es importante dado que provee una API para programadores, sobre la cual trabajan las implementaciones de ataques utilizadas en esta prueba práctica.

2. **Mfcuk**[4]**.** Creado por Andrei Costin, es una implementación del ataque 2.5.5, presentada originalmente en [9], la cual permite la recuperación de una clave de alguno de los sectores de la tarjeta probada luego de una serie de consultas, utilizando información que entregaban los mensajes de error obtenidos cada ciertos intentos de autentificación. Para este análisis se utilizó la revisión 94 del código en cuestión.

[1] `http://www.ubuntu.com/`
[2] `http://www.ubuntu.com/`
[3] `http://nfc-tools.org/index.php?title=Libnfc`
[4] `https://github.com/nfc-tools/mfcuk`

3. **Mfoc**[5]**.** Implementación realizada por el grupo Nethemba del ataque 2.5.6, presentado originalmente en [12]. Este permite, a groso modo, encontrar todas las claves de una tarjeta a partir de alguna de ellas, aprovechando que se requiere menos información al autentificarse desde un sector a otro que al autentificarse en un sector por primera vez. Con esto es posible obtener finalmente control total sobre la información que se encuentra en la tarjeta. Se utilizó la versión 0.10.7 del código.

4.3. Accediendo a la tarjeta *bip!*

Conocidos los requerimientos para ejecución del análisis de los ataques propuestos en 4.1, se procedió a su ejecución. El experimento a realizar cuenta de tres pasos:

Experimento 1 Acceder a las tarjetas *bip!*

1: Establecer comunicación con tarjetas del sistema mediante el *lector de tarjetas* y la *estación de trabajo*.
2: Ejecutar *mfcuk* sobre alguna tarjeta *bip!* y recuperar alguna clave de la misma.
3: Entregándole la clave obtenida en el paso 2, correr *mfoc* sobre la misma tarjeta para obtener las 31 claves restantes.

Se comenzó el experimento con la instalación del *software* requerido junto a la ejecución de las pruebas pertinentes, a fin de verificar que la conexión entre el *lector de tarjetas* y la *estación de trabajo* operaba de la forma esperada. Hecho esto, se procedió a revisar la capacidad de reconocer correctamente las tarjetas *bip!* por el lector, con lo que recibida una respuesta positiva se pudo continuar con la aplicación de ataques. Esto completaba el paso 1 del experimento 1.

Siguió en el proceso la ejecución de *mfcuk* sobre diversas tarjetas, procurando abarcar todos los tipos de tarjetas existentes en el sistema, descritos previamente en 3.2. Dada la dificultad para conseguir algunas de éstas, en particular las *tarjetas personalizadas* (3.2.2), se realizaron pruebas iniciales con un número pequeño de tarjetas de cada tipo, a fin de obtener información inicial relevante, la que sería corroborada posteriormente con un mayor número de tarjetas.

La primera tarjeta probada con *mfcuk* fue una *tarjeta bip! al portador* (3.2.1). El proceso de prueba se ejecutó durante un periodo de 12 horas, y dado que no mostraba señales de obtener resultados, fue detenido. Este experimento se repitió un par de veces más con idénticos resultados, por lo que se procedió a utilizar otro tipo de tarjetas. Se cambió a una tarjeta *bip! personalizada*, con la que repitiendo el mismo experimento, los resultados fueron distintos. Luego de aproximadamente 3 horas de procesamiento se obtuvo finalmente la primera clave que participa en el sistema: `3df14c8000a1`.

[5] `https://github.com/nfc-tools/mfoc`

A pesar de tener con la clave obtenida información suficiente para continuar el análisis hacia el paso número 3, al menos para el caso particular de las *tarjetas bip! personalizadas*, se continuó aplicando el programa *mfcuk* sobre los dos tipos de tarjetas restantes. En el caso de la *tarjeta nacional estudiantil* (3.2.2) el resultado fue similar al de la *tarjeta bip! personalizada*, entregando al finalizar la clave `3df14c8000a1`, idéntica a la ya obtenida. Para el caso de la *tarjeta bip! bancaria* (3.2.2), el proceso se extendió tanto tiempo como sucedió con la tarjeta *bip! al portador*, por lo cual se le detuvo en los distintos intentos realizados, sin obtener resultado positivo. Resaltar así, que no se pudo explicar por qué este último experimento no funcionó tanto en la *tarjeta bip! al portador* como con la *tarjeta bip! bancaria* probadas.

Dado que ya se había encontrado una clave para dos tarjetas particulares de dos tipos distintos de tarjetas del sistema, se continuó con la utilización de la aplicación *mfoc* sobre ellas, a fin de obtener todas las claves posibles. En ambos casos la ejecución del programa duró aproximadamente 10 minutos, obteniéndose en el proceso las diversas claves para todos los sectores de las tarjetas, y finalmente, toda la información que se encontraba almacenada dentro de ellas. Con esto, el experimento propuesto mostró su efectividad con al menos dos tarjetas particulares de dos tipos de tarjetas del *Transantiago*, logrando vulnerar la privacidad de la información resguardada en estas tarjetas particulares. De esta forma, se completó el experimento 1 exitosamente, sentando las bases para estructurar los ataques siguientes.

4.4. Resultados

4.4.1. Claves del sistema

Concretado el acceso a dos tarjetas particulares de distintos tipos en el *Transantiago* y obteniendo completo acceso a su información, se comprobó que en ambos casos la clave obtenida mediante la ejecución de *mfcuk* correspondía a una clave tipo A del sector 15. A su vez, y más importante aún, fue evidente que ambas tarjetas poseían idénticas claves para todos sus sectores, lo que se tradujo directamente en el planteamiento de la siguiente hipótesis:

Hipótesis 1 *Todas las tarjetas utilizables en el Transantiago ocupan las mismas claves de autenticación para sus respectivos sectores.*

Tal como presenta el trabajo [36], esta modalidad de implementación en la que las tarjetas comparten las claves de autenticación para todos los sectores, es una de las opciones disponibles para el sistema. Asimismo descarta de lleno la utilización de diversificación de claves como medida de seguridad, tal como se ocupó en el sistema Holandés. Por otro lado, como ventaja permite una mayor facilidad de implementación y probablemente de desempeño para las distintas partes que requieran acceso a tarjetas en el sistema.

Para probar esta hipótesis se utilizaron las 32 claves obtenidas para intentar acceder a un amplio número de tarjetas, a fin de validar la información. Dada la cantidad de tarjetas con

que esto efectivamente funcionó, tal como representa la tabla 4.1 y considerando que no existió ningún tipo de complicación ni excepción, se consideró una conjetura sólida finalmente que las 32 claves encontradas en el experimento 1 son constantes para todo el sistema y utilizadas en cada tarjeta que pertenezca al *Transantiago*. De este modo, la Hipótesis 1 se consideró cierta, convirtiéndose en el primer hecho de este trabajo.

Tipo Tarjeta bip!	Cantidad
Al portador	18
Bancaria	8
Personalizada	1
TNE Antigua	20
TNE Nuevas	12
Total	**59**

Tabla 4.1: Cantidad de tarjetas accedidas con las claves del sistema.

Hecho 1 *Todas las tarjetas utilizables en el Transantiago ocupan las mismas claves de autenticación para sus respectivos sectores.*

La lista de todas las claves presentes en el sistema, tanto tipo A como B, se encuentran en la tabla 4.2.

Sector	Clave A	Clave B
0	3a42f33af429	1fc235ac1309
1	6338a371c0ed	243f160918d1
2	f124c2578ad0	9afc42372af1
3	32ac3b90ac13	682d401abb09
4	4ad1e273eaf1	067db45454a9
5	e2c42591368a	15fc4c7613fe
6	2a3c347a1200	68d30288910a
7	16f3d5ab1139	f59a36a2546d
8	937a4fff3011	64e3c10394c2
9	35c3d2caee88	b736412614af
10	693143f10368	324f5df65310
11	a3f97428dd01	643fb6de2217
12	63f17a449af0	82f435dedf01
13	c4652c54261c	0263de1278f3
14	d49e2826664f	51284c3686a6
15	3df14c8000a1	6a470d54127c

Tabla 4.2: Claves A y B del sistema

4.4.2. Caso particular: TNE

A partir del año 2015, la TNE cambió su base tecnológica, pasando de las tarjetas MIFARE Classic de 1 kB, al mismo modelo de tarjeta, pero esta vez, de una capacidad de 4 kB. Tal como se explicó en la sección 2.2.2, las tarjetas MIFARE Classic de 4 kB utilizan una estructura de sectores diferente a su versión de 1 kB, y en particular, tienen un mayor número de claves, dado que tienen una mayor cantidad de sectores. Puesto que este cambio ocurrió en medio de la realización de este estudio, su análisis no estaba contemplado, y con mayor razón, si es que esto consideraba cambios importantes respecto de la versión que ya se estaba analizando.

A pesar de esto, un análisis simple reveló que los cambios en las tarjetas realizados en este proceso no eran relevantes, al menos superficialmente para la mantención del sistema antiguo, el cual se presumía seguiría funcionando de manera idéntica, en paralelo, en los otros tipos de tarjetas. Esto puede asegurarse, dado que todos los sectores extra que contiene una tarjeta MIFARE Classic de 4 kB utilizada como TNE respecto a una de 1kB mantienen las claves A y B que traen las tarjetas desde su fabricación por defecto, y se encuentran vacíos, sin ninguna medida de validación, como los *bytes de verificación* (analizados en 5.4.2), ni muestra de alteración con los usos. Las claves utilizadas en estos bloques extra son `ffffffffffff`, tanto para las tipo A como para las tipo B. En los anexos 7.8 y 7.9, es posible ver lecturas tarjetas MIFARE Classic utilizada en el sistema, de tanto 1 kB como de 4 kB, respectivamente, para poder comprobar las diferencias aquí expuestas.

De esta forma, podemos asegurar el siguiente *hecho*:

Hecho 2 *Las TNE de 4 kB se utilizan de igual forma y con la misma estructura que las demás tarjetas bip! de 1 kB en el Transantiago, manteniendo sus sectores extra sin ocupar y vacíos.*

4.4.3. Información en una Tarjeta Bip!

En consecuencia al acceso obtenido a los datos de los distintos tipos de tarjetas *bip!* producto del experimento anterior, más el análisis que requirieron los distintos experimentos y ataques propuestos en el siguiente capítulo, se logró compilar toda la información detallada en las figuras 4.2 y 4.3 respecto a los datos almacenados en las tarjetas *bip!* en general. Esta información está contenida en cada tarjeta del sistema con leves diferencias para determinados tipos. Es menester expresar que, toda la estructura aquí presentada fue obtenida y probada sólo experimentalmente, por lo cual podría no corresponder absolutamente con la especificación real del sistema (esto es, puede diferir en casos puntuales).

A la derecha de le figura 4.2 y a la izquierda de 4.3 se explican los significados de los datos almacenados en cada uno de los bloques de una tarjeta. Entre corchetes (`"[]"`) se encuentran los bytes exactos en los que está la información en cuestión (estos bytes se cuentan de izquierda a derecha, comenzando de 0). En color amarillo se denota el bloque 0, bloqueado para escritura desde su fabricación, el cual almacena el UID de la tarjeta y datos del fabricante. En verde se encuentran bloques con información únicamente disponible en las TNE. Con fondo azul, se

tienen bloques configurados como de *valor* (para dudas sobre los tipos de bloques, referirse a 2.2.2). Por último, en fondo rojo, se pueden encontrar los bloques *trailers* de cada sector, los cuales contienen las claves A y B para limitar el acceso al sector al cual corresponden (para más información respecto a este tipo de bloques, dirigirse a 2.2.2).

Aparte de esta información recopilada, es importante destacar los procedimientos utilizados en la obtención de la misma y el significado que ésta tiene en el sistema. El primero, y más evidente, fue la lectura y análisis de los datos obtenidos de distintas tarjetas utilizadas constantemente, efectuando en la práctica un seguimiento de todos los cambios que fueron ocurriendo durante aproximadamente 3 meses en ellas. Este conjunto de datos almacenados permitió concluir comportamientos en determinados escenarios y deducir el significado de ciertos datos que variaban dentro de las tarjetas en determinadas condiciones.

Además de este proceso, se estudió la información entregada en el estudio de información almacenada en la OV-Chipkaart [21], la cual no aportó de mucho a este estudio, debido a las notorias diferencias entre las implementaciones de los sistemas.

Como siguiente medida, se realizó ingeniería inversa de la aplicación *Android Saldo Bip!*[6] (detallada en 2) con la herramienta web *android APK decompiler*[7]. La idea de reversar esta aplicación *Android* emerge al procurar entender cómo ésta logra, mediante el uso de tecnología NFC, entregar información de cualquier tarjeta sin ningún mayor requerimiento que el sólo acceso a ella. Entre la información que entrega esta aplicación destaca el saldo, los últimos tres usos, las últimas tres recargas, el número de la tarjeta, el número de chip y el último uso registrado.

Al aplicar ingeniería inversa a la aplicación *Saldo Bip!* se permitió comprender su lógica, y a su vez ayudó a entender la estructura de los datos almacenados en la tarjeta para los distintos tipos posibles; conocer donde se almacena información respecto a los viajes y recargas efectuadas en cada tarjeta; y, más importante aún, corroborar el hecho de que se mantiene almacenada información relevante en cada tarjeta que podría modificar las respuestas de los validadores y el flujo de uso que tendría la misma en el sistema. Esta deducción es considerada relevante para este estudio y por consiguiente, se destaca en el siguiente *hecho*:

Hecho 3 *Las tarjetas contienen información vital para determinar cómo los validadores reaccionarán ante sus usos; información en la cual estos validadores confían. Esto no es cierto para el backend*[8] *del sistema.*

[6] `https://play.google.com/store/apps/details?id=bip.app.saldobip`

[7] `http://www.decompileandroid.com/`

[8] *Backend*: Procesos que se utiliza en el servidor de un sitio o un sistema, con todos sus respectivos mecanismos para lograr resolver las peticiones de los usuarios. En este ejemplo particular, sería toda la lógica que mantiene los datos del sistema en el servidor central funcionando de forma correcta y coherente.

La última aseveración, nace del hecho de que el *backend* del sistema sí busca validar la información que le entregan las tarjetas y los distintos usos que éstas tienen, tal cual como se pudo ver en el análisis de los bloqueos en la sección 3.4.2. Por último, y debido a lo comentado previamente en 3.4 respecto a la confianza de los validadores para con la información contenida en las tarjetas a los que son expuestos, la modificación de información de cada tarjeta claramente entrega un vector de ataque. Esta debilidad será explotado en el próximo capítulo de múltiples maneras.

000: ea c4 76 80 d8 88 04 00 c1 85 14 99 51 00 45 12	UID [0-3]
001: 00 00 00 00 27 60 ff 00 00 00 00 00 00 00 00 42	ID Transantiago [4-7]
002: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
003: 3a 42 f3 3a f4 29 78 77 88 00 1f c2 35 ac 13 09	Trailer de Sector
004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
007: 63 38 a3 71 c0 ed ff 07 80 00 24 3f 16 09 18 d1	Trailer de Sector
008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
011: f1 24 c2 57 8a d0 ff 07 80 00 9a fc 42 37 2a f1	Trailer de Sector
012: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	Nombre usuario TNE
013: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	RUN usuario TNE
014: 01 e1 cc e7 09 00 00 00 00 00 00 00 00 00 00 62	
015: 32 ac 3b 90 ac 13 78 77 88 00 68 2d 40 1a bb 09	Trailer de Sector
016: 11 22 20 49 d7 06 5f 19 90 74 79 21 00 40 00 7c	Mes y día último uso [10-11]
017: 70 6b 00 00 00 00 00 00 00 00 00 40 08 84 04 01	Número de Uso [1-3], Registro [0-1]
018: 70 6b 00 00 00 00 00 00 00 00 00 40 08 84 04 01	Número de Uso [1-3], Registro [0-1]
019: 4a d1 e2 73 ea f1 7e 17 88 00 06 7d b4 54 54 a9	Trailer de Sector
020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
021: 40 de 47 cf 03 00 20 52 82 04 00 02 00 04 04 91	
022: 40 de 47 cf 03 00 20 52 82 04 00 02 00 04 04 91	
023: e2 c4 25 91 36 8a 7e 17 88 00 15 fc 4c 76 13 fe	Trailer de Sector
024: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
025: 00 00 00 00 ff ff ff ff 00 00 00 00 19 e6 19 e6	
026: 00 00 00 00 ff ff ff ff 00 00 00 00 1a e5 1a e5	
027: 2a 3c 34 7a 12 00 18 77 8e 00 68 d3 02 88 91 0a	Trailer de Sector
028: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5	
029: 00 00 00 00 ff ff ff ff 00 00 00 00 1d e2 1d e2	
030: 00 00 00 00 ff ff ff ff 00 00 00 00 1e e1 1e e1	
031: 16 f3 d5 ab 11 39 18 77 8e 00 f5 9a 36 a2 54 6d	Trailer de Sector

Figura 4.2: Representación de los primeros 8 sectores almacenados en una tarjeta *bip!*

	032:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
Saldo	033:	c4	04	00	00	3b	fb	ff	ff	c4	04	00	00	90	6f	90	6f
Saldo	034:	c4	04	00	00	3b	fb	ff	ff	c4	04	00	00	90	6f	90	6f
Trailer de Sector	035:	93	7a	4f	ff	30	11	18	77	8e	00	64	e3	c1	03	94	c2
	036:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
Datos Última Recarga	037:	00	00	00	00	00	00	00	00	f0	05	54	02	37	00	00	b5
	038:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
Trailer de Sector	039:	35	c3	d2	ca	ee	88	7c	37	88	00	b7	36	41	26	14	af
Información Últimas Cargas	040:	5f	b2	97	56	47	04	00	40	19	a0	0f	00	00	00	00	74
Información Últimas Cargas	041:	5f	b8	d7	04	5a	04	00	40	19	40	1f	00	00	00	00	96
Información Últimas Cargas	042:	95	b0	27	d3	50	cc	b8	67	19	a0	0f	40	00	00	00	a2
Trailer de Sector	043:	69	31	43	f1	03	68	78	77	88	00	32	4f	5d	f6	53	10
Información Últimos Usos	044:	c4	b2	b7	02	33	c0	26	00	46	19	00	0a	40	f0	39	e4
Información Últimos Usos	045:	04	ce	a7	0c	23	c0	26	00	46	19	00	0a	40	8c	57	96
Información Últimos Usos	046:	41	de	17	cf	23	80	94	20	45	19	50	0a	00	00	00	fc
Trailer de Sector	047:	a3	f9	74	28	dd	01	7f	07	88	00	64	3f	b6	de	22	17
	048:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	049:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
Trailer de Sector	051:	63	f1	7a	44	9a	f0	ff	07	80	00	82	f4	35	de	df	01
	052:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	053:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	054:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
Trailer de Sector	055:	c4	65	2c	54	26	1c	ff	07	80	00	02	63	de	12	78	f3
	056:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	057:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	058:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
Trailer de Sector	059:	d4	9e	28	26	66	4f	ff	07	80	00	51	28	4c	36	86	a6
	060:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	061:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
	062:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	f5
Trailer de Sector	063:	3d	f1	4c	80	00	a1	ff	07	80	00	6a	47	0d	54	12	7c

Figura 4.3: Representación de los últimos 8 sectores almacenados en una tarjeta *bip!*

Capítulo 5

Ataques sobre el Transantiago

5.1. Introducción

En el presente capítulo se exponen diversos ataques realizados sobre el sistema de transporte chileno, conocido como *Transantiago*, engendrados a partir de las posibilidades de ataque abiertas posterior a la realización del experimento 1 detallado en el capítulo 4, con el cual se logra acceder a toda la información escrita en cada tarjeta *bip!*.

Estos ataques se dividen en dos tipos dependiendo su naturaleza, clasificándose entre *ataques de lectura* y *ataques de escritura*. Para todos estos ataques, se analizará la vulnerabilidad explotada, su aplicación práctica, explotabilidad, los riesgos que conlleva su posible realización y por último una propuesta de mitigaciones factibles.

Además, y a modo de buena fe respecto a las intensiones de esta tesis y sus fines últimos, se estableció la siguiente cláusula, la cual regula el proceder seguido ante todos los experimentos efectuados en este trabajo.

"Cada vez que se pruebe algún tipo de ataque sobre el sistema que resulte en algún tipo de beneficio para el atacante -en este caso el autor de este trabajo- el servicio no será utilizado realmente, a fin de minimizar el eventual efecto negativo que estas acciones puedan tener sobre el mismo sistema".

La idea de esta cláusula es poder evitar, de la mayor forma posible, los efectos negativos que tengan sobre el sistema los experimentos efectuados. Puesto que además estos experimentos se realizan con el sistema en funcionamiento (ya que no es posible detenerlo o crear un ambiente de pruebas, debido a la poca información que tenemos del mismo), no se conoce realmente cuál puede ser el efecto final del uso de datos falsos o tarjetas corruptas, por lo que se realizan el menor número de pruebas posibles. Además, en caso de encontrar algún ataque que se traduzca en beneficios económicos de distinto tipo, como por ejemplo viajes gratis o saldo extra, estos no se ocuparán una vez validados o probados en sus experimentos, con la intención de evitar cualquier costo que pueda implicar al sistema. Una situación importante que se evita con esta medida, por ejemplo, es contribuir a la congestión en el servicio mismo.

Finalmente, es importante dejar en claro que el espíritu de este estudio es revelar problemas del sistema de transporte chileno a fin de contribuir de alguna forma a su mejora, buscando evitar eventos catastróficos como los ocurridos con las cargas de tarjetas en el año 2014 [11], aportando así a su perfeccionamiento.

5.1.1. Estructura de los análisis

El análisis de los ataques se divide en cinco partes: *vulnerabilidad*, *ataque*, *experimentos asociados*, *riesgo* y *mitigación*.

1. En *vulnerabilidad* se detalla todo el contexto en el sistema que hace posible que el ataque a describir sea explotable.
2. En *ataque* se describe, a grandes rasgos, cómo utilizar la vulnerabilidad recién mostrada, tanto contra los usuarios como contra el sistema.
3. En *experimentos asociados* se detallan múltiples procedimientos realizados para poder estudiar la vulnerabilidad que se intenta explotar en cada caso, además de la factibilidad y/o efectividad del ataque propuesto. En esta sección se plantean hipótesis, experimentos y hechos, que ayudan a entender, de mejor forma, la vulnerabilidad referida. En general, el relato no corresponde a todo lo efectivamente realizado en el análisis original, sino que a los datos más relevantes, dispuestos de una manera lógica y comprensible de seguir. Por último, cabe destacar que esta sección sólo tiene cabida en los ataques de escritura.
4. En *riesgo* se analiza cuál es el peligro de que esta vulnerabilidad actualmente exista en el sistema. Dada esta definición, en la clasificación que obtenga cada ataque influye tanto la probabilidad y facilidad con que el ataque puede ser concretado, como las repercusiones que este mismo tendría sobre las víctimas involucradas. En los casos que se considere necesario, este análisis puede realizarse de forma separada para darle el énfasis necesario a los casos estudiados. La clasificación que se le da a los riesgos asociados a un ataque puede ser *bajo*, *medio* o *alto*, dependiendo de las conclusiones obtenidas de su análisis.
5. Finalmente, en *mitigación* se plantean propuestas de solución a la vulnerabilidad descrita y/o a ataques particulares mencionados, según requiera el caso.

5.2. Preparación previa

A continuación, se detallan los recursos utilizados en los distintos ataques especificados en este capítulo, que no fueron detallados en el capítulo 4 en su sección 4.2.

5.2.1. Hardware utilizado

Sumándose al hardware especificado en la sección 4.2.1, se utilizó el siguiente dispositivo.

1. **Smartphone Moto X (primera generación).** Teléfono celular fabricado por la compañía *Motorola*, compatible con el sistema operativo *Android*. Este teléfono posee tecnología NFC, la que se utiliza para la lectura y escritura de tarjetas *bip!* y otras, gracias a distintas aplicaciones especificadas en 5.2.2. Durante este estudio, se utilizaron las versiones de *Android KitKat* (4.4) y *Lollipop* (5.1) como sistemas operativos de este dispositivo. Las especificaciones generales del teléfono pueden encontrarse en el anexo 7.6.

5.2.2. Software Utilizado

Sumándose al software ya especificado en la sección 4.2.2, se utilizaron las siguientes aplicaciones para trabajar con el *smartphone* descrito en 1.

1. **Mifare Classic Tool**[1]**.** Aplicación creada por *IKARUS Projects*[2] para leer, escribir y analizar *dumps* de tarjetas MIFARE Classic. Permite manejar archivos con distintas claves, leer bloques particulares o tarjetas enteras, guardar lecturas de tarjetas, escribir sobre bloques específicos o escribir tarjetas con copias completas de otras tarjetas. Dada la versatilidad del teléfono móvil, esta aplicación lo transforma en una estación de trabajo móvil utilizable en cualquier momento, lo cual fue útil por sobre todo, en la generación de registros de usos de tarjetas para posteriores análisis de cambios en diversos experimentos.

2. **Saldo Bip!**[3]**.** Aplicación creada por el desarrollador de aplicaciones *Android casalazar.ri*[4] la cual está orientada específicamente en las tarjetas *bip!* chilenas. Esta aplicación utiliza tecnología NFC del equipo en donde se instaló, permitiendo leer el saldo de la tarjeta *bip!* más otras informaciones útiles. Aparte de estas funcionalidades, esta aplicación fue una de las bases que inspiró el proceso de análisis de datos presentes en la tarjeta *bip!*, ya que lee información no especificada en otros medios oficiales disponible en todo momento en la tarjeta. Para mayor detalle de este proceso, dirigirse a 4.4.3.

5.3. Ataques de Lectura

En esta sección se presentan dos ataques que, exclusivamente, requieren la lectura de una tarjeta para poder realizarse.

[1] `https://play.google.com/store/apps/details?id=de.syss.MifareClassicTool`
[2] `https://play.google.com/store/apps/developer?id=IKARUS+Projects`
[3] `https://play.google.com/store/apps/details?id=bip.app.saldobip`
[4] `https://play.google.com/store/apps/developer?id=casalazar.ri`

5.3.1. Obtención de datos del usuario desde la TNE

Vulnerabilidad

Tal y como se clasificaron las tarjetas en la sección 3.2 de este trabajo, tenemos dos grandes grupos en las que dividirlas: *tarjetas anónimas* y *tarjetas vinculadas a un usuario*. Dentro de este último grupo, la unión entre un usuario y una tarjeta *bip!* personalizada o una tarjeta *bip!* bancaria sólo se ve reflejada dentro del sistema mismo, por ejemplo, revisando en páginas como el portal *bip! en línea*. Esto es distinto en los *pases escolares* o TNE, ya que la unión no sólo está en la lógica del sistema, sino que escrita explícitamente en los datos de la tarjeta. Como puede verse en la figura 5.1, los bloques número 12 y 13 tienen los datos correspondientes al Rol Único Nacional (o RUN) y al nombre del dueño del dispositivo, respectivamente; datos que también están impresos de forma física en la tarjeta misma.

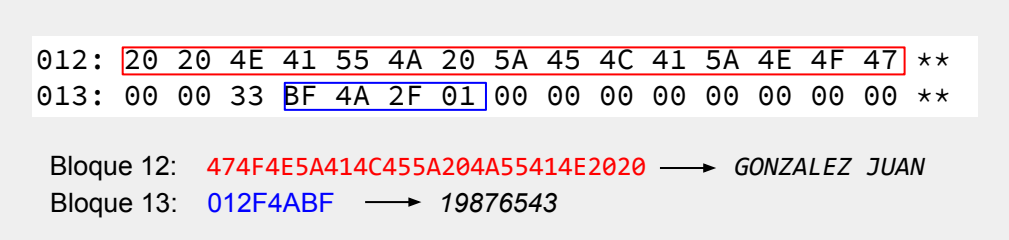

Figura 5.1: Datos en los bloques 12 y 13, en una TNE. La primera flecha representa una conversión a *chars* de números hexadecimales, mientras que la segunda representa una conversión hexadecimales a decimal.

En el caso de los datos del bloque número 12, que corresponden al RUN del usuario, estos están escritos entre los bytes 3 y 7 en números hexadecimales, sin incluir el dígito verificador. Como siempre, y tal como se mencionó previamente, la representación de esta información sigue el formato de palabras *little-endian*. Por otro lado, los datos del bloque número 13, los cuales corresponden al nombre del usuario, están escritos carácter por carácter en su codificación `ASCII` en valores hexadecimales, desde el bit 0 al 14.

Ataque

Obtención del RUT y nombre (un apellidos y el nombre de pila, hasta 15 caracteres, incluyendo espacio entre medio) del dueño de la tarjeta, sin necesidad de ver la tarjeta, sino que únicamente tenerla dentro del rango de acción de un dispositivo NFC, como un teléfono celular.

Riesgo

El riesgo de este ataque es **bajo**. A pesar de la variedad de datos que pueden obtenerse de una persona exclusivamente con su nombre y en particular con su RUN, la privacidad de éstos no puede considerarse alta. Además, destacar que la dificultad de obtenerlos es alta

comparada con otros métodos para realizar lo mismo; por ejemplo, simplemente fotografiando dicha tarjeta.

Mitigación

Dado que según la información con la que se cuenta, el nombre y el RUN grabados en la tarjeta no se ocupan en algún proceso importante en el sistema de transporte, al menos en lo que respecta a viajes, eliminarlos de las tarjetas y del sistema en general, se plantea como el mejor método de mitigación posible para esta amenaza. A pesar de esto, debido al bajo riesgo de ataque, no se considera esta medida como algo del todo necesario, ni esta vulnerabilidad como un problema mayor.

5.3.2. Obtención de la dirección de un usuario.

Vulnerabilidad

Este ataque se basa en el trabajo realizado por miembros de la Universidad de Chile detallado en [4], en el cual utilizaron técnicas de *Data Minning* logrando procesar los datos entregados por la plataforma *bip en línea!*[5] para alguna tarjeta *bip!*, obteniendo así, con una probabilidad de casi un 50 %, la ubicación del paradero más cercano a la casa del dueño de dicha tarjeta. Para este ataque, sólo se necesita la información del ID de la tarjeta en el sistema, encontrado en el bloque número 1 de todas, bytes del 4 al 7. Cabe destacar que esta información también se puede encontrar impresa a un costado de cada tarjeta .

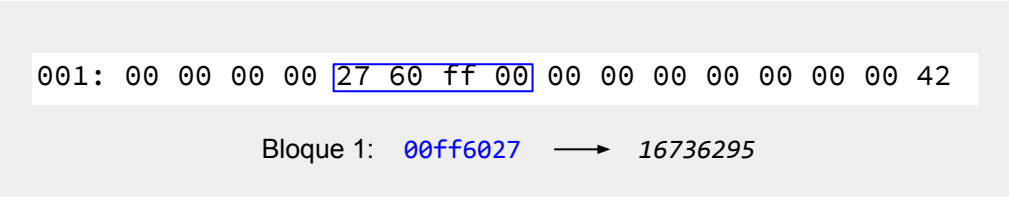

Figura 5.2: Datos en el bloque 1 de una tarjeta *bip!*, el cual contiene el ID de dicha tarjeta en el sistema. La flecha representa una conversión de dígitos en formato hexadecimal a digital.

Es importante mencionar que esta vulnerabilidad no distingue entre los tipos de tarjetas, ya que la información requerida para explotarla está explícita en los datos contenidos en cada tarjeta o puede incluso ser omitida del proceso sin mayores problemas. Esto podría no parecer así, en particular para el caso de las *tarjetas personalizadas*, ya que para acceder a la información del sitio *bip! en línea* se requiere ingresar el RUN del usuario a modo de autentificación, además del número de tarjeta correspondiente. Pero tal como se manifiesta en el trabajo [4], esta validación en la página indicada no presenta un bloqueo real para adversarios que deseen acceder a la información almacenada.

[5] `http://pocae.tstgo.cl/PortalCAE-WAR-MODULE/`, accesible desde `tarjetbip.cl`

Para las TNE, tal como se mostró en el ataque 5.3.1, dicha información puede obtenerse directamente del bloque 12, por lo cual esta validación no representa una dificultad significativa. Para los demás casos de *tarjetas personalizadas* que no contienen esta información, es posible saltar el requerimiento efectuado en la página consultando directamente al servidor, utilizando cualquier otro RUN válido sin necesariamente ser el correspondiente, con lo cual se obtiene directamente la información requerida obviando el bloqueo.

Ataque

Obtención del paradero más cercano a la casa de algún usuario del sistema, con una probabilidad del 50 %, necesitando únicamente el ID en el *Transantiago* de la tarjeta que utiliza. Para esto, sólo se necesita tener la tarjeta dentro del rango de acción de un dispositivo NFC, como un teléfono celular, y posteriormente extraer la información que entrega la página *bip! en línea* respecto a sus últimos movimientos. El procesamiento de esta última requiere conocimientos básicos de *Data Minning*.

Riesgo

Se considera el riesgo de este ataque como **medio**. Esta evaluación se basa en el hecho de que el ataque no es contra el sistema, sino contra el usuario, con datos que este último supone no deberían implicar algún riesgo. Probablemente, ningún usuario esperaría que al entregar el número de su tarjeta *bip!* pudiera tener algún tipo de efecto negativo, menos revelar información privada como su dirección.

A pesar de que el ataque posee cierta dificultad técnica, ésta no es mayor, encontrándose como ejemplo el caso de estudio del trabajo [4]. Por último, los riesgos disminuyen dado que la probabilidad de éxito del ataque es de un 50 %, en la cual influyen principalmente el comportamiento del usuario y la frecuencia de uso de la tarjeta en cuestión. Destacar que el ataque no es un acto conflictivo propiamente tal, sino que la información que entrega da cabida a la realización de otro tipo de ataques sobre el usuario.

Mitigación

La principal fuente de obtención de información para este ataque es la página de *bip! en línea*. Dado que su servicio es útil para el usuario, ya que le permite monitorear sus viajes y recargas registrados (con un retraso por el proceso de *clearing* previamente mencionado), su eliminación no es una opción considerable. La entrega de esta información que parece no generar problemas al ser pública, podría realizarse de una forma mucho más responsable, ya que como muestra el trabajo [4], procesada termina entregando información privada de un usuario, como su dirección o su barrio.

Se propone como opción para solucionar la problemática la instauración de un sistema de cuentas para todos los usuarios, en virtud del cual, cada uno pueda vincular sus tarjetas,

en el momento de compra o en algún establecimiento apropiado, a fin de entregar un acceso limitado para los demás usuarios que desconocen su cuenta y clave. De este modo, podría mantenerse el servicio entregado por *bip! en línea*, pero de forma más restringida para las tarjetas anónimas. Sumado a esto, se propone una revisión del diseño efectuado para la implementación de la aplicación web de *bip! en línea*, dado que la validación de claves puede saltarse sin problemas, con cualquier RUN, utilizando consultas directas al servidor (no por la página misma). Esto es de vital importancia, y solucionarlo no debería ser difícil.

Como otra opción, podría considerarse la eliminación del ID en el *Transantiago* de las tarjetas. Esto se presume no factible, dado que se considera importante la existencia de alguna identificación en el sistema para cada dispositivo funcionando en él, aunque no se conozca un uso concreto de éste. Entre los distintos experimentos realizados en este trabajo, en varios de los cuales se pensó que el ID podía marcar alguna diferencia, siempre el proceso se mostró indiferente a su existencia o cambios efectuados en el mismo. A modo de ejemplo, se logró viajar con IDs del *Transantiago* alterados sin que el sistema de *bip! en línea* siquiera lo notara, realizando los cobros al ID vinculado al UID original de cada tarjeta y no al modificado y almacenado en las mismas. Incluso los bloqueos de tarjetas no se guían por el ID del sistema sino que por el UID de la tarjeta, tal como veremos en el experimento 5.4.1.

Esta última decisión de diseño puede parecer muy sensata. Sabiendo que los datos internos de una tarjeta son en su mayoría modificables por algún eventual adversario, incluyendo el bloque 1 con el ID del *Transantiago*, no parece ser seguro centrar la identificación de una tarjeta, y con ésta la de todas sus transacciones, en un campo así de vulnerable. Por esta razón, la identificación utilizada para todo este tipo de procesos en el sistema recae en el UID de cada tarjeta, ubicado en el bloque 0, el cual está bloqueado para escritura desde su fabricación en todas las tarjetas MIFARE Classic. Esta idea se resume en el siguiente hecho.

Hecho 4 *Para los procesos de identificación de las tarjetas el sistema utiliza el UID ubicado en el bloque 0 de la tarjeta, en vez de la ID del Transantigo ubicada en el bloque 1. Esto principalmente se debe a que el bloque 0 viene bloqueado para escritura vía hardware, lo que impediría eventualmente una modificación.*

De esta manera, aunque esta resolución pueda parecer la lógica, posteriormente en 5.4.5 se presentarán diversas razones por las que dicha resolución no elimina realmente ninguno de los riesgos mencionados a cabalidad.

5.4. Ataques de Escritura

En esta sección se presentan ataques que además de la lectura de datos, requieren modificar campos dentro de la tarjeta misma, atacando la integridad de la información contenida en la tarjeta.

5.4.1. Modificación del saldo en una tarjeta.

Vulnerabilidad

El ataque masivo del 2014, en el cual se utilizaban aplicaciones *Android* para incrementar el saldo de tarjetas *bip!* vía NFC, clarificó la posibilidad de modificación de estos campos en cada tarjeta y lo vulnerable que era el sistema para detectar este tipo de modificaciones en el momento de su uso. Esta posibilidad de ataque se analizará en la presente sección, mostrando además experimentos realizados para comprender cómo funcionan las transacciones ocurridas entre un validador y una tarjeta *bip!* al momento de cobrar.

Tal como se puede ver en la figura 5.3, el saldo de las tarjetas *bip!* está almacenado en los bloques 33 y 34, guardado en ambos casos como *bloques de valor* (para más información respecto a este tipo de bloques, visitar la sección 2.2.2). De esta manera, conociendo el formato de los datos, una modificación correcta implicaría un cambio en el saldo de una tarjeta del sistema, permitiendo esto viajar sin gastar dinero real, pudiendo incluso crear inconsistencias en los saldos totales, movimientos de dineros inexistentes y varios otros problemas asociados.

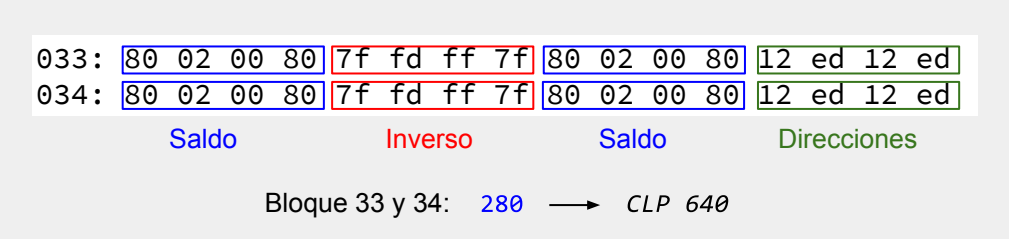

Figura 5.3: Datos en los bloques 33 y 34 de una tarjeta *bip!*. En azul se muestran los saldos, en rojo su valor complemento (en binario) y en verde las direcciones para almacenamiento. La flecha representa una conversión a decimales desde hexadecimales.

A pesar de las advertencias de bloqueos y distintas medidas de disuasión efectuados por los administradores del sistema y el gobierno, el ataque se realizó en masa el año 2014, encontrándose incluso casos de usuarios que aseguraban haber viajado gratis modificando su tarjeta en repetidas ocasiones, sin ser bloqueados. Esto, suponiendo incluso un eventual bloqueo, vuelve rentable y atractivo el ataque principalmente debido a la cantidad de usos posibles frente al costo inicial de una tarjeta nueva. Esta razón, junto a la alta disponibilidad de *smartphones* compatibles con tecnología NFC y la facilidad de acceso a las aplicaciones que lograban incrementar el saldo, fueron las principales causales de la masividad del ataque dicho año (más información de este ámbito, dirigirse a 3.4.2).

Ataque

Modificación del saldo de una tarjeta *bip!* para obtener un beneficio económico, pudiéndose acceder al servicio ofrecido por el *Transantiago* sin generar un costo al atacante. El uso de una tarjeta modificada, eventualmente, se traducirá en el bloqueo de la misma.

Experimentos asociados

Modificando tarjetas. Entendida la distribución de la información de varios de los bloques en las tarjetas *bip!* gracias a haber realizado procesos como los descritos en la sección 4.4.3, se procedió a utilizar este conocimiento y modificar los datos en tarjetas buscando manipular su comportamiento en el *Transantiago* a voluntad. Dado que tampoco se conoce totalmente el sistema, esto además permite entender sus reacciones ante distintos escenarios, comprendiendo cómo maneja errores de consistencia de datos y encontrar, al final, vulnerabilidades explotables contra las que se considera importante defender al sistema.

El primer experimento realizado de modificación de datos fue la manipulación de los saldos descritos en los bloques 33 y 34. Estos están configurados como *bloques de valor*, probablemente para el uso de funciones predefinidas de las tarjetas MIFARE Classic, como aumentos o decrementos. En estos bloques, el máximo tamaño de datos a guardar es de 4 bytes, por lo cual, dado que se sigue el estándar *two's complement* en esta representación, el rango de valores posibles a almacenar es, en teoría $[-2147483648, 2147483647]$. En la práctica, el rango válido para los saldos es mucho menor, ya que el máximo valor válido en una tarjeta para el sistema es de CLP 25.500[6], y el mínimo, dadas las tarifas y la posibilidad de préstamo de dinero en ciertos horarios, es de CLP -720[7]. Así, este rango de valores es fácilmente representable por 2 bytes, por lo cual siempre habrán bits no ocupados del total disponible.

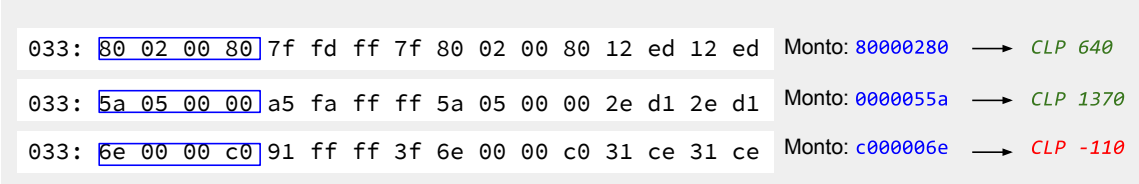

Figura 5.4: Tipos de saldos posibles en el *Transantiago*. Las flechas representan conversiones a decimales desde números hexadecimales. En verde, los distintos tipos de saldos positivos encontrados; en rojo, un saldo negativo.

Revisando saldos guardados en distintas tarjetas, tal y como ejemplifica la figura 5.4, en particular saldos negativos, es fácil notar que no siguen el formato especificado por el estándar *two's compliment*, sino que una convención propia en la que el bit 1 del byte 0 de los bloques que contienen los saldos (34 y 34) será 1 en caso de que el valor almacenado sea negativo, y 0 cuando no lo sea. Este comportamiento está descrito en el siguiente hecho:

[6]Fuente: `http://tarjetabip.cl/como-funciona.php`

[7]Fuente: `http://www.transantiago.cl/tarifas-y-pagos/conoce-las-tarifas`, en Noviembre del 2015

Hecho 5 *Los saldos almacenados en los bloques 33 y 34 se escriben directamente de forma estándar, tanto para valores positivos como para valores negativos. Dado que el espacio disponible es de 4 bytes, pero el saldo no puede ocupar más de 2 bytes de capacidad por los montos permitidos en el sistema, el byte 1 de los bloques 33 y 34 siempre es* 00 *en representación hexadecimal. Por otro lado, el byte 0 de estos bloques puede variar. Su primer bit puede ser* 0 *o* 1 *por razones no conocidas. En cambio, el bit 1 será* 1 *cuando el saldo guardado de la tarjeta sea negativo, y* 0 *en el caso contrario. Cabe destacar que el conteo de bits y bytes se realiza desde el más al menos significativo (izquierda a derecha) y siempre se comienza con el valor 0.*

Habiendo analizado múltiples saldos de tarjetas, y conocida su organización en el sistema, se procedió a la modificación y la prueba en terreno. Revisando aplicaciones que realizaron esta modificación fraudulenta, se comprobó que el cambio que efectúan se centra únicamente en estos dos bloques, el 33 y 34, intercambiándolos directamente por bloques almacenados en la aplicación. A modo de ejemplo, si se quisiera cambiar el saldo a CLP 5000, revisando bloques de otras tarjetas con este saldo, podríamos encontrar que son buenos candidatos:

```
Bloque 33: 88 13 00 00 77 ec ff ff 88 13 00 00 21 de 21 de
Bloque 34: 88 13 00 00 77 ec ff ff 88 13 00 00 22 dd 22 dd
```

Así, los pasos efectuados para probar esta información, se resumen en el experimento 2.

Experimento 2 Modificación de saldo en tarjeta *bip!*

1: Modificar tarjeta *bip!* en los bloques que almacenan los saldos, es decir 33 y 34, con un nuevo monto (por ejemplo, utilizando los descritos anteriormente para CLP 5000).
2: Intentar usar esta tarjeta en algún transporte del sistema.

Las pruebas de este experimento, en general, funcionaron con éxito, aceptando los validadores el uso de tarjetas con saldos modificados, siendo bloqueadas en el sistema posteriormente. Esto no estuvo exento de complicaciones, sobre todo en intentos realizados el segundo semestre del 2015, ya que hubo casos en los cuales fallaron las modificaciones vinculadas a experimentos posteriores, entregando como resultado *Tarjeta no habilitada*, a pesar de haber funcionado de forma perfecta en los primeros periodos de este estudio. A pesar de esto, sí se lograron modificar saldos en este nuevo periodo del año 2015, sabiéndose así que la vulnerabilidad no ha sido solucionada del todo. Los casos con resultado *Tarjeta no habilitada* nunca lograron comprenderse a cabalidad, incluso cuando ocurrieron en el contexto del ataque estudiado en la sección 5.4.5, y sólo se levantaron variadas hipótesis para su interpretación que no pudieron ser posteriormente demostradas.

Finalmente, es importante considerar que, debido a la masividad del ataque efectuado en el 2014, se presume la realización de modificaciones por parte de los encargados del *Transantiago* (sobre todo desde mitad del 2015, con el cambio de TNE en el sistema), ya que previamente, no se registraron problemas de estas características entre la masiva cantidad de usuarios de las aplicaciones *Android* fraudulentas que implementaban este ataque. Esta interpretación es una posible explicación para fallos masivos ocurridos en el sistema, como los descritos en [6], donde en determinada fecha, un gran número de usuarios tuvo problemas con tarjetas *bip!* que previamente funcionaban de forma normal. Esta evidencia es una consecuencia normal

ante cambios importantes efectuados en un sistema que se mantiene operativo siempre; sin espacio a periodos de prueba o de evaluación de este tipo de medidas.

Bloqueo: Cómo se determina cuándo bloquear. Tal como se mencionó previamente, este ataque aparte de pretender probar una vulnerabilidad, buscaba comprender el funcionamiento del sistema, y en particular, cómo funciona el bloqueo de tarjetas en el mismo. Respecto a como éste se realiza, se consideraban posibles dos hipótesis, totalmente contrarias, expuestas a continuación.

Hipótesis 2 *El bloqueo no se efectúa en base a la existencia de diferencias en el saldo de una tarjeta con respecto a cuánto debería ser, sino que, lo hace en referencia a su estado. Es decir, en caso de efectuarse varios viajes sin pago, para los cuales el sistema no tiene una recarga registrada que pueda costear dichos gastos, acumulándose un saldo menor al posible (CLP -720), se procede a agregar dicha tarjeta a la lista negra, para así aplicar posteriormente el bloqueo correspondiente.*

Hipótesis 3 *El bloqueo de tarjetas se realiza revisando los cambios de saldo en cada transacción. En caso de que alguno no sea congruente con lo que se tiene almacenado en el sistema, la tarjeta pasa a lista negra en los validadores, a fin de efectuar el bloqueo en su próximo uso.*

Para poder demostrar alguna de estas suposiciones, se experimentó utilizando tarjetas -con un saldo considerable- capaces de viajar en varias ocasiones, modificándolas y esperando el tiempo apropiado para que se efectuara el bloqueo en el sistema, evitando que ocurra la situación descrita en la hipótesis 2. Esta idea se representa en el experimento 3.

Experimento 3 Modificación de saldo en tarjeta *bip!* buscando su bloqueo

1: Cargar una tarjeta *bip!* con suficiente dinero para viajar una vez y quedar con saldo.
2: Modificar saldo de la tarjeta con uno mayor al que ya poseía.
3: Intentar usarla en algún transporte del sistema.
4: **if** *Tarjeta funciona en el sistema* **then**
5: Esperar 3 días a que se haga efectivo el bloqueo, en caso de ser cierta la hipotesis 3.
6: Volver a intentar usar la tarjeta, analizando los resultados de su uso.

Posterior a la realización del experimento 3, la primera sorpresa encontrada fue que en el portal *bip en línea!*, al usar la tarjeta en cuestión, el saldo modificado se agregaba a los registros del sistema, pasando a ser el saldo oficial de dicha tarjeta en la página *bip! en línea*, tal como muestra la imagen 5.5. La segunda sorpresa, consistió en que el bloqueo no se hizo esperar, inclinándose así el comportamiento del sistema a la realidad expuesta en la hipótesis 3, tal como también se puede ver en la imagen 5.5. De esta forma, tenemos que la idea respecto al bloqueo expuesta en la hipótesis 3, pasa a describir el hecho 5.5.

Hecho 6 *El bloqueo de tarjetas se realiza revisando los cambios de saldo en cada transacción. En caso de que alguno no sea congruente con lo que se tiene almacenado en el sistema, la tarjeta pasa a lista negra en los validadores, a fin de efectuar el bloqueo en su próximo uso.*

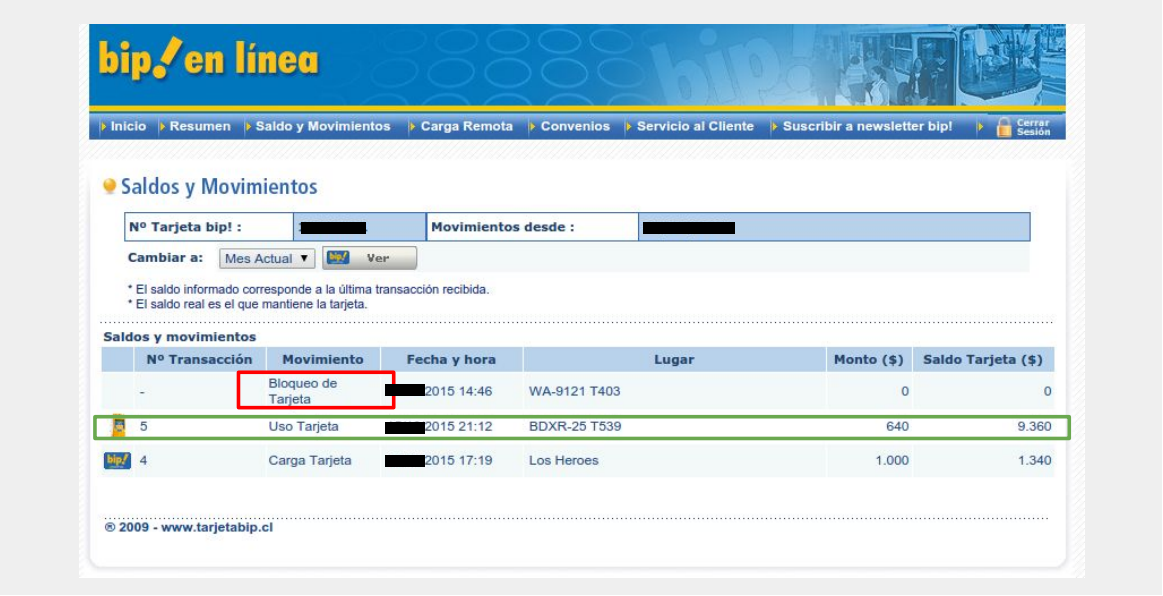

Figura 5.5: Cambio de saldo en tarjeta *bip!* y bloqueo en el portal *bip! en línea*. En verde, puede verse el primer uso con la tarjeta modificada. En rojo, se ve el bloqueo aplicado por parte del sistema. Se cubrieron con negro datos privados irrelevantes.

Es importante destacar con esto no se cierra el análisis de esta parte del sistema, dado que posteriormente se descubrieron realidades que modifican este hecho. Esto será expuesto en el ataque detallado en al sección 5.4.6.

Bloqueo: Cómo se representa en las tarjetas. Una vez que el sistema agrega una tarjeta a su lista negra para ser bloqueada, hace efectivo el bloqueo la próxima vez que dicha tarjeta entra en funcionamiento en cualquier validador del sistema, tal como demostró el experimento 3. Para entender cómo este proceso funciona, se monitorearon los cambios en las tarjetas que habrían de ser bloqueadas luego de efectuadas las modificaciones descritas en los experimentos anteriores, a fin de entender si sólo sucede algún cambio en los datos almacenados en ellas, o el bloqueo se hace efectivo por parte de la lógica del sistema de forma permanente sobre las tarjetas.

Una de las primeras deducciones efectuadas en este ámbito, fue el hecho de que puesto que la mayoría de los validadores del sistema no están en línea con algún tipo de servidor central (con excepción probablemente de los ubicados en las estaciones de *Metro*), para lograr hacer funcionar una medida de este estilo, es necesario cargar en cada validador los nuevos prospectos de tarjetas a anular, cada vez que se agreguen miembros a la lista negra. Este probablemente sea un proceso manual en ciertas áreas, efectuado por los mismos encargados de extraer los datos de los validadores que no funcionan en lugares con extracción automática de datos, traduciéndose así en un proceso lento, poco práctico y riesgoso.

Además, mantener un medida de esta magnitud en los validadores requiere espacio debido a la cantidad de nombres posibles de aparecer en dichas listas negras, y procesamiento, para poder manejar esta estructura de datos de una forma eficiente, a fin de no interrumpir el uso

normal del usuario. Esto se traduce finalmente en una decisión complicada al momento de desarrollar una plataforma de este calibre, entre la seguridad deseada para el sistema y los costos monetarios que significan este tipo de implementaciones.

Para clarificar este proceso, se realizaron las siguientes actividades, resumidas en el experimento 4.

Experimento 4 Modificación de saldo en tarjeta *bip!* buscando su bloqueo

1: Modificar dos tarjetas, *tarjeta1* y *tarjeta2*, tal como sugiere el experimento 3, para que sean bloqueadas en el sistema.
2: Utilizarlas y esperar los 3 días correspondientes.
3: Utilizar *tarjeta1* en un metro, a fin de que sea bloqueada.
4: Utilizar *tarjeta2* en un bus, a fin de que sea bloqueada.
5: Analizar sus cambios y ver la posibilidad de revertirlos.

Respecto a la ejecución del bloqueo, este análisis obtuvo distintos resultados para los 2 tipos distintos de validadores. Para los validadores que se encuentran en el metro, se obtuvieron cambios mínimos en los bloques 17 y 18, cambiando el 6^{to} bit del primer byte, de 0 a 1, además de sus *secuencias de validación* almacenadas en sus bytes 15^{vos} (de estas *secuencias de validación* se hablará más en el próximo ataque). Dado este pequeño cambio, se decidió probar la tarjeta revirtiéndolo, es decir, volviendo a escribir las mismas líneas previas al bloqueo, tal como se detalla en el experimento 5.

Experimento 5 Revertir bloqueo de tarjeta efectuado en el *Metro*

1: Obtener *tarjeta1* del experimento 4, luego del paso 3.
2: Volver tarjeta al estado anterior, revirtiendo los cambios efectuados por el validador.
3: Utilizar *tarjeta1* en un metro y analizar el resultado.

La realización del experimento 5 permite medir que tan persistentes son los bloqueos efectuados en el sistema, es decir, si lo regula principalmente el *backend* del sistema, o una interacción entre los datos de la tarjeta y el *validador* en cuestión. En particular, permitiría evadir el bloqueo en este segundo caso, dado que ocurriría un cambio en la tarjeta, y posteriormente podría borrarse de la lista negra, suponiendo que nadie podría volver a modificarla. Esto no fue así, ya que al realizarse la prueba, los cambios descritos anteriormente volvieron a ocurrir en la tarjeta en cuestión, independiente de los cambios efectuados en ella.

Para los validadores en los buses, los cambios fueron mucho más significativos. Al validar y revelarse el mensaje de *Tarjeta bloqueada*, los sectores 4 y 9 de la tarjeta en cuestión se vaciaron completamente, cambiando además las llaves A y B del sector y sus condiciones de acceso (AC). Al estudiar estos nuevos parámetros para acceder a los sectores 4 y 9, se logró entender que el cambio efectuado por el sistema sobre la tarjeta es irreversible, incluso encontrando las nuevas claves A y B de dichos bloques (`a0a1a2a3a4a5` y `b0b1b2b3b4b5` respectivamente), ya que por las nuevas AC todo cambio queda absolutamente bloqueado en dichos sectores. De esta forma, el bloqueo efectuado por los buses fue mucho más fuerte que el realizado por el *Metro*, quedando las tarjetas de estos experimentos absolutamente inutilizables a futuro, tanto para este sistema, como para cualquiera que necesite utilizar los sectores en cuestión. La imagen 5.6 representa las diferencias ocurridas entre ambos bloqueos.

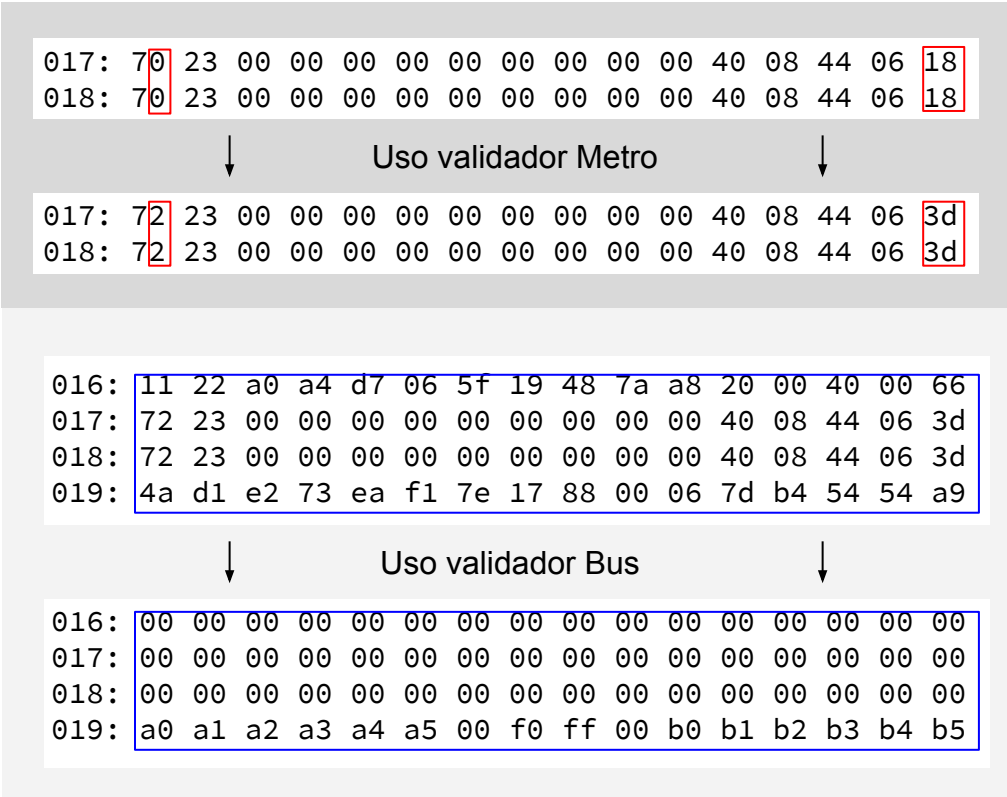

Figura 5.6: Comparación tipos de bloqueos en tarjetas *bip!*. Arriba, se ve el bloqueo efectuado en un validador del *Metro*; abajo, por otro lado, se muestra el bloqueo que realiza un validador de bus. Esto también se repite para el sector 9, es decir, de los bloques 36, 37, 38 y 39.

Aparte de estas pruebas, se procuró determinar si el bloqueo ocurría mediante el ID del *Transantiago* o mediante el UID de la tarjeta. Para poder esclarecer esta duda se realizó el experimento 6.

Experimento 6 Determinar identificador de bloqueo

1: Modificar y usar tarjeta, tal como se realiza en el experimento 3.
2: Esperar los 3 días correspondientes para que la tarjeta sea agregada a lista negra.
3: Modificar ID del Transantiago en la tarjeta por otro ID de alguna tarjeta válida.
4: Utilizar tarjeta por bloquear y analizar resultados.

Al realizarse el experimento 6, el bloqueo se obtuvo de todas formas, independiente del ID del sistema grabado en la tarjeta. Esto simplemente ratificó la información previamente planteada respecto a las características del sistema, resumida en el hecho 4, el cual plantea que la identificación final de las tarjetas en el sistema es el UID de cada una, y no el ID que el mismo *Transantiago* le entrega.

De esta manera, la información aprendida en este grupo de experimentos puede reducirse en el siguiente hecho.

Hecho 7 *La ejecución del bloqueo sobre las tarjetas modificadas es distinta dependiendo del validador que las aplica. Si el validador es uno de los torniquetes del Metro, este sólo modificará el sexto bit del primer byte de los bloques 16 y 17, más sus bytes de verificación. Esto puede revertirse en la tarjeta, pero para el sistema la tarjeta seguirá estando bloqueada, repitiendo estos cambios cada vez que se intente reutilizar. Por otro lado, en el caso de que el validador que aplica el bloqueo haya sido un validador de bus, el bloqueo efectuado es más fuerte, ya que borra el contenido de los sectores 4 y 9, además de cambiar sus claves A y B, y más importante aún, sus condiciones de acceso, dejándolos bloqueados para cualquier tipo de escritura futura. Este bloqueo es irreversible sobre la tarjeta, perdiéndose toda posibilidad de reutilización de la misma en el sistema. Por último, los bloqueos se realizan guiándose por el UID de las tarjetas, y no el ID de las mismas en el Transantiago.*

Riesgo

El riesgo de este ataque es **alto**. Esto se debe principalmente a que el ataque es simple, pues no requiere mayor conocimiento técnico de qué está efectuando, sino que sólo ciertas nociones respecto al cómo se almacena la información en la tarjeta. Además, las contramedidas para el atacante son prácticamente nulas, dado que a parte de ser un uso anónimo, el bloqueo ocurre tiempo después de cometido el ataque, con lo cual no es vinculable a la persona que está ocupando la tarjeta en el momento del bloqueo mismo. Por último, el ataque es rentable, ya que dados los costos de las tarjetas, muchas veces el uso reiterado de una tarjeta modificada y la compra de otra nueva una vez efectuado el bloqueo es más conveniente que pagar los pasajes a utilizar en el periodo necesario antes de que se aplique el bloqueo.

Como ejemplo, basta analizar la situación de lo ocurrido el 2014, para comprender que el poder modificar el saldo de una tarjeta, de forma tan simple, puede detonar en un costo millonario para el sistema. En dicha ocasión, más de 30 mil tarjetas se vieron comprometidas, por lo cual, si consideramos que todas viajaron al menos una vez evadiendo un pasaje cercano a los CLP 600, existieron al menos perdidas de CLP 180 millones para el sistema. Aunque no se cuenta con cifras del porcentaje de usuarios con tarjetas comprometidas en la actualidad, se presume que se sigue efectuando este ataque de manera regular, lógicamente con bastante menor publicidad que en el período de su descubrimiento y enfrentando distintas nuevas medidas de contención, como las analizadas posteriormente en la sección 5.4.1.

Mitigación

La principal problemática en este ataque es la confianza que se tiene, desde el diseño del sistema mismo, en la seguridad que entregan las tarjetas MIFARE Classic por sí mismas. Dado que se conocían, o al menos se estaban iniciado estudios referente a las primeras vulnerabilidades en éstas en un periodo muy cercano al en que se estaba implementando el sistema de transporte, sorprende el hecho de que la confianza se mantuviera hasta casi 7 años después, sin implementar algún tipo de medida de seguridad que impidiera un caos tal como el que se vivió con la proliferación de aplicaciones fraudulentas el año 2014. De esta forma, una de las principales mitigaciones que se proponen, para poder continuar utilizando el sistema de

forma similar a la actual, es el cambio de modelo de tarjeta a otra de funcionamiento similar, pero que utilice primitivas criptográficas reales como la Mifare Plus[8], que utiliza `AES-128` para sus transacciones de autentificación e integridad.

Cómo segunda medida y en caso de querer mantener el sistema tal cual está (ya sea por costos o distintas dificultades de cambio), dado que se utilizan los *bloques de valor* para el manejo de los saldos, y éstos no tienen mayores medidas de seguridad, se recomienda agregar algún cálculo de integridad en la tarjeta que incluya como uno de sus parámetros; el saldo de la misma. Tal como se explica en 6.1, lo ideal sería poder tener algún tipo de *protocolo criptográfico* (tipo MAC o *Message Authentication Code*[9]) con una clave para el sistema que guarde un resumen de la información de la tarjeta y lo actualice cada vez que se efectúe alguna transacción. Así, se comprueba este resumen previo a cualquier intento de uso, y si no es correcto, se sabe que alguna modificación indeseable ocurrió en la tarjeta.

5.4.2. Obtención de viaje gratis simulando viaje previo en tarjetas *bip!*

Vulnerabilidad

Se ha concluido de los distintos análisis efectuados, en particular del hecho 3, que las tarjetas *bip!* tienen información importante para poder comunicarle al validador qué debiese ocurrir en su encuentro. Esto se debe principalmente a que los validadores no tienen acceso a una fuente de información central que les indique qué cosas suceden en cada momento y en qué lugar del sistema. De esta manera, al realizarse un transbordo la tarjeta le comunica al validador hace cuánto tiempo viajó y en qué recorrido lo hizo, pudiendo éste determinar cuánto dinero debe cobrar en dicho instante por el nuevo viaje. Así, conociéndose cuáles son los distintos campos en las tarjetas que contienen esta información, y sabiendo de qué manera modificarlos correctamente, se presume posible engañar a un validador haciéndole creer que un nuevo viaje no debiese ser cobrado, ya que simplemente debe considerarse como un transbordo y no un primer viaje.

Ataque

Simular en determinada tarjeta que el viaje a realizar es un transbordo, y no el primer viaje dentro de las dos últimas horas, con el fin de obtener una tarifa reducida al momento de la validación.

[8] `http://www.nxp.com/documents/short_data_sheet/MF1SPLUSX0Y1_SDS.pdf`
[9] `https://en.wikipedia.org/wiki/Message_authentication_code`

Experimentos asociados

Byte de verificación. Una diferencia importante de los datos a modificar de las tarjetas en éste y los próximos ataques, con respecto al ataque detallado en 5.4.1, es que este último trabaja sobre información almacenada en *bloques de valor*, mientras que los futuros consideran información almacenada en *bloques de datos*. Tal como se detalló en la sección 2.2.2, los *bloques de datos* pueden almacenar información a libre disposición del sistema, por lo cual, su modificación requiere entender el formato en el cuál se encuentran y su significado en el sistema mismo.

Efectuados diversos análisis tales como los mencionados en la sección 4.4.3, buscando comprender datos almacenados vinculados a información relevante para este ataque, se descubrió una medida importante de seguridad considerada por el sistema del *Transantiago*, que pretende defender la integridad de la información almacenada en las tarjetas *bip!* en sus *bloques de datos*. Esta medida consiste en el uso del último byte de cada *bloque de datos* almacenado en las tarjetas, el cual llamaremos *byte de verificación*, que sabemos es un valor calculado dependiente de la información almacenada en los demás 15 bytes del bloque en cuestión. Como ejemplo de esto, está el caso del bloque de datos más común encontrado en las tarjetas, el cual corresponde a un bloque vacío de información. Éste tiene siempre como *byte de verificación* `f5`, con lo cual, la representación hexadecimal del bloque completo sería:

`00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5`.

De esta manera, podemos resumir esta idea en el hecho 8, el cual fue demostrado empíricamente.

Hecho 8 *Los bloques de datos de las tarjetas del Transantiago poseen un byte de verificación a fin de poder cerciorar al momento de utilizar sus datos, la integridad de ellos en la tarjeta. Este byte está siempre almacenado al final del bloque en cuestión, en el byte 15, y sólo depende de los demás datos almacenados en el bloque, correspondientes a los primeros 15 bytes.*

Cabe destacar que, tal como se analizó en 4.4.2, estos *byte de verificación* sólo se encuentran en los *bloques de datos* que el sistema utiliza, es decir, para el caso de las TNE basadas en tarjetas MIFARE Classic de 4 kB, sólo se pueden encontrar entre sus primeros 64 bloques (1 kB). En el espacio restante, el que no es utilizado en el sistema, se encuentran todos sus bytes absolutamente vacíos, es decir, sin bytes de verificación.

Tal como se menciona en la sección 6.2.1, se realizaron distintos intentos de descubrir el cómo calcular esta fórmula de verificación, lo cual lamentablemente no entregó resultados fructíferos en este trabajo. A pesar de esto, se continuaron los experimentos con el fin de lograr vulnerar las medidas de seguridad implementadas para el sistema.

Análisis de bloque utilizados al viajar. El primer paso en esta búsqueda de vulnerabilidades en el cobro de los viajes y su distinción, fue lograr comprender cómo varía la información en las tarjetas luego de cada viaje y qué significa cada cambio. A partir de lectu-

ras de múltiples tarjetas en distintos estados, y siguiendo sus cambios en periodos de tiempo convenientes, se pudo determinar varias partes de la información expuesta en las figuras 4.2 y 4.3.

De esta forma, luego de analizar los datos recolectados, se determinó que los bloques importantes a manipular en cada experimento son el número 16, el par 17 y 18 (idénticos), el par 21 y 22 (idénticos), y alguno de la tríada 44, 45 y 46, ya que todos estos bloques son los que precisamente se modifican luego de efectuarse un viaje con alguna tarjeta.

1. El bloque 16 almacena datos referente al último uso de la tarjeta, sea un viaje o una recarga, correspondientes al día y mes, entre sus bytes 10 y 11.
2. Los bloques 17 y 18, almacenan el número de uso en la tarjeta, el cual siempre es ascendente y coincide con el indicado por el portal *bip! en línea*. También almacena en qué bloque del registro del uso se escribirá la información del viaje efectuado (entre los bloques 44, 45 y 46). Además, existe información que no se logró entender completamente, pero se sabe que posee una relación al número en específico del servicio utilizado, por ejemplo, del bus en cuestión o la estación del Metro en la que se usó la tarjeta.
3. Los bloques 21 y 22 guardan la fecha del último viaje de la tarjeta, considerando desde los segundos hasta el año para su precisión, en horario $UTC+0$. Esto se presume necesario para trabajar con los validadores, es decir, que el reloj interno de estos últimos no está coordinado con la hora de Chile ($UTC-3$) sino que se mantiene en $UTC+0$ por defecto. Además, guardan el tipo de viaje utilizado en una secuencia de dos bits, pudiendo diferenciarse así entre usos de metro o micro.
4. Por último, los bloques 44, 45 y 46 guardan registro de los últimos 3 viajes efectuados con la tarjeta, almacenando la fecha de igual forma que los bloques 21 y 22 pero en formato $UTC-3$. Además, poseen el cobro efectuado por dicho viaje. En paralelo, los bloques 40, 41 y 42 almacenan información similar respecto a las últimas 3 recargas efectuadas en la tarjeta.

En los experimentos descritos a continuación se relata cómo se experimentó con esta información, con el fin de determinar la posibilidad de vulnerar el mecanismo utilizado para determinar el cobro por parte de los validadores, buscando sortear las barreras de seguridad desarrolladas para el sistema.

Ignorando bytes de verificación El primer experimento en el cual se probó el conocimiento descubierto en el punto anterior, consistió en intentar modificar la información respecto a la hora y fecha en la cual se realizó el último viaje en determinada tarjeta, probándola luego en algún transporte distinto al último utilizado, intentando conseguir con esto, un viaje gratis. Dado que no se conoce el método con el cual se calculan los *bytes de verificación* de los *bloques de datos*, simplemente se ignoraron, dejando exactamente el mismo que estaba antes en el bloque a modificar. Así, se puede describir el procedimiento realizado con los distintos pasos enumerados en el experimento 7.

Experimento 7 Ignorando bytes de verificación.

1: Modificar tarjeta en los bloques que involucran información de la hora y fecha del último viaje, es decir, bloques 16, 21, 22 y alguno entre 44, 45 y 46, con la información temporal del momento en que se está modificando. Obviar bytes de verificación en este proceso.
2: Probar tarjeta en algún medio de transporte, en menos de dos horas de la escritura efectuada en la tarjeta.

Al realizar el experimento 7, no se obtuvo ningún resultado significativo en el uso de la tarjeta modificada con los validadores. De hecho, ni siquiera se obtuvo una respuesta, ya que estos ignoraban la tarjeta modificada completamente. A partir de este punto, se presentaron múltiples hipótesis sobre cuál era la lógica del sistema en estos casos, pero finalmente, los experimentos que dieron éxito para este ataque, se reducen en los casos expuestos a continuación.

Quebrando las medidas de seguridad. Un problema grave que posee la medida de seguridad del *byte de verificación* por bloque, es el hecho de que considera sólo la información del bloque para calcularse, junto a probablemente una entrada como clave grabada en el sistema mismo. Puesto que no ocupa más información extraída de la tarjeta, el reemplazar un bloque completo por otro que obtenido de otra tarjeta válida, genera un estado en la primera igual de válido que el estado inicial, dado que la validación de su nuevo bloque está probada por pertenecer a la segunda tarjeta. Esto es un riesgo alto de diseño, y es exactamente lo que se utilizará en el experimento 8 para concretar la simulación de transbordo en una tarjeta.

Experimento 8 Copiando bloques completos.

1: Contar con dos tarjetas: *tarjeta1* y *tarjeta2*.
2: Realizar un viaje válido en *tarjeta1*.
3: Copiar la información de su viaje, es decir, los bloques 16, 17, 18, 21, 22 y el bloque correspondiente del trío 44, 45 y 46, y pegarlos en la *tarjeta2*.
4: Utilizar *tarjeta2* en algún medio de transporte antes de 2 horas del viaje hecho con *tarjeta1* y analizar los resultados obtenidos.

Una vez realizado el experimento 8, se obtuvo de forma exitosa un transbordo simulado con la *tarjeta2*, lográndose así un viaje con tarifa reducida en una tarjeta *bip!*, que no había viajado en las últimas 2 horas. Así, se probó la posibilidad efectiva de explotar esta vulnerabilidad, estableciéndose una nueva directiva de ataque hasta ahora desconocida.

Originalmente éste no fue el primer experimento, ya que previamente se suponía que los bloques 17 y 18 no contenían información importante respecto al último viaje efectuado. Así, se intentó este ataque en reiteradas ocasiones con distintas combinaciones de bloques,intentando así obtener un viaje gratis.

Luego de probar la validez del caso presentado en el experimento de 8, fue importante revisar en la plataforma *bip! en línea* el estado de la tarjeta utilizada pasado un proceso de *clearing*, es decir, unos 3 días después, para así saber si no existía algún tipo de medida o bloqueo que se detonara al realizar este ataque. El paso del tiempo demostró que nada sucedió

con las tarjetas en cuestión, no teniendo así el sistema forma implementada de detectar un ataque como el aquí presentado. En la figura 5.7, se muestra la prueba del ataque en la plataforma *bip! en línea*.

Figura 5.7: Prueba de simulación de transbordo en *bip! en línea*. De rojo, el último uso previo al ataque ocurrido el día anterior. Con cobro 0 se encuentra el ataque de un transbordo sin actividad en las 2 horas anteriores. En verde, destaca el desplazamiento de número de operación. Se omitieron en negro datos privados irrelevantes.

Para continuar mejorando este ataque, se procuró determinar el conjunto mínimo de bloques necesarios para su ejecución. Así, se realizó una seguidilla de experimentos, con los que finalmente logró determinarse que de los bloques utilizados en el experimento 8, el trío 44, 45 y 46 no son relevantes en el momento de la validación, sino que simplemente mantienen un registro en la tarjeta, del cual no se consulta información en ninguna parte del proceso. De esta forma, el ataque completo y sus consideraciones quedan descritas en el hecho 9.

Hecho 9 *En los bloques 16, 17, 18, 21 y 22 está la información necesaria para que una tarjeta le indique a un validador cuándo fue la última vez que viajó y en que medio, y por ende, si debe cobrarle o no un pasaje. Copiando estos bloques de una tarjeta que acaba de viajar a otras, y utilizándolas en menos de 2 horas desde ejecutado el viaje, es posible simular transbordos que nunca ocurrieron como tales, accediéndose a descuentos de pasaje en tarjetas que no deberían obtenerlos. En este ataque, el sistema asume este tipo de uso como un caso normal, y no aplica ningún tipo de bloqueo ni restricción a las tarjetas comprometidas.*

Por último, una consecuencia obtenida del estudio de este ataque fue el hecho de que el número de uso de una tarjeta, almacenado en los bloques 17 y 18, no tiene mayor uso en el sistema, o al menos, no existen bloqueos ni medidas a partir de comportamientos extraños que tenga este valor en el historial de una tarjeta.

Inicialmente, cuando no se conocía este último detalle, el ataque aquí descrito fue probado exclusivamente con tarjetas coordinadas en la cantidad de usos, a fin de poder mantener los cambios indetectables para el sistema en caso de que significara algún problema tener incongruencias en el número de usos. Posteriormente, este detalle se descuidó, y tal como muestra la figura 5.7 en verde, esto no significó mayor problema en el sistema, quien simplemente asume que esta información es correcta, desplegándola incluso en el portal *bip! en línea* como la

información oficial. Esto facilita mucho los experimentos, y permite utilizar la información de cualquier tarjeta con otra sin preocupaciones, lo que le entrega mayor versatilidad al ataque e importantes posiblidades de masificación. Es importante notar que estos valores pueden no tan sólo aumentar, sino que también disminuir y repetirse, todo sin haber problemas en el sistema. Esta información la describiremos en el hecho 10, el último de esta sección.

Hecho 10 *El número de uso de una tarjeta, almacenado en los bloques 17 y 18 y visible en el portal bip! en línea, no son usados en ninguna medida de seguridad para mantener el flujo correcto de las tarjetas. Esto permite que la modificación de estos bloques en cuestión se realice sin mayores cuidados, y que comportamientos bastante anómalos se puedan representar en el flujo de las tarjetas bip!, sin suponer mayores problemas.*

Riesgo

El riesgo de este ataque es **alto**. A pesar de que es necesario contar con un viaje previo y utilizar sus datos dentro de una ventana no mayor a 2 horas desde que se efectuó, el ataque es expansible fácilmente a un amplio número de usuarios, para el que actualmente no existen medidas que detengan ni penalicen su aplicación.

Similar a lo ocurrido con el ataque detallado en 5.4.1 el año 2014, podría diseñarse una aplicación *Android* en la cual, usuarios dispuestos a "colaborar" voluntariamente con otros, entreguen los bloques importantes para este ataque al efectuar su último viaje, a fin de que otros usuarios descarguen dicha información y simplemente la apliquen en su tarjeta. Con esto, obtendrían hasta 3 viajes gratis dentro de las 2 horas que se permiten *transbordos*, y dado que este estado es nuevo en cada tarjeta, no hay problemas en el uso paralelo en varias tarjetas del mismo caso. A partir de esto, sin mayores problemas, un alto número de usuarios podría beneficiarse de cada usuario "colaborador", abriéndose una opción de ataque masivo y múltiples pérdidas con cada viaje compartido que se realice. Cabe destacar que esta aplicación podría ser muy simple y fácil de usar, de modo que cualquier persona con un dispositivo *Android* con NFC pueda ocuparla, sin tener mayor conocimiento de qué es lo que realmente se está realizando en su dispositivo móvil o en su tarjeta, a fin de agregarle versatilidad y simpleza al ataque, aumentando el riesgo para el sistema.

Mitigación

La principal propuesta para solucionar la vulnerabilidad descrita en esta sección, es la implementación de un sistema de seguimiento del flujo de una tarjeta similar al que probablemente existe para con los saldos, pero esta vez con la lógica centrada en el comportamiento de los cobros o viajes que una tarjeta correctamente utilizada debería tener. Probablemente, flujos de uso de tarjetas *bip!* normales en las cuales un viaje gratis o de CLP 20 aparecen sin tener antes de 2 horas un viaje previo, han sido adulteradas, siendo claras candidatas a bloqueo. Para efectuar este último paso, se propone la utilización del mismo sistema usado en el bloqueo de tarjetas por modificación de saldos, es decir, utilizando sistemas de listas negras (analizado en el experimento en 5.4.1), a fin de obtener los candidatos a bloqueo posterior al

periodo del *clearing* correspondiente a la fecha de efectuado el ataque.

Siendo más estricto, incluso se podrían monitorear comportamientos anómalos en la variable que determina el número de uso en la tarjeta ubicada en los bloques 17 y 18, ya que en caso de querer realizar este ataque, el imponer la condición de que los usos deben estar coordinados entre la tarjeta proveedora de información y la modificada, reduciría su posibilidad de realización de forma importante. Esto sólo es una mejora si consideramos que el atacante no tiene idea de cómo generar los *bytes de verificación* en los distintos bloques modificados, por lo cual se ve obligado a copiar información de otras tarjetas, tal como sucede en el caso de este trabajo. Esta es una suposición relativamente fuerte, ya que a pesar de los distintos procesos que se realizaron en este trabajo para obtener dicha fórmula y que fracasaron, expuestos en la sección 6.2.1, existen razones de peso para asegurar que 1 *byte* como medida de protección para la información de cada bloque, calculado únicamente con la información en el mismo y de forma tan determinista, no es una medida de seguridad robusta[10]. Basta que se realice un estudio acabado del cómo se realizan estos procesos y se encuentre dicha fórmula, para darle a todo este ataque mayores facilidades de realización y absoluta personalización.

Cómo último detalle, destacar que el principal problema ya mencionado con respecto a los *byte de verificación*, es el hecho de que sirven para comprobar exclusivamente la integridad de los datos de un bloque, los que pueden terminar siendo reemplazados de forma completa, como se mostró en este ataque. Estos cobros mantienen la tarjeta *bip!* en cuestión válida, ya que dicho bloque también es válido en otra tarjeta. Este es un problema principalmente de diseño, ya que si lo importante era mantener la integridad de los datos de la tarjeta, entonces la medida de verificación debía considerar información de toda su información, y no únicamente de bloques particulares de forma independiente. Esta idea se explica de forma más profunda, en la sección 6.1.

5.4.3. Negación de servicio a tarjeta *bip!* cercana

Vulnerabilidad

Dado que se conocen las claves de todos los sectores de una tarjeta *bip!* como se mencionó en el hecho 1, es posible modificar cualquier parte de su información, tal como se realizó en el análisis de la vulnerabilidad anterior. Además, analizada la importancia que tienen en la validación de los bloques de datos los *bytes de verificación*, tanto en la sección 5.4.2 como en múltiples experimentos posteriores, se comprendió que en caso de que alguna de estas validaciones sean incorrectas, la tarjeta dejará de ser reconocida como válida por el sistema, quedando inhabilitada para su futuro uso.

De esta manera, combinando ambos conocimientos, obtenemos un ataque simple de negación del uso de una tarjeta, invalidándola al modificar algún byte de su contenido por alguno distinto. Este ataque es de corto alcance, pero totalmente rápido y eficiente, tal como se

[10]Como ejemplo de esta aseveración, tenemos el caso de la función de *Hash* MD5, de largo 128 bits, la cual ya ha sido vulnerada en distintos estudios. *Fuentes*: https://en.wikipedia.org/wiki/MD5 y [41]

detallará en las siguientes secciones.

Ataque

Invalidar una tarjeta *bip!* cercana mediante la modificación apropiada de su información.

Experimentos asociados

Para el análisis de esta vulnerabilidad no se realizaron experimentos especializados, ya que realizados todos los demás experimentos de este trabajo, se ha logrado comprender que cualquier cambio que el sistema no comprenda en alguna tarjeta, lo reconoce como un problema y entrega el mensaje *Tarjeta no habilitada* en sus validadores. Como ejemplo de estos casos, se encuentran los resultados de los experimentos 11, 12, entre otros.

Riesgo

Este ataque ha sido evaluado de **alto** riesgo. Entre los factores que influyen en esta evaluación, es el hecho de que se puede desarrollar sin mayor dificultad una aplicación *Android* que al detectar una tarjeta, modifique algún bloque por otro que se sepa incorrecto, invalidando así completamente la tarjeta. Como ejemplo simple, y dado que sabemos que el bloque de información con 15 bytes vacíos tiene de *byte de verificación* los caracteres `F5`, basta con modificar un bloque frecuentemente utilizado -como los bloques 16, 17 o 18- por uno completamente vacío, con cualquier *byte de verificación* distinto a `F5`.

Un ataque implementado de esta forma, se transforma en un arma simple que ataca directamente a los usuarios del sistema, y aunque la escala del ataque es menor (dependiente de la difusión y disponibilidad de los adversarios a usar esta aplicación), sus consecuencias son importantes sobre la gente afectada y sin posibles repercusiones para el atacante. Así, la severidad del ataque radica en su potencialidad para causar perdida de confianza en el sistema, especialmente considerando que la detección de los posibles atacantes es prácticamente nula. La escritura de las tarjetas funciona de manera muy veloz, por lo cual no se necesita más que cercanía a la víctima en algún espacio, cosa que frecuentemente sucede en los mismos transportes públicos.

Mitigación

La eliminación de esta vulnerabilidad es bastante compleja, dado que este ataque utiliza el poder que entrega tener las claves de los distintos sectores de cada tarjeta *bip!* para editar su información, más las verificaciones que se realizan respecto a los datos almacenados utilizando los *bytes de verificación*.

La instauración de claves por sector y la forma en que ésto se implementa no presenta reales problemas, sigue un estándar correcto y funcionaría de forma perfecta si el sistema de cifrado fuese seguro, pudiendo mantener así secretas las claves de sus sectores. Por otro lado, los *bytes de verificación* son utilizados para evitar una libre modificación de la tarjeta por parte de los usuarios comunes, lo cual difícilmente podría ser eliminado actualmente sin correr riesgos mayores.

De esta manera, la única mitigación posible es evitar que los usuarios comunes puedan modificar la tarjeta de forma directa, para lo cual, es necesario cambiar de tecnología a una tarjeta con un cifrado más estricto que utilice alguna primitiva criptográfica probada, tal como se presentó anteriormente con la tarjeta *MIFARE Plus*. De esta forma, las claves del sistema serían prácticamente imposibles de recuperar y medidas de contención como los *bytes de verificación* no serían necesarios, cerrándose con ellos este tipo de posibilidades de inhabilitación. Es importante destacar que en general, cualquier medida de validación que se calcule a partir de información almacenada en la tarjeta, terminará permitiendo la realización de este ataque, ya que genera la existencia de estados válidos e inválidos para tarjetas en el sistema.

5.4.4. Simulación de TNE en tarjeta *bip!*

Vulnerabilidad

Tal y como se planteó en el ataque anterior, hay información importante en las tarjetas del *Transantiago* que determinan la acción a seguir por parte de los validadores en cada uso. Una distinción que se realiza entre ellas en cada transacción, es el tipo de tarjeta utilizada, ya que esto determinará el costo que el validador debe cobrar en algún uso. En el caso de las TNE, el validador interpreta a partir de la información en la tarjeta en uso que el usuario puede acceder a la tarifa rebajada escolar, por lo que le cobra un saldo inferior al que paga cualquier usuario normal. Dado que es la tarjeta la parte del sistema que contiene esta información, se presume posible cambiarla para lograr pasar una tarjeta *bip!* al portador como una TNE. Esta vulnerabilidad es la que se intentará explotar en la presente sección.

Ataque

Hacer pasar una tarjeta *bip!* al portador como una TNE frente a los validadores del sistema, a fin de poder acceder a la tarifa rebajada para escolares, sin realmente merecerlo.

Experimentos asociados

Identificando diferencias con otros tipos de tarjetas. El primer paso realizado buscando explotar la vulnerabilidad previamente descrita, fue el analizar múltiples lecturas de distintas TNE y compararlas con los distintos tipos de tarjetas en el *Transantiago*, intentando

determinar diferencias importantes entre cada tipo de tarjeta que ayudaran a clarificar cómo detonar este ataque. Se creía que, dado que los funcionamientos en el sistema son considerablemente distintos entre los diferentes tipos de tarjetas, la información que volvía especiales a las TNE estaba en bloques que contenían información notoriamente distinta a los de sus contrapartes, por ejemplo, en las tarjetas *bip!* normales.

Los bloques con información considerablemente distinta que se utilizaban en las TNE y no en tarjetas de otro tipo, eran los bloques número 1, 12, 13, 14, 20, 28 y 37. Estos fueron analizados por separado, a fin de determinar si alguno entregaba indicios de poseer la información que lograba distinguir el tipo de tarjeta que se estaba utilizando y el cobro a efectuar. Cabe destacar que para este análisis se utilizaron tanto lecturas de TNE habilitadas desde el año 2015, como de TNE generadas de forma previa a este periodo.

1. El bloque 1 contiene el ID de la tarjeta en el sistema, y contiene información recurrentemente distinta, debido a que pareciera ser que existen rangos de IDs posibles para los tipos de tarjetas en el *Transantiago* . Por ejemplo, hasta la fecha de este trabajo, para tarjetas *bip!* al portador se han encontrado tarjetas con IDs desde los `10000000` hasta los `20000000` aproximadamente, TNE escolares desde los `60000000` a los `70000000`, y TNE de educación superior, desde los `70000000` en adelante.
2. Los bloques 12 y 13 contienen información referente al dueño de la TNE, almacenando el nombre y el RUN, respectivamente. Las demás tarjetas, tal como se vio en el ataque de la sección 5.3.1, no contienen información en estos bloques.
3. El bloque 14 es siempre similar entre distintas TNE, y se presume tiene relación con el año en el cual dicha tarjeta entró en circulación. De igual manera, los bloques 14 de los demás tipos de tarjetas en el sistema son similares entre los de determinado tipo. Por ejemplo, en el caso de las tarjetas *bip!* al portador, todas tienen como bloque 14 la secuencia: `01 e1 cc e7 09 00 00 00 00 00 00 00 00 00 00 62`.
4. El bloque 20 está vacío en todos los tipos de tarjetas, excepto en las TNE. Este bloque es distinto entre distintos tipos de TNE, pero idéntico para los tipos similares. Por ejemplo, es idéntico entre las TNE de educación superior válidas desde el 2015, también es idéntico entre las TNE de educación básica, y por último, levemente distinto entre las TNE más antiguas.
5. Los bloques número 28 de varias TNE antiguas (es decir, previas a las actualizadas el año 2015) tienen información variada, pero desde el 2015, al igual que las tarjetas *bip!* en general, no tienen información alguna.
6. Por último, el bloque 37 tiene dos partes importantes a analizar. Los primeros 8 bytes, contienen información similar en cada tarjeta de determinado tipo, por ejemplo, sólo ceros en las *bip!* al portador, o la secuencia `9e 17 00 30 01 00 00 00` para el caso de las nuevas TNE. Los demás 7 bytes, cambian de variadas formas en los distintos tipos y de forma periódica, lo cual luego de múltiples análisis, resultó ser información referente a la última recarga efectuada en la tarjeta, principalmente relacionada al lugar en el cual se cursó.

En general, para poder determinar si estos bloques tienen la información relevante para conseguir la simulación de una TNE en una tarjeta *bip!* al portador, se realizaron los pasos descritos en el experimento 9.

Experimento 9 Primer intento TNE con bloques distintos

1: Obtener los bloques 1, 12, 13, 14, 20, 28 y 37 del último estado de una TNE válida.
2: Modificar una tarjeta *bip!* al portador con estos bloques.
3: Utilizar la tarjeta modificada en algún medio de transporte y analizar los resultados obtenidos.

La ejecución del experimento 9 no entregó resultados positivos, y aunque variantes de él fueron probadas de forma previa con subconjuntos de los cambios mencionados en el experimento 9, estos también entregaron resultados negativos, manteniéndose el costo tradicional en los intentos realizados. De este punto, se comprendió que probablemente no había información suficiente en las modificaciones efectuadas a la tarjeta para lograr conseguir el viaje con tarifa reducida, y que quizás, la revisión debería ser más profunda, a nivel de bits diferentes de forma constante, y no solamente de bytes completos, como se revisó hasta este punto.

Identificando diferencias a nivel de bits. Para poder determinar cuáles bloques poseían la información buscada, se procedió a analizar todos los bits que se mantenían en las TNE de determinada manera y variaban en otros tipos de tarjetas. Para esto, se crearon pequeños programas en *Java* que analizaban los archivos de texto donde se almacenaban las lecturas de tarjetas, determinando similitudes entre todas las de igual tipo y contrastando estas similitudes por clase de tarjeta, encontrando así diferencias importantes de las cuales continuar el análisis de este ataque.

Estos procedimientos arrojaron diferencias en las TNE con respecto a las otras tarjetas en los bloques 0,1, 12, 14, 16, 17, 18, 20 y 37. De estos, el bloque 0 es inmodificable, por lo cual, aunque fuera relevante para la determinación del saldo a pagar por las TNE, no podría ser considerado como una posibilidad a cambiar en experimentos, omitiéndose inmediatamente. Además, los bloques 1, 12, 13, 14, 20 y 37 ya se sabían insuficientes del experimento 9 para lograr efectuar la simulación del pase escolar en la tarjeta *bip!* normal. Con esto, analizando los datos, y sabiendo que los bloques 16, 17 y 18 poseen información relacionada a los viajes de una tarjeta, siendo relevantes para la decisión que toma un validador respecto al cobro a efectuar en cualquier tarjeta al momento de viajar, se consideran como importantes para realizar los próximos experimentos. Esto no sucede con el bloque 28, ya que por sus diminutas diferencias entre los tipos de tarjeta, se quitó de los experimentos, a fin de no considerarse como incidente en el cobro del pasaje. De esta misma manera, los datos de los bloques 12 y 13, que probablemente tampoco se consideran al momento de realizar un cobro ni siquiera en una TNE, se omite en la siguiente prueba, resumida en los pasos descritos en el experimento 10.

Experimento 10 Segundo intento TNE con diferencias a nivel de bits

1: Obtener los bloques 1, 14, 16, 17, 18, 20 y 37 del último estado de una TNE válida.
2: Copiar a una tarjeta *bip!* al portador estos bloques.
3: Utilizar la tarjeta modificada en algún medio de transporte y analizar los resultados obtenidos.

El experimento 10 sí entregó finalmente el resultado positivo esperado, logrando así concretarse un viaje con tarifa escolar, es decir CLP 210, en una tarjeta *bip!* al portador, tal como se muestra en el rectángulo de color verde, en la figura 5.8.

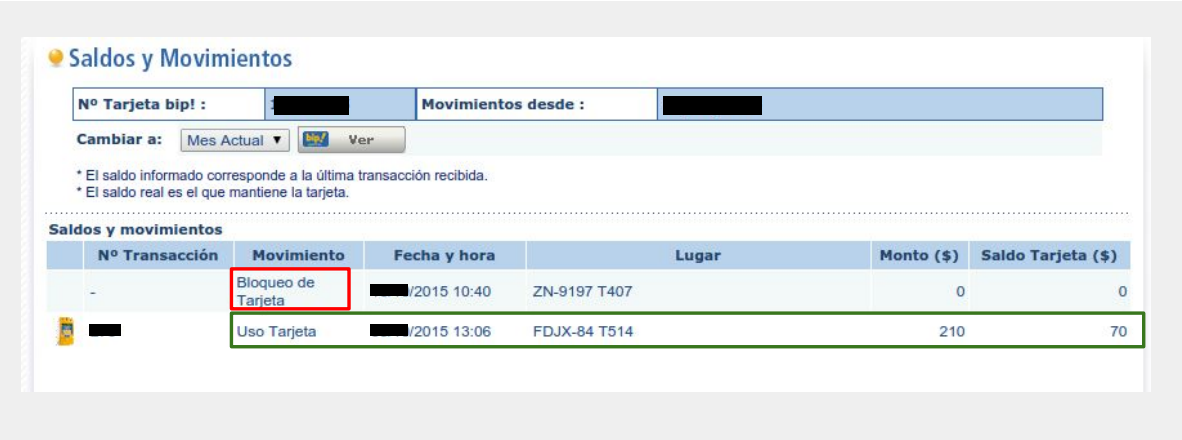

Figura 5.8: Prueba de simulación de TNE en tarjeta *bip!* al portador, en el portal *bip! en línea*. De verde, el viaje efectuado por CLP 210 pesos. De rojo, se muestra el bloqueo sobre la tarjeta modificada. Se omitieron en negro datos privados irrelevantes.

Posterior a esto y contra todo pronóstico, al esperar los días apropiados para la realización del siguiente *clearing* en el *Transantiago*, la tarjeta resulta ser agregada a la lista negra del sistema, bloqueándose al momento de volver a ser usada en el sistema. Esto también puede verse reflejado en el portal *bip! en línea*, tal como demuestra la figura 5.8 con un rectángulo de color rojo. A pesar de ocurrir este bloqueo, no se realiza ninguna medida contra la TNE que presta la información de los bloques utilizados, aunque el bloque 1 de la tarjeta modificada contenía datos como el ID en el *Transantiago* de la tarjeta proveedora de información y no la utilizada, por lo cual fácilmente podría ser referenciada.

Así, las características de las tarjetas *bip!* y del sistema propocionadas por este experimento, se reflejan en el hecho 11, presentado a continuación.

Hecho 11 *Es posible simular una TNE en una tarjeta bip! al portador, a fin de acceder a la tarifa rebajada de estudiante, copiando en la tarjeta normal los bloques 1, 14, 16, 17, 18, 20 y 37 de alguna TNE. Luego del proceso de clearing correspondiente al periodo en donde se realizó este viaje, la tarjeta bip! modificada pasará a estar bloqueada. Con respecto a la TNE que entregó los datos, no se toman mayores medidas, manteniéndose su continuo uso sin problemas.*

Riesgo

El riesgo de este ataque es **bajo**. A pesar de que la vulnerabilidad presentada pareciera tener un futuro prometedor como vector de ataque, el cómo el sistema valida los viajes y discrimina entre los tipos de tarjetas, mostró ser lo suficientemente robusto al momento de determinar cuándo se está cometiendo una infracción. De esta forma, a pesar de la ganancia monetaria temporal que pudiera obtenerse con estas modificaciones, no parece ser realmente

un ataque rentable frente a otras opciones, como la presentada en el ataque expuesto en 5.4.1. Cabe destacar que la ejecución del ataque eventualmente falla en detalles mejorables, pero con la información obtenida en este estudio no fue posible perfeccionarlo.

Mitigación

El sistema actual de mitigación fue eficiente con todos los ejemplos probados. Las tarjetas *bip!* al portador modificadas, lograron funcionar como TNE exclusivamente una vez, accediendo a la tarifa de CLP 210 en esa única ocasión, siendo posteriormente bloqueadas; de igual forma que sucedió en el pasado experimento 2.

Cómo mejora, poco quedaría por hacer sin que significara un esfuerzo importante, el que a su vez, no necesariamente reflejaría un gran aporte a la seguridad del sistema. Lo que se cree más relevante a realizar, y que ya fue mencionado en la sección 5.4.1, es considerar un cambio de modelo de tarjeta a una en la cual se utilice un sistema de encriptación o cifrado probado, no como el vulnerable CRYPTO-1 perteneciente a la tarjeta MIFARE Classic, analizado previamente en el capítulo 4. Como buen ejemplo de cambio, y a fin de poder mantener un funcionamiento similar al actual, destaca la ya mencionada tarjeta MIFARE Plus, la que utiliza `AES-128` para sus transacciones de autentificación e integridad. Este cambio impediría obtener la información de las tarjetas, al ser prácticamente imposible recuperar las claves para leer o escribir la información en las mismas, con lo cual incluso se impediría la pérdida de dinero que genera el adversario al realizar este ataque, antes de ser bloqueado.

5.4.5. Bloqueo de tarjetas mediante el uso de tarjeta especial

Vulnerabilidad

Como ya se mencionó en el hecho 4, la identificación que ocupan las tarjetas en el *Transantiago* para poder validarse como miembros del sistema es el UID, almacenado en el bloque 0 de cada tarjeta. Esta decisión se toma basada en que el bloque 0 de cada tarjeta del sistema viene inhabilitado para escritura desde su fabricación, aceptándose este hecho como una medida de prevención suficiente para evitar cualquier tipo de suplantación de tarjetas *bip!*. Lamentablemente, esta última consideración no es exacta, por lo cual se analizarán sus falencias de forma detallada en la presente sección.

En el mercado, actualmente, existen tarjetas compatibles con el modelo MIFARE Classic de 1 kB, las cuales son absolutamente modificables, incluyendo su bloque 0, conocidas como *tarjetas de UID modificable*[11]. Para este estudio, se utilizó una de estas tarjetas especiales, comprada a la empresa *Clone My Key*[12] desde su sede en Estados Unidos, y se procuró

[11] `http://www.clonemykey.com/uid-changeable-mifare-1k-s50-classic-compatible-card-block-0-direct-write/`

[12] `http://www.clonemykey.com/`

utilizarla como una tarjeta perteneciente al *Transantiago*, intentando incluso en el caso final, negarle el servicio a alguna otra tarjeta *bip!* bloqueándola debido a acciones realizadas mientras se le suplanta con esta tarjeta especial.

Ataque

Lograr bloquear una tarjeta *bip!* normal por el uso fraudulento de una tarjeta de UID modificable, en la cual se estaría utilizando la identificación de la primera. Esto permitiría, bloquear cualquier tarjeta en caso de ser efectivo, sin que realmente estas víctimas cometan alguna acción ilegal.

Experimentos asociados

Comprendiendo los AC. Dado que se cuenta con una tarjeta especial de UID modificable, lo primero que se realizó en el estudio de esta posible vulnerabilidad fue investigar respecto a los bytes que representan las condiciones de acceso de cada sector, conocidos como AC, a fin de que al escribir alguna información en la tarjeta especial no quede grabada de forma permanente de forma innecesaria.

Tal como se indicó en la sección 2.2.2, los AC ocupan los bytes del número 6 al 9 en cada último bloque de cada sector, el cual también es conocido como el *trailer del sector*. Dependiendo de los valores que estos bytes tengan, se determina el acceso de lectura o escritura a cada bloque del sector, la forma con la cuál se debe autentificar un validador en caso de querer trabajar con alguno de estos bloques, e incluso el tipo de bloque que será cada uno de los 3 disponibles en dicho sector para escritura. Para más información respecto a las configuraciones posibles de los AC y sus implicancias, dirigirse al documento [27].

Del análisis efectuado respecto a los AC ocupados en el sistema, se encontraron 6 grupos, para lo cuales varía el tipo de claves A o B, que admite escribir en los bloques, permitiendo además que con cualquiera de ellas se pueda leer sus contenidos. Para mayor detalle de cómo están configurados los bloques en el *Transantiago*, visitar el anexo 7.7, donde se detallan que AC ocupa cada sector y cuáles son las implicancias que tienen sobre sus bloques.

La conclusión más importante del análisis efectuado, es el hecho que el bloque 0 del sector 0 no está bloqueado por los AC en el sistema, sino que únicamente por el bloqueo de sólo lectura que proviene de fábrica en las tarjetas. De esta manera, si se escribe la tarjeta especial con la misma información de una tarjeta *bip!* válida, no habría problemas para volver a escribir los distintos campos en ella un sinnúmero de veces. Esto se describe en el hecho 12.

Hecho 12 *El bloque 0 del sector 0 de cada tarjeta bip! en el Transantiago, no está bloqueado para escritura debido al valor de los bytes que representan las condiciones de acceso del sector, sino que sólo por su bloqueo de fábrica.*

Reemplazando una tarjeta en el sistema. El análisis de este posible ataque continuó con la revisión de la posibilidad de uso transparente de la tarjeta de UID modificable en el sistema, procurando ser otra, utilizando su UID a modo de identificador. Para realizar esto, se siguió la lógica descrita en el experimento 11.

Experimento 11 Reemplazando a otra tarjeta

1: Crear el respaldo de los datos de una tarjeta *bip!* al portador.
2: Escribir tarjeta especial con los datos guardados en el respaldo.
3: Utilizar la tarjeta modificada en algún medio de transporte y analizar los resultados obtenidos.

El uso descrito en el experimento 11 de la tarjeta especial resultó exitoso sólo la primera vez, lográndose un uso transparente ante los ojos del sistema. En las distintas repeticiones de este experimento en el futuro, todas recibieron un mensaje de *Tarjeta no habilitada* por parte del validador, inclusive cuando exactamente la misma información (lo cual se denotará como el mismo estado) funcionó correctamente en la tarjeta original.

En el caso que el experimento 11 funcionó, posterior a los días apropiados para la realización del *clearing* del sistema, el viaje apareció en la página *bip en línea* asociado a la tarjeta original, es decir al UID suplantado. Luego, al escribir la tarjeta original con la nueva información obtenida en la tarjeta especial, posterior a su uso de prueba, ésta también siguió funcionando sin problemas, continuándose el flujo esperado de la tarjeta en el sistema sin ninguna dificultad. Destacar que de este experimento no se adjuntarán imágenes del portal *bip! en línea*, ya que únicamente se vería el flujo normal de una tarjeta *bip!*, lo cual es exactamente lo que se esperaba conseguir.

De esta forma, como en algún momento este ataque tuvo éxito, se procedió a intentar bloquear una tarjeta con esta misma metodología, sólo utilizando sus datos, pero sin usar el plástico original en las transacciones ilícitas.

Intentando bloquear una tarjeta del Transantiago cambiando su saldo. Para completar este ataque, se buscó bloquear alguna tarjeta usando la tarjeta especial con UID modificable como su clon, a fin de incriminar a la tarjeta original con acciones merecedoras de bloqueo. El primer procedimiento probado se describe en el experimento 12, el cual intenta combinar el experimento 11 con el experimento 3 realizado en el ataque 5.4.1, el cual obtiene el bloqueo luego de modificar el saldo de la tarjeta *bip!* en cuestión.

Experimento 12 Intento de bloqueo estándar

1: Crear el respaldo de los datos de una tarjeta *bip!* al portador.
2: Escribir tarjeta de UID modificable con los datos guardados en el respaldo.
3: Modificar saldo de tarjeta de UID modificable con otro saldo.
4: Utilizar la tarjeta modificada en algún medio de transporte y analizar los resultados obtenidos.

Los resultados del experimento 12 en todos sus casos fueron negativos, es decir, no se logró el bloqueo, y el mensaje *Tarjeta no habilitada* apareció en los validadores en cada intento

realizado. Múltiples variantes de este experimento se probaron, incluyendo el uso de distintas tarjetas *bip!* base, además de la modificación de los *bytes de dirección* ubicados en los *bloques de valor* de los saldos, intentando comprender su relación en esta eventual negativa.

A pesar de esta exhaustiva búsqueda, todos los experimentos entregaron el mismo resultado. Este hecho se relacionó con el evento destacado en la sección 5.4.1, en el cual se hace referencia a cómo los supuestos cambios ocurridos desde el segundo semestre del años 2015 impidieron que los ataques de modificación de saldo se pudieran realizar a la misma escala en la que se vivieron el año 2014.

En la realización de estos experimentos, una evidencia dejó en claro que alguna medida se había implementado en el *Transantiago* contra este tipo de casos. El bloque 0 de la tarjeta de UID modificable, posterior a mostrar el mensaje *Tarjeta no habilitada* en el validador, fue borrado completamente a excepción de su UID, tal como muestra la figura 5.9. Esto aclaró que la implementación del sistema considera la posibilidad del uso de este tipo de tarjetas, desarrollando una manera de protección para encontrarlas al ser utilizadas. Puesto que la tarjeta de UID modificable funcionó, al menos en la primera ocasión en el experimento 11, la lógica de cómo el sistema reconoce los casos de *Tarjeta no habilitada* con la tarjeta de UID modificable no resulta dilucidada, quedando su detalle fuera de los alcances de este estudio.

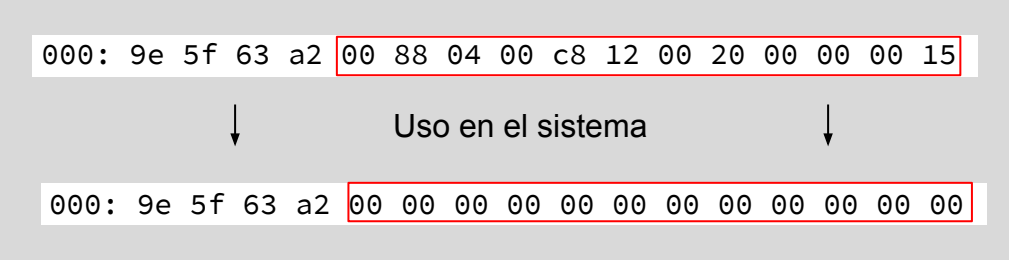

Figura 5.9: Borrado de bloque 0 de tarjeta de UID modificable, luego de desplegarse en el validador el mensaje de *Tarjeta no habilitada*.

Intentando bloquear una tarjeta con simulación de TNE. Un experimento más se realizó buscando como objetivo el bloqueo de una tarjeta, sólo teniendo su UID y la tarjeta de UID modificable. En este caso, a diferencia del experimento 12, se intentó bloquear la tarjeta con el segundo método descubierto para bloquear tarjetas en este estudio: la simulación de una TNE con una tarjeta *bip!* tradicional. El procedimiento a realizar entonces consiste de la mezcla del experimento 10 con el experimento 11, lo cual se describe a continuación en el experimento 13.

Experimento 13 Intento de bloqueo con cambio a TNE

1: Crear el respaldo de los datos de una tarjeta *bip!* al portador.
2: Obtener los bloques 1, 14, 16, 17, 18, 20 y 37 del último estado de una TNE válida.
3: Escribir tarjeta de UID modificable con los datos guardados en el respaldo.
4: Modificar tarjeta de UID mdificable con los bloques obtenidos de la TNE.
5: Viajar con la tarjeta modificada.
6: Esperar bloqueo de la tarjeta normal.

La ejecución de este experimento, al igual que en el caso del experimento 12, nuevamente entregó como respuesta el mensaje *Tarjeta no habilitada* para todos sus intentos, borrando a su vez, el bloque 0 de la tarjeta especial en cada ocasión, tal como ocurrió los experimentos anteriores. Así, esta eventual vulnerabilidad ha sido contrarrestada de forma exitosa por el sistema, al menos en los intentos de bloqueo detallados en este estudio, lo cual se representa en el hecho 13.

Hecho 13 *El sistema detecta, en la mayoría de los casos, el uso de este tipo de tarjetas especiales con UID modificable. En cada caso en el que intentó bloquearse alguna tarjeta bip! al portador usando la tarjeta de UID modificable en su lugar, el validador entregó el mensaje de Tarjeta no habilitada, seguido del borrado del bloque 0 desde el byte 4 en adelante. Esto ocurrió con ataques que son efectivos bloqueando tarjetas bip! al portador a la fecha de realización de este trabajo. A pesar de esto, el sistema aceptó en una ocasión, de forma transparente, una tarjeta especial de UID modificable simulando ser una tarjeta bip!.*

De esta manera, sólo existen distintos supuestos respecto a este comportamiento del sistema, pero no nociones absolutas. Lamentablemente, la cantidad de experimentos a poder realizar es limitada, como también lo son los recursos, tal como sucede en este caso con las tarjetas de UID modificables. De todas formas, un hecho es hasta este punto totalmente cierto: el bloqueo no logró efectuarse, y se detectaron medidas de contención por parte del sistema al momento de cobrar en una suplantación. Destacar que este hecho no implica que realizar el ataque aquí presentado sea imposible, sino simplemente, que los métodos utilizados en este trabajo no bastaron para realizarlo.

Simulación de tarjeta bip! bloqueando el bloque 0 de la tarjeta de UID modificable. Dado que al utilizar la tarjeta de UID modificable en el sistema éste borra el bloque 0 de la misma desde el byte 4 en adelante, tal como lo declara el hecho 13, se presume que dicha medida es la forma que utiliza el sistema para poder reconocer si una tarjeta es de UID modificable, pudiendo cambiar así su bloque 0, o es una perteneciente al sistema, por lo cual tiene su bloque 0 inhabilitado para escritura. Así, sabiendo además que los bloques de las tarjetas pueden bloquearse para escritura utilizando los AC correctos en el bloque *trailer* del sector correspondiente, se presume posible bloquear para escritura el bloque 0 de la tarjeta de UID modificable y emular el comportamiento de una tarjeta normal, pasando inadvertida en el sistema.

De esta manera, para validar la suposición planteada, se realizó el experimento 14, tratando de comprender el mecanismo de reconocimiento de tarjetas que utiliza el sistema para estos casos.

Experimento 14 Bloqueando bloque 0 en tarjeta de UID modificable

1: Crear el respaldo de los datos de una tarjeta *bip!* al portador.
2: Copiar el respaldo guardado en la tarjeta de UID modificable.
3: Modificar los AC del sector 0 para bloquear la escritura sobre el bloque 0.
4: Intentar viajar con la tarjeta modificada y analizar los resultados.

Para realizar el paso 3 del experimento 14, se analizaron los AC correspondientes al sector 0 de todas las tarjetas *bip!* del sistema. Estos tienen el valor `78 77 88`, el cual, tal como se detalla en el anexo 7.7, permite la lectura del bloque 0 por ambas claves y su escritura solamente con su clave B. De esta forma, siguiendo las instrucciones de [27] para la estructuración de bytes AC se determinó que, para dejar todo los demás bloques en el mismo estado y lograr que el bloque 0 siga siendo leíble por ambas claves, pero no editable por ellas, el valor que debían tener dichos bytes es `69 67 89`.

Al realizar el experimento 14, el resultado en su prueba fue positivo, lográndose así emular una tarjeta *bip!* al portador en la tarjeta de UID modificable, validándose sin problemas en el *Transantiago*, reconociéndose esta transacción incluso en la plataforma *bip! en línea*. Este resultado revela un hecho particular, y es que no existe una revisión exhaustiva de todos los valores que por defecto deberían mantener todas las tarjetas *bip!* en el sistema, como son los AC de los sectores. Esta idea, se resume en el hecho 14.

Hecho 14 *El sistema no comprueba en cada transacción la correctitud de los AC en la tarjeta usada, que por defecto son iguales en todas las tarjetas en el sistema.*

Posterior a realizar el experimento 14, se continuó intentando realizar una nueva versión de los experimentos 12 y 13, pero esta vez utilizando el método aquí descrito para impedir la detección de la tarjeta de UID modificable en el sistema. A pesar de lo prometedor que estos experimentos parecían ser, ambos terminaron entregando el mensaje *Tarjeta no habilitada* al intentar bloquear remotamente una tarjeta *bip!*, con lo cual terminó conjeturándose que el problema no está de forma principal en el uso de esta tarjeta de UID modificable, sino que en la dificultad de generar estados válidos en esta tarjeta usando un UID diferente al de la cual se saca la información base. Comprender este ámbito en su totalidad no se logró en el desarrollo de este trabajo.

Intentando bloquear una tarjeta con inconsistencias en el saldo. Un experimento distinto que se planeó con el fin de lograr el bloqueo usando la tarjeta de UID modificable como sustituto de una tarjeta *bip!* normal, y que resultó ser la base del siguiente ataque, propone usar un estado válido de la tarjeta original, tanto en ella como en la tarjeta de UID modificable al mismo tiempo en el sistema. Esto lograría que, al realizarse el *clearing* correspondiente en el sistema, se arroje un error de monto, dada la inconsistencia en el flujo de uso de la tarjeta, ya que tendría dos usos desde el mismo estado, omitiéndose un cobro. Esta idea se relata en el experimento 15, y a pesar de lo esperado, tampoco consiguió el bloqueo. Las implicancias de este nuevo ataque, incluyendo sus riesgos y mitigaciones propuestas, se analizan en la sección 5.4.6.

Experimento 15 Intento de bloqueo con inconsistencia

1: Crear el respaldo de los datos de una tarjeta *bip!* al portador.
2: Escribir tarjeta de UID modificable con los datos guardados en el respaldo.
3: Viajar con la tarjeta *bip!* normal.
4: Viajar con la tarjeta modificada con el mismo estado de la tarjeta *bip!* normal usada.
5: Esperar bloqueo.

Al ejecutar el experimento 15 el resultado obtenido fue contrario a lo esperado, por lo cual el bloqueo no se efectuó. Es decir, se aceptaron ambos estados grabados en las respectivas tarjetas como válidos en el sistema, saltando entre ellos posterior a distintos usos. Con ésto, se encontró una nueva opción de ataque que permitía a su vez, obtener beneficios económicos si se utilizaba de la forma apropiada, ya que acepta inconsistencias en los saldos registrados para una tarjeta en el sistema, sin cuestionar su validez en el *backend*, sólo su estado en la tarjeta. Para un análisis más detallado, se analizará esta vulnerabilidad completamente en la sección 5.4.6.

Riesgo

Dado que finalmente este ataque no logró implementarse de ninguna forma, el riesgo asociado se considera como **nulo**.

Mitigación

De igual forma que en la vulnerabilidad anterior, sin ataque concreto, la mitigación no tiene un mayor sentido para el ataque en si. A pesar de esto, hay hechos de los experimentos asociados que sí merecen un análisis, dados los peligros que dejan de manifiesto con sus resultados. Puesto que no se tiene una comprensión completa del sistema, no haber logrado efectivamente el bloqueo no implica necesariamente que no se pueda realizar, ni que eventualmente no puedan usarse tarjetas especiales de este tipo o distintas, para diversos fines. Y eso es uno de los principales problemas de las implementaciones privativas como ésta: nunca se puede eliminar la posibilidad de que vulnerabilidades permanezcan aún después de haber sido revisadas por sus diseñadores. Hechos como el primer resultado del experimento 11, dejan en duda las capacidades del sistema de realmente poder bloquear comportamientos indeseados, y abren la puerta a suponer que, basta comprender parte de cómo el sistema reconoce un estado en una tarjeta como válido para poder vulnerar el sistema de cobros completamente.

Tal como se mencionó en múltiples ataques, y como se concluirá en el siguiente capítulo, la posibilidad de leer y editar la información a la cual no se debería poder acceder como usuario del sistema, es una complicación demasiado grande con la que lidiar para mantener el sistema seguro. La mejor mitigación para cualquiera de estos ataques, es el invertir en el cambio de implementación del sistema, pasando del actual a otro en el cual su contenido realmente esté almacenado de forma segura.

Por último, respecto a la detección de las tarjetas de UID modificable por los validadores, lo realizado en el experimento 14 demuestra que la medida de prevención del sistema consistente en modificar el bloque 0 a fin de cerciorarse de que es o no modificable, y por lo tanto, saber si pertenece dicha tarjeta al sistema. Como medida alternativa, se propone cambiar los AC del sector 0, a fin de que por su configuración el bloque 0 y el *trailer* del sector también estén bloqueados. Ambas medidas combinadas, impiden la repetida utilización de estas tarjetas de UID modificable, ya que bloquean la escritura del primer sector y la edición de los AC, los cuales modificados correctamente podrían cambiar estas condiciones. Esta medida, se considera útil como prevención, en caso de poder realizarse el bloqueo de otras tarjetas, cosa

no efectuada en este trabajo, desactivando así la ventaja que entregan las tarjetas de UID modificable respecto a las tarjetas *bip!* existentes en el sistema.

5.4.6. Repetición de estados previos de tarjeta *bip!*

Vulnerabilidad

Una vulnerabilidad descubierta mientras se analizaba la vulnerabilidad detallada en 5.4.5, en particular al realizar el experimento número 15, fue la incapacidad del sistema de distinguir como un uso incorrecto la repetición de estados válidos en una tarjeta. Esto funciona incluso si existen otros usos entre el estado actual y el anterior que se vuelve a utilizar, incluso provocandose así inconsistencias en el flujo de los saldos de la misma tarjeta. Es decir, almacenando el estado de determinada tarjeta, es posible volver reiteradas veces al mismo copiando los datos almacenados en la tarjeta, sin que el sistema lo reconozca como un caso problemático. Esto replanteó análisis y conclusiones obtenidas previamente en este trabajo, lo que se analizará y detallará en la presente sección.

Ataque

Volver una tarjeta a estados previos, los cuales fueron válidos en su momento. Al utilizarla, si funciona sin problemas, evidentemente se generarán perdidas de dinero en el sistema, en particular, si el estado anterior tiene mayor saldo al estado actual.

Experimentos asociados

Uso de estados previos de una misma tarjeta. Tal como se detalló en los experimentos asociados a la sección 5.4.5, en particular en el experimento 15 el sistema no reconoció como errónea la utilización en paralelo de determinada tarjeta *bip!* con su copia almacenada en la tarjeta especial con UID intercambiable, ambas usadas con el mismo estado, apareciendo ambos viajes sin problemas en la plataforma *bip! en línea* posterior al periodo de actualización de información correspondiente. Es decir, se generó un viaje gratis, sin detectarse conflictos de saldos ni detonarse algún bloqueo posterior. Aparte del experimento 15, se realizó el experimento 16 a modo de verificación de este ataque.

Experimento 16 Uso estado anterior

1: Crear el respaldo de los datos de una tarjeta *bip!* al portador.
2: Viajar con la tarjeta normal.
3: Reemplazar los datos de la tarjeta con los almacenados en el respaldo.
4: Viajar con la tarjeta modificada.
5: Analizar resultados del uso.

Efectivamente, este ataque funcionó sin problemas, tal como puede evidenciarse en la figura 5.10 en dos ocasiones, con información proveniente de la plataforma *bip! en línea*.

Cabe destacar, que las implicancias respecto al funcionamiento del sistema expuestas con este ataque, atentan directamente con lo presentado en el hecho 6, y plantean más dudas que respuestas respecto al cómo funciona, efectivamente, el sistema de bloqueos. Variadas hipótesis se intentaron probar respecto al funcionamiento de los bloqueos, pero finalmente, no pudieron ser esclarecidas. De esta forma, es necesario actualizar el hecho 6, para que así contemple este caso como excepción, lo cual se resumirá en el siguiente texto.

Hecho 15 ***: Reemplazo Hecho 6.*** *El bloqueo de tarjetas se realiza revisando los cambios de saldo en cada transacción. En caso de que alguno no sea congruente con los que se tiene almacenado en el sistema, la tarjeta pasa a lista negra en los validadores, quedando pendiente la ejecución del bloqueo para su próximo uso. Extrañamente, si el estado de la tarjeta se repite, el sistema no deja de considerarlo como un estado válido, sino que sigue desde el último estado utilizado en el sistema, por lo cual, el bloqueo no se realiza.*

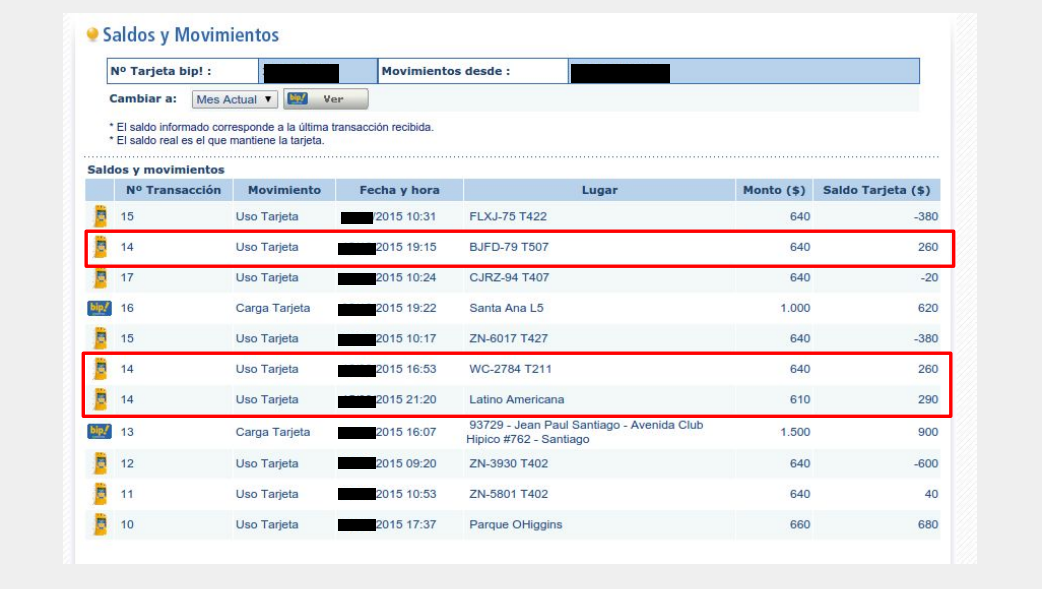

Figura 5.10: Prueba del ataque de repetición de estados previos en tarjeta, revisado en *bip! en línea*. De rojo, todas las ocasiones en que se volvió al estado almacenado (transacción 13), comienzando desde el estado 14. Se omitieron en negro datos privados irrelevantes.

Uso de estados entre distintas tarjetas. Dado que se logró realizar el retorno de una tarjeta a sus estados pasados sin problemas, emergió rápidamente la inquietud de qué tan necesario es tener un estado pasado de la misma tarjeta, en vez de utilizar el estado actual de otra tarjeta; por ejemplo, en la cual se tiene un saldo mucho mayor. Para probar esta proposición se planteó el experimento 17.

Experimento 17 Uso estado de otra tarjeta

1: Obtener el respaldo de los datos de una tarjeta *bip!* al portador.
2: Copiar respaldo en una tarjeta *bip!* a utilizar distinta a la primera (se pueden pegar todos los bloques, menos el bloque 0 por bloqueo de escritura)
3: Utilizar tarjeta modificada en el sistema y analizar resultados del uso.

Este experimento, se realizó en dos variantes; una copiando toda la información de otra tarjeta a excepción del bloque 0; y otra, copiando todo excepto el bloque 0 y el bloque 1, suponiendo que eventualmente pudiera existir alguna correlación del UID de una tarjeta con su ID en el *Transantiago*, comprobable en el momento del uso. Para ambos experimentos, el resultado fue el mismo, respondiendo el validador con el mensaje *Tarjeta no habilitada*, no siendo así posible realizar este ataque con éxito.

Este hecho nos entrega información respecto a cuándo una tarjeta es válida o no en el sistema. Evidentemente, existen bloques importantes en cada tarjeta, los cuales de alguna forma se relacionan con más información en la misma, a fin de mantener una consistencia de los datos, y poder validarse o no el estado completo de una tarjeta previo a cada uso. De esta forma, el bloque 0 forma una parte vital en este proceso de validación, y por experimentos previos realizados, sabemos que incluso hay bloques, por ejemplo los usados para conseguir simular una combinación de viaje en la sección 5.4.2, que no lo son. Esta idea, se resume completamente en el hecho 16, el cual da una base sobre el estudio de los estados válidos de tarjetas en el *Transantiago*.

Hecho 16 *El bloque 0 es utilizado en el cálculo de la validación del estado de una tarjeta, efectuado previo a autorizar su uso en un validador. De igual forma, los bloques 16, 17, 18, 21, y 22 no forman parte de ese proceso, o el ataque 5.4.2 no funcionaría.*

Riesgo

El riesgo de esta vulnerabilidad es **alto**. Probablemente, el primer ataque planetado en esta sección sea el más fácil de realizar de todos los analizados en este estudio, y a su vez, el más amenazante para todo el sistema. Cualquier usuario, sin mayores requerimientos que el acceso a algún *smartphone* compatible con tecnología NFC, puede utilizar herramientas como *MIFARE Classic Tool* (detallada en 1), la cual permite crear *dumps* de tarjetas (teniendo las claves del sistema agregadas), y luego utilizarlos para devolver una tarjeta al estado previo almacenado. De esta forma, el ataque es rápido, accesible, puede generar perdidas económicas importantes sin mayor costo que el poseer una *bip!* cargada la primera vez, un *smartphone*, y el mínimo conocimiento de uso de alguna aplicación que permita escribir en la tarjeta misma.

Mitigación

Las principales mitigaciones a considerar son dos, las cuales ya fueron propuestas para ataques anteriores. Primero, tal como se mencionó en el ataque 5.4.2, y dado que probablemente es la medida más fácil de implementar (pero no la más segura,) se recomienda como vital para tener un sistema coherente, un análisis de los flujos de los usos de las tarjetas. Evidentemente hay cambios de montos disponibles que no pueden ocurrir sin un respaldo apropiado, o saltos de número de usos que no tienen mayor sentido. Estos casos son evidentemente sospechosos, y aunque no se puede culpar absolutamente al usuario de estos comportamientos irregulares, al menos se convierten en un material importante de análisis. Se estima que en el mejor de los casos, el sistema debería funcionar con un comportamiento similar a lo expuesto en el

hecho 6.

Por último, y como se ha mencionado en las mitigaciones para los ataques propuestos en las secciones 5.4.1 y 5.4.4, el avance del sistema hacia el uso de una tarjeta que impida el acceder a la información resguardada, con algún tipo de protocolo seguro basado en *criptografía*, es vital. Las mitigaciones actuales que han ido levantándose con el tiempo en el sistema, son cada vez más robustas, pero el hecho de tener que lidiar con el acceso a información importante en las tarjetas por parte de algún adversario en todo momento, es un requerimiento demasiado estricto para implementar un sistema seguro, que a su vez, no cuenta con el tiempo ni el procesamiento suficiente para plantear mejoras. Basta simplemente que el tiempo transcurra para que al final, alguien entienda todas las consideraciones que necesita el sistema para tener por ejemplo, un estado válido en una tarjeta, con lo cual finalmente podría vulnerar completamente el sistema de cobros del *Transantiago*.

5.5. Cuadro de resumen de los ataques presentados

A continuación se presenta una tabla resumen con todos los ataques presentados en este capítulo, junto a una pequeña descripción y su efectividad en el sistema.

Nombre del Ataque	Descripción	Tipo	Estado	Contramedidas
Obtención de datos del usuario desde la TNE	Obtención del nombre y RUN del dueño de una TNE a partir de los datos de su tarjeta	Lectura	Funciona	No tiene
Obtención de la dirección de un usuario	Obtención de la dirección de un usuario a partir del número de su tarjeta, con una probabilidad del 50 %	Lectura	Funciona	No tiene
Modificación del saldo de una tarjeta	Cambio del saldo de una tarjeta *bip!* modificando los bloques de valor 33 y 34, obteniendo un beneficio monetario	Escritura	Funciona	La tarjeta es bloqueada días después del uso
Obtención de viaje gratis simulando viaje previo	Emulación de transbordo en una tarjeta *bip!* usando bloques de otra que haya viajado hace menos de dos horas	Escritura	Funciona	No tiene
Negación de servicio a tarjeta *bip!* cercana	Modificación de algún bloque de una tarjeta *bip!* cercana a fin de invalidarla en el sistema por no corresponder sus *bytes de verificación* con sus datos	Escritura	Funciona	No tiene
Simulación de TNE en tarjeta *bip!*	Intento de simular el comportamiento de las TNE en el sistema en una tarjeta *bip!* al portador, a fin de tener acceso al cobro reducido de estudiantes	Escritura	Funciona	La tarjeta es bloqueada días después del uso
Bloqueo de tarjetas mediante el uso de tarjeta especial	Intento de bloqueo remoto de una tarjeta *bip!* normal, utilizando su UID en una tarjeta de UID modificable para simular usos conflictivos	Escritura	No funciona	Borrado de bloque 0 al ser detectada. Esto puede ser evitado con el uso de AC convenientes.
Repetición de estados previos de tarjeta *bip!*	Uso de estados previos válidos de tarjeta *bip!* nuevamente sobre la misma, obteniendo un beneficio monetario con sus cambios de saldo	Escritura	Funciona	No tiene

Tabla 5.1: Cuadro resumen de los ataques presentados en este trabajo.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Uno de los objetivos de la presente tesis fue poder analizar de forma profunda el sistema de pago del *Transantiago*, proyecto que de sus inicios fue cuestionado abiertamente por sus implicaciones políticas y sociales, pero inusualmente criticado en el aspecto tecnológico. Si bien, una vez puesto en marcha, expertos en el tema advirtieron de problemas con las tecnologías que usaba (en particular con sus tarjetas MIFARE Classic), en la práctica dichas advertencias fueron aparentemente minimizadas o simplemente ignoradas.

Este trabajo de tesis entrega un análisis de un grupo de vulnerabilidades que a pesar de los intentos de parchar el sistema no pudieron ser eliminadas. Inhabilitar tarjetas de usuarios, obtener información privada de ellos e incluso realizar viajes gratis emulando transbordos son ejemplos de ellas. Además, este trabajo evidencia la existenciade problemas que de forma segura nunca van a poder ser erradicados, si no se realizan cambios importantes en el diseño e implementación del sistema completo. Porque a pesar de los esfuerzos de los encargados del *Transantiago*, las tecnologías utilizadas siguen siendo absolutamente inseguras para efectuar transacciones monetarias, lo que entrega un riesgo totalmente innecesario a una componente vital para el transporte público santiaguino.

Como estudio, esta tesis innova al ser uno de las primeras propuestas en analizar un sistema abiertamente conocido como inseguro en plena aplicación, probando las medidas de prevención tomadas por los implementadores en terreno, evaluando la eficiencia de estos procedimientos precautorios y concluyendo respecto a su real efectividad tanto para con el sistema, como para sus distintos usuarios.

Sin embargo, este trabajo no se queda en la enumeración de potenciales ataques sino que también propone mitigaciones y soluciones, medidas que podrían considerarse a fin de mejorar el *Transantiago*, y evitar así, complicaciones futuras.

Así, la principal medida propuesta a considerar para la prolongación de la vida del sistema es el reemplazo de las tarjetas MIFARE Classic por una mejor tecnología, que efectivamente

cifre su información siguiendo *protocolos criptográficos* probados y seguros, y que de forma obligatoria, no sea abiertamente conocida como vulnerable, ni se encuentre bajo sospecha debido a estudios que pronostiquen importantes vulnerabilidades. Tal como se mencionó previamente, existen tecnologías muy similares a las ya utilizadas en funcionamiento, como la tarjeta MIFARE Plus. Esta tarjeta utiliza AES-128, una primitiva criptográfica probada, tanto para su acceso y verificaciones. De todas maneras, un cambio de tecnología requiere la realización de un estudio acabado previo (ya que incluso se han reportado potenciales ataques a tarjetas en principio más seguras que las MIFARE Classic, como la MIFARE DESFire[1]), por lo cual cualquier cambio de este tipo, debe ser analizado en el largo plazo.

Complementario a esto, se proponen medidas paralelas, a fin de permitir convivir a los usuarios con el sistema del *Transantiago* por un mayor tiempo sin preocupaciones respecto a su funcionamiento. El objetivo de esto es evitar las dudas respecto a la seguridad del sistema, que se podrían volver a generar al encontrarse nuevas fallas a escala nacional, tal como las que se proponen en este estudio o las que se vivieron el año 2014. Entre las medidas de contención destacan la instauración de un sistema de análisis de flujos de usos de las tarjetas más robusto y la creación de un sistema de verificación a nivel de cada tarjeta respecto a la validez de su estado.

En particular, en esta primera medida debería considerar, en los momentos en que se almacenan en el sistema los nuevos datos recopilados, un análisis de los flujos de uso que poseen las distintas tarjetas, a modo de comprobar que efectivamente no existen saltos ilógicos entre estados, ni dineros sin sentido, ni ninguna inconsistencia como las mencionadas en este estudio. En particular, en la comprobación de los montos, esta revisión debería ser implacable a diferencia de lo que ocurre hoy, ya que si el saldo no coincide con lo que el sistema determina que debería ser, considerando tanto recargas como usos, la tarjeta debería ser bloqueada. De igual manera debería funcionar con los usos o viajes sin sentido. Así implementadas de forma correcta, estas medidas evitarían los dos ataques más importantes presentados en este trabajo (ver secciones 5.4.2 y 5.4.6).

Como segunda medida de mitigación, está la instauración de un sistema de verificación del estado de los datos de una tarjeta, el cual a diferencia de los *bytes de verificación* estudiados, debiera funcionar a nivel global en la tarjeta y con un sistema criptográfico demostradamente seguro. El principal problema de los *bytes de verificación* implementados actualmente es que sólo mantienen la integridad de un bloque de la tarjeta en estudio, y no de todos los bloques que conforman la tarjeta, en forma completa. Esto permite, tal como se realizó en el ataque de la sección 5.4.2, copiar bloques completos sin miedo a romper esta verificación, ya que se sabe que al estar en otra tarjeta válida en el sistema, el bloque es válido. Por consiguiente, la idea consiste en instaurar una medida de verificación a nivel tarjeta, que en particular utilice todos los bloques que el mismo sistema utiliza en una misma interacción, los cuales se espera varíen entre las distintas tarjetas dependiendo de las distintas operaciones.

Este trabajo muestra evidencia empírica que algo de este estilo estaría siendo realizado en ciertas partes de la tarjeta. Sin mebargo, dicho cambio no sería suficiente, tal como se

[1] *Fuente*: `http://blog.trendmicro.com/trendlabs-security-intelligence/hacking-rfid-payment-cards-made-possible-with-android-app/`

menciona en el hecho 16 (Sección 5.4.6). Como una sugerencia de implementación, se propone el uso de un MAC (*Message Authentication Code* o *Esquema de Autentificación de Mensajes*, como por ejemplo HMAC[2]), o de al menos un mecanismo criptográfico considerado seguro. En particular, un sistema cuya seguridad recaiga en la mantención de una clave secreta del sistema, y no en qué tan oculto se encuentra el algoritmo de cifrado para sus usuarios. Por cierto, la factibilidad de este tipo de sugerencias deberá ser evaluada en función de los recursos disponibles actualmente. Por ejemplo, es probable que los validadores tengan limitaciones en términos de procesamiento para ejecutar operaciones criptográficas más sofisticadas en el tiempo deseado, y por ello, limitan la aplicabilidad de dichas operaciones. Cabe destacar por último, que esta última medida temporal favorece la posibilidad de inhabilitar tarjetas alteradas, tal como se planteó en la sección 5.4.3

Por último, una de las conclusiones más importantes en este trabajo se refiere a la alta criticidad de los ataques encontrados, pese a partir de la hipotesis que las motivaciones de los atacantes serían principalmente de índole personal y económicas. En varias ocasiones estos ataques claramente implican un peligro adicional para la robustez del sistema en su conjunto. En este trabajo explícitamente se ignoró el efecto de atacantes más complejos, tales como grupos organizados con objetivos disruptivos (vandalismo o incluso terrorismo), los cuales simplemente deseen hacer el mayor daño posible al sistema de transporte. En estos casos, los riesgos aumentan significativamente, convirtiendo en una real posibilidad el colapso del sistema de pago estudiado mediante la explotación conjunta y simultanea de varias de las vulnerabilidades mencionadas, y tornando inútiles muchas de las medidas de protección actualmente instauradas en el sistema. En este sentido, la mejora de la seguridad del sistema de pago de *Transantiago* debiera considerarse una prioridad en cuanto a que su falla puede tener serios riesgos en la disponibilidad de la operación de una infraestructura crítica chilena.

6.2. Trabajo Futuro

6.2.1. Búsqueda de fórmula para calcular los bytes de verificación.

Durante este estudio, mientras se realizó el análisis de los primeros ataques de modificación de *bloques de datos*, en particular del ataque 5.4.1, se intentó durante un periodo de tiempo encontrar la fórmula para calcular los *bytes de verificación* en el sistema. En este análisis se consideraron distintas posibilidades de verificación de 8 bits, entre las cuales se profundizó en variados *checksums* y CRC (sigla de *verificaciones por redundancia cíclica*, en inglés).

Para el caso de los *checksums*, se implementaron varios tipos conocidos que entregaran resultados de 8 bits en *Java*. En el caso del CRC, se comenzó probando con *polinomios característicos* conocidos en este tipo de usos, que entregaran como resultados cadenas de 8 bits. Ante el fracaso de esta búsqueda, se implementaron distintos tipos de variaciones de CRC, probando finalmente todas las combinaciones posibles de *valor de inicio* con *polinomio característico*. Asumiendo la posibilidad de errores en la implementación y fallas en el manejo

[2] `https://en.wikipedia.org/wiki/Hash-based_message_authentication_code`

de la representación de la información utilizada en las tarjetas, se utilizaron herramientas como *CRC RevEng*[3], con propósito de buscar mayores variaciones posibles con casos de prueba, pero lamentablemente se terminó sin conseguir resultados positivos.

Esta búsqueda se detuvo, no sin antes recopilar cerca de 350 valores de prueba[4] de estos bytes de verificación, obtenidos de todos los usos efectuados con las tarjetas registradas, con objeto de contar un buen número de pruebas para considerar al retomar esta búsqueda.

6.2.2. Comprensión del sistema.

Dado el periodo en el cual se realizó este trabajo y lo dinámico que resultó ser el sistema del *Transantiago* en este mismo, se vivieron en paralelo una serie de modificaciones, las cuales no pudieron analizarse a cabalidad. Así, sería interesante como complemento de este trabajo, lograr definir completamente las características del sistema que quedaron sin ser dilucidadas.

Como ejemplo importante, se destaca la comprensión cabal de las situaciones donde se muestra el mensaje *Tarjeta no habilitada*, es decir, cuándo efectivamente una tarjeta es válida para el sistema y cuándo no. También, se mantienen dudas respecto a las medidas efectuadas en el sistema, que lograron que los casos de modificaciones de saldos que funcionaban sin problemas empezaran a fallar pasado el periodo de invierno del 2015, tal como se menciona en los experimentos 5.4.1 y 5.4.5. Por último sería útil comprender cuáles son las condiciones efectivas para que se realice un bloqueo de tarjeta, ya que dado lo descubierto en el experimento 5.4.6, pareciera ser que las medidas planteadas en el hecho 15 en un comienzo del estudio, es decir una comprobación exhaustiva en el flujo de saldos de una tarjeta y sus usos, no era tan robusta como se pensaba.

[3] `http://reveng.sourceforge.net/`

[4] Estos valores están disponibles en formato CSV para descarga en `https://drive.google.com/file/d/0B6JbuMwMyO18MlRWbzVJWnFpNk0/view?usp=sharing`

Bibliografía

[1] Identification cards - Contactless integrated circuit cards - Proximity cards. ISO/IEC 14443, 2001.

[2] TVN 24Horas.cl. Hacker descubre cómo cargar una tarjeta bip sin dinero. Online, 10 2013. http://www.24horas.cl/tendencias/mundodigital/hacker-descubre-como-cargar-una-tarjeta-bip-sin-dinero-906583.

[3] Massachusetts Bay Transportation Authority. The charlie card reusable ticket system. Online. http://www.mbta.com/fares_and_passes/charlie/.

[4] Jorge Bahamonde, Alejandro Hevia, Giselle Font, Javier Bustos-Jimenez, and Camila Montero. Mining private information from public data: The transantiago case. *Pervasive Computing, IEEE*, 13(2):37–43, 2014.

[5] ISG Smart Card Center. Counter expertise review of the TNO security analysis of the ducth OV-Chipkaart. *Royal Holloway University of London, Information Security Gropu - Smart Card Center*, 2008.

[6] Cooperativa.cl. Usuarios reportan problemas con la tarjeta bip! Online, 05 2015. http://www.cooperativa.cl/noticias/site/artic/20150512/pags/20150512112100.html.

[7] EasyCard Corporation. Easycard. Online. http://www.easycard.com.tw/.

[8] Nicolas Courtois, Karsten Nohl, and Sean O'Neil. Algebraic attacks on the Crypto-1 stream cipher in mifare classic and oyster cards. *IACR Cryptology ePrint Archive*, 2008:166, 2008.

[9] Nicolas T. Courtois. The dark side of security by obscurity and cloning mifare classic rail and building passes anywhere, anytime. Cryptology ePrint Archive, Report 2009/137, 2009. http://eprint.iacr.org/.

[10] Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D Garcia. A practical attack on the mifare classic. In *Smart Card Research and Advanced Applications*, pages 267–282. Springer, 2008.

[11] Equipo FayerWayer. Aplicación permite hackear sistema de pagos del transporte público de santiago. Online, 10 2014. http://www.fayerwayer.com/2014/10/logran-hackear-sistema-de-pagos-del-transporte-publico-de-santiago/.

[12] Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Wirelessly pickpocketing a mifare classic card. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 3–15, Washington, DC, USA, 2009. IEEE Computer Society. http://dx.doi.org/10.1109/SP.2009.6.

[13] FlavioD. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, RonnyWichers Schreur, and Bart Jacobs. Dismantling mifare classic. In Sushil Jajodia and Javier Lopez, editors, *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer Berlin Heidelberg, 2008. http://dx.doi.org/10.1007/978-3-540-88313-5_7.

[14] Kishan Gupta. Hacking mifare classic. http://www.nicolascourtois.com/MifareClassicHack.pdf.

[15] J Kamlofsky. Selective attacks to mifare classic cards.

[16] Timo Kasper, Michael Silbermann, and Christof Paar. All you can eat or breaking a real-world contactless payment system. In *Financial Cryptography and Data Security*, pages 343–350. Springer, 2010.

[17] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, vol. IX, 1883.

[18] Keith E. Mayes and Carlos Cid. The mifare classic story. *Inf. Secur. Tech. Rep.*, 15(1):8–12, February 2010. http://dx.doi.org/10.1016/j.istr.2010.10.009.

[19] DTP Metropolitano. Conoce las tarifas. Online. http://www.transantiago.cl/tarifas-y-pagos/conoce-las-tarifas.

[20] El Mostrador. 34 mil tarjetas bip serán bloqueadas por efectuar carga ilegal. Online, 10 2014. http://www.elmostrador.cl/pais/2014/10/24/34-mil-tarjetas-bip-seran-bloqueadas-por-efectuar-carga-ilegal/.

[21] Ruben Muijrers. Reverse engineering the memory layout of the ov-chipkaart.

[22] Andrés Navarro. Nuestra verdad sobre el transantiago: una lección de resiliencia. Online, 04 2007. http://www.economiaynegocios.cl/noticias/noticias.asp?id=26306.

[23] Karsten Nohl. Cryptanalysis of crypto-1. *Computer Science Department University of Virginia, White Paper*, 2008.

[24] Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. Reverse-engineering a cryptographic RFID tag. In *USENIX Security Symposium*, volume 28, 2008.

[25] Karsten Nohl and Henryk Plötz. Mifare, little security, despite obscurity. In *Presentation on the 24th congress of the Chaos Computer Club in Berlin*, 2007.

[26] NXP. Mifare classic. Online. http://www.nxp.com/documents/leaflet/MIFARE_Classic.pdf.

[27] NXP. Mf1s50yyx: Mifare classic 1k - mainstream contactless smart card ic for fast and easy solution development. Product data sheet, 5 2011. http://www.nxp.com/documents/data_sheet/MF1S50YYX.pdf.

[28] NXP. Mf1s70yyx: Mifare classic 4k - mainstream contactless smart card ic for fast and easy solution development. Product data sheet, 5 2011. http://www.nxp.com/documents/data_sheet/MF1S70YYX.pdf.

[29] Octopus. Octopus cards. Online. http://www.octopuscards.com/.

[30] Mayor of London. Oyster online. Online. http://oyster.tfl.gov.uk/.

[31] Ov-chipkaart. Ov-chipkaart. Online. http://www.ov-chipkaart.nl/.

[32] Cristian Palma. Chileno asegura haber hackeado la tarjeta bip del transantiago para cargarla gratis, 10 2013. http://www.guioteca.com/internet/chileno-asegura-haber-hackeado-la-tarjeta-bip-del-transantiago-para-cargarla-gratis/.

[33] portalnet.cl. Analizando tarjeta bip! de transantiago. Online, 10 2014.

[34] Publimetro. Sujeto detenido por vender tarjetas bip! adulteradas. Online, 6 2014. http://www.publimetro.cl/nota/cronica/sujeto-detenido-por-vender-tarjetas-bip-adulteradas/xIQnfc!5FTVWXiW5T2OY/#.

[35] Ricardo Francino Saldivia. Hacker viola tarjeta bip! y dice que se puede viajar gratis. Online, 11 2013. http://noticias.terra.cl/chile/,45a9bb790c742410VgnVCM5000009ccceb0aRCRD.html.

[36] Ronny Wichers Schreur, Peter Van Rossum, Flavio Garcia, Wouter Teepe, Bart Jacobs, Gerhard De Koning Gans, Roel Verdult, Ruben Muijrers, Ravindra Kali, and Vinesh Kali. Security flaw in mifare classic. 2008.

[37] Skyetek. Using mifare classic tags. Online. www.skyetek.com/docs/m2/mifareclassic.pdf.

[38] TNE.cl. Junaeb entregará nueva tne a todos los estudiantes de educación media y superior durante 2015. Online, 2014. http://tde.tne.cl/noticias/junaeb-entregara-nueva-tarjeta-nacional-estudiantil-tne-a-todos-los-estudiantes-de-ed ucacion-media-y-superior-durante-2015/.

[39] TNO. Security analysis of the dutch ov-chipkaart. *Technical report TNO report 34643*, 2008.

[40] Transperth. Smartriders. Online. http://www.transperth.wa.gov.au/.

[41] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. pages 19–35, 2005.

Capítulo 7

Anexos

7.1. Anexo 1: Representación sector 0

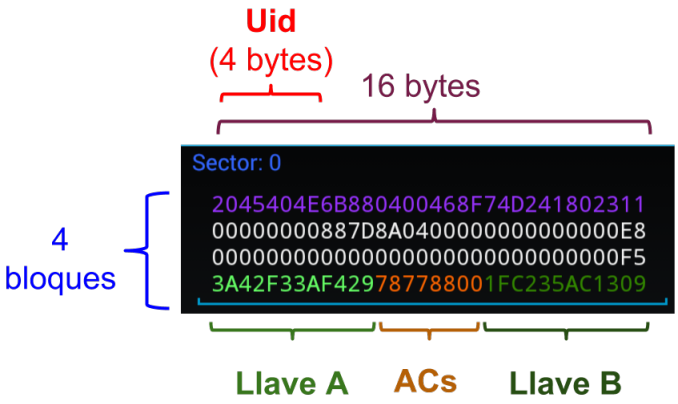

Figura 7.1: Representación del sector 0 de una tarjeta.

7.2. Anexo 2: Protocolo de anticolisión y autentificación.

N	Elemento	Bytes	Significado	Proceso
1	Lector	`26`	Req	**Anticolisión**
2	Tarjeta	`04 00`	Respuesta req	
3	Lector	`93 20`	Selección	
4	Tarjeta	`2a 69 8d 43 8d`	Envío UID tarjeta	
5	Lector	`93 70 2a 69 8d 43 8d 52 55`	Selección del uid	
6	Tarjeta	`08 b6 dd`	Tipo de tarjeta	
7	Lector	`60 04 d1 3d`	Autentificación en un bloque	**Autentificación**
8	Tarjeta	`3b ae 03 2d`	Desafío tarjeta	
9	Lector	`c4! 94 a1d2 6e! 96 86! 42`	Respuesta desafío tarjeta más desafío lector	
10	Tarjeta	`84 66! 05! 9e!`	Respuesta desafío lector	

Tabla 7.1: Flujo de bits que ejemplifican protocolos de anticolisión y autentificación.

7.3. Anexo 3: Especificaciones lector SCL3711

Interfaces		
Host	USB	• USB 2.0 • PC/SC v2.01 • Capa de Abstracción de Hardware (HAL) por sobre la API PC/SC. Común a todos los lectores SCM
Tarjetas inteligentes	Sin contacto	• Antena ISO/IEC 14443 • Hasta 848 Kbps • Soporta: - Type A until ISO/IEC 14443-3 - Type B until ISO/IEC 14443-3 - ISO/IEC 14443-4 - MIFARE: Classic 1K y 4K, Mifare plus, DESFire, Ultralight - my-d™ NFC - FeliCa™ - NFC forum tag tipo 1, 2, 3, 4 - ISO/IEC 18092 peer-to-peer protocolo hasta 412 Kbps
Humano	Optico	LED para estado de conexión
Firmware		No actualizable
Poder	USB	Potenciada vía Bus
Temperatura Operación	[C]	-20 a 70
Temperatura Almacenamiento	[C]	-40 a 85
Dimensiones	La x An x Al [mm]	66 x 24 x 8
Peso	[g]	10
Aprobaciones		• USB, CE, FCC, VCCI, WEEE, RoHS, WHQL • Frecuencia de Radio de Japan
Soporte		Kit de desarrollo de software

Tabla 7.2: Especificaciones lector SCL3711

7.4. Anexo 4: Especificaciones de la primera estación de trabajo

Información General	Fabricante	Dell, Inc
	Nombre del producto	Inspiron 14 N4030 Notebook
	Modelo del producto	N4030
	Tipo del producto	Notebook
	Linea del producto	Inspiron
	Serie del producto	14
Memoria	Memoria RAM	3 GB
	Tecnología de la Memoria	DDR3 SDRAM
Procesador	Fabricante del procesador	Intel
	Tipo del procesador	Core i3
	Modelo del procesador	i3-370M
	Velocidad del procesador	2.40 GHz
	Núcleos del procesador	Dual-core (2 Core)
	Cache	3 MB
Almacenamiento	Tamaño de Almacenamiento	500 GB
	RPM Disco Duro	5400
Sistema operativo	Nombre	Ubuntu 12.04
	Arquitectura	64-bit

Tabla 7.3: Especificaciones de la primera estación de trabajo

7.5. Anexo 5: Especificaciones de la segunda estación de trabajo

Información General	Fabricante	Lenovo
	Nombre del producto	ThinkPad Edge E420
	Modelo del producto	E420
	Tipo del producto	Notebook
	Linea del producto	ThinkPad
	Serie del producto	Edge
Memoria	Memoria RAM	2 GB
	Tecnología de la Memoria	DDR3
Procesador	Fabricante del procesador	Intel
	Tipo del procesador	Core i5
	Modelo del procesador	2520M
	Velocidad del procesador	2.50GHz
	Núcleos del procesador	Dual-core (2 core)
	Cache	3 MB
Almacenamiento	Tamaño de Almacenamiento	500 GB
	RPM Disco Duro	7200rpm
Sistema Opertivo	Nombre	Ubuntu 14.04
	Arquitectura	64-bit

Tabla 7.4: Especificaciones de la segunda estación de trabajo.

7.6. Anexo 6: Especificaciones Moto X

Red	GSM 850 / 900 / 1800 / 1900 - HSDPA 850 / 900 / 1900 / 2100 - LTE 700 / 850 / 1700 / 1900 / 2100
Tamaño	129.3 x 65.3 x 10.4 mm
Peso	130 g
RAM	2GB RAM
Procesador	Procesador Qualcomm Snapdragon S4 Pro dual-core 1.7 GHz, GPU Adreno 320
Almacenamiento	16 GB
NFC	Sí
Bluetooth	Sí, v4.0

Tabla 7.5: Especificaciones Smartphone Moto X (primera generación)

7.7. Anexo 7: Tipos de AC usados en las tarjetas *bips!*

Valor AC	Sectores que los ocupan	Implicancias de dichos AC
`78 77 88`	0, 3, 10	- 3 primeros bloques y AC leíbles por ambas claves, editables sólo con la clave B - Ambas claves no leíbles como datos, editables sólo con la clave B
`FF 07 80`	1, 2, 12, 13, 14, 15	- 3 primeros bloques leíbles y editables por ambas claves. - Clave A no leíble como dato. - AC y Clave B leibles y editables por clave B
`7E 17 88`	4, 5	- Primer bloque y AC leíbles por ambas claves, editables por clave B. - Segundo y tercer bloque leíbles y editables por ambas claves. - Ambas claves no leíbles como datos, editables sólo con la clave B
`18 77 8E`	6, 7, 8	- 3 primeros bloques leíbles y editables por ambas claves. - Segundo y tercer bloque usado como bloques de valor. - AC leíbles por ambas claves, editables sólo por clave B. - Ambas claves no leíbles como datos, editables sólo con la clave B.
`7C 37 88`	9	- Primeros dos bloques y AC leíbles por ambas claves, editables sólo por clave B. - Tercer bloque leíble y editable por ambas claves. - Ambas claves no leíbles como datos, editables sólo con la clave B.
`7F 07 88`	11	- 3 primeros bloques leíbles y editables por ambas claves. - AC leíbles por ambas claves, editables sólo porclave B. - Ambas claves no leíbles como datos, editables sólo con la clave B.
`00 F0 FF`	4 y 9 (en bloqueo por Bus)	- Sólo AC leible con ambas claves - Todo lo demás ni leíble ni editable.

7.8. Anexo 8: Ejemplo de datos MIFARE Classic 1K

Lectura ejemplo de una tarjeta MIFARE Classic de 1K, correspondiente a una *tarjeta no vinculada a un usuario*. Las primeras filas están ocultas para anonimizar la información referente a la tarjeta en cuestión.

```
000: ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
001: 00 00 00 00 ** ** ** ** 00 00 00 00 00 00 00 **
002: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
003: 3a 42 f3 3a f4 29 78 77 88 00 1f c2 35 ac 13 09
004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
007: 63 38 a3 71 c0 ed ff 07 80 00 24 3f 16 09 18 d1
008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
011: f1 24 c2 57 8a d0 ff 07 80 00 9a fc 42 37 2a f1
012: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
013: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
014: 01 e1 cc e7 09 00 00 00 00 00 00 00 00 00 00 62
015: 32 ac 3b 90 ac 13 78 77 88 00 68 2d 40 1a bb 09
016: 11 22 20 49 d7 06 5f 19 90 74 79 21 00 40 00 7c
017: 70 6b 00 00 00 00 00 00 00 00 00 40 08 84 04 01
018: 70 6b 00 00 00 00 00 00 00 00 00 40 08 84 04 01
019: 4a d1 e2 73 ea f1 7e 17 88 00 06 7d b4 54 54 a9
020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
021: 40 de 47 cf 03 00 20 52 82 04 00 02 00 04 04 91
022: 40 de 47 cf 03 00 20 52 82 04 00 02 00 04 04 91
023: e2 c4 25 91 36 8a 7e 17 88 00 15 fc 4c 76 13 fe
024: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
025: 00 00 00 00 ff ff ff ff 00 00 00 00 19 e6 19 e6
026: 00 00 00 00 ff ff ff ff 00 00 00 00 1a e5 1a e5
027: 2a 3c 34 7a 12 00 18 77 8e 00 68 d3 02 88 91 0a
028: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
029: 00 00 00 00 ff ff ff ff 00 00 00 00 1d e2 1d e2
030: 00 00 00 00 ff ff ff ff 00 00 00 00 1e e1 1e e1
031: 16 f3 d5 ab 11 39 18 77 8e 00 f5 9a 36 a2 54 6d
032: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
033: c4 04 00 00 3b fb ff ff c4 04 00 00 90 6f 90 6f
034: c4 04 00 00 3b fb ff ff c4 04 00 00 90 6f 90 6f
035: 93 7a 4f ff 30 11 18 77 8e 00 64 e3 c1 03 94 c2
036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
037: 00 00 00 00 00 00 00 00 f0 05 54 02 37 00 00 b5
038: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
039: 35 c3 d2 ca ee 88 7c 37 88 00 b7 36 41 26 14 af
040: 5f b2 97 56 47 04 00 40 19 a0 0f 00 00 00 00 74
041: 5f b8 d7 04 5a 04 00 40 19 40 1f 00 00 00 00 96
042: 95 b0 27 d3 50 cc b8 67 19 a0 0f 40 00 00 00 a2
043: 69 31 43 f1 03 68 78 77 88 00 32 4f 5d f6 53 10
044: c4 b2 b7 02 33 c0 26 00 46 19 00 0a 40 f0 39 e4
045: 04 ce a7 0c 23 c0 26 00 46 19 00 0a 40 8c 57 96
046: 41 de 17 cf 23 80 94 20 45 19 50 0a 00 00 00 fc
047: a3 f9 74 28 dd 01 7f 07 88 00 64 3f b6 de 22 17
048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
049: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
051: 63 f1 7a 44 9a f0 ff 07 80 00 82 f4 35 de df 01
052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
053: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
054: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
055: c4 65 2c 54 26 1c ff 07 80 00 02 63 de 12 78 f3
056: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
057: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
058: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
059: d4 9e 28 26 66 4f ff 07 80 00 51 28 4c 36 86 a6
060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
061: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
062: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
063: 3d f1 4c 80 00 a1 ff 07 80 00 6a 47 0d 54 12 7c
```

7.9. Anexo 9: Ejemplo de datos MIFARE Classic 4K

Lectura ejemplo de una tarjeta MIFARE Classic de 4K, correspondiente a una TNE del año 2015. Las primeras filas están ocultas para anonimizar la información referente a la tarjeta en cuestión.

```
000: ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
001: 00 00 00 00 ** ** ** ** 00 00 00 00 00 00 00 **
002: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
003: 3a 42 f3 3a f4 29 78 77 88 00 1f c2 35 ac 13 09
004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
007: 63 38 a3 71 c0 ed ff 07 80 00 24 3f 16 09 18 d1
008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
011: f1 24 c2 57 8a d0 ff 07 80 00 9a fc 42 37 2a f1
012: ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
013: ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
014: 03 21 de e7 0f 00 00 00 00 00 00 00 00 00 00 c6
015: 32 ac 3b 90 ac 13 78 77 88 00 68 2d 40 1a bb 09
016: 11 24 60 88 57 2b 5f 19 40 7b d8 60 7d c0 00 6d
017: b4 7e 00 00 20 c4 eb c0 00 00 00 40 08 84 04 26
018: b4 7e 00 00 20 c4 eb c0 00 00 00 40 08 84 04 26
019: 4a d1 e2 73 ea f1 7e 17 88 00 06 7d b4 54 54 a9
020: 5e 7d 2c fa ff 8f 07 fc ff 4f 0d 00 00 00 00 c3
021: 00 b6 47 6f 13 00 20 32 86 24 00 00 00 04 04 da
022: 00 b6 47 6f 13 00 20 32 86 24 00 00 00 04 04 da
023: e2 c4 25 91 36 8a 7e 17 88 00 15 fc 4c 76 13 fe
024: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
025: 00 00 00 80 ff ff ff 7f 00 00 00 80 19 e6 19 e6
026: 00 00 00 80 ff ff ff 7f 00 00 00 80 19 e6 19 e6
027: 2a 3c 34 7a 12 00 18 77 8e 00 68 d3 02 88 91 0a
028: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
029: 00 00 00 00 ff ff ff ff 00 00 00 00 1d e2 1d e2
030: 00 00 00 00 ff ff ff ff 00 00 00 00 1e e1 1e e1
031: 16 f3 d5 ab 11 39 18 77 8e 00 f5 9a 36 a2 54 6d
032: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
033: c8 05 00 00 37 fa ff ff c8 05 00 00 73 8c 73 8c
034: c8 05 00 00 37 fa ff ff c8 05 00 00 73 8c 73 8c
035: 93 7a 4f ff 30 11 18 77 8e 00 64 e3 c1 03 94 c2
036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
037: 9e 17 00 30 01 00 00 00 f0 09 60 0b 37 00 00 dd
038: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
039: 35 c3 d2 ca ee 88 7c 37 88 00 b7 36 41 26 14 af
040: df b5 a7 40 4b 04 00 40 19 a0 0f 00 00 00 00 85
041: 1f b4 57 21 54 04 00 40 19 40 1f 00 00 00 00 db
042: df b5 a7 3e 5d 04 00 40 19 a0 0f 00 00 00 00 99
043: 69 31 43 f1 03 68 78 77 88 00 32 4f 5d f6 53 10
044: c2 b5 57 23 2e 00 a9 44 46 7d 48 03 40 f4 dc cf
045: 01 b6 17 6f 33 80 8c 21 45 7d 48 03 00 00 00 bc
046: c1 b5 a7 c4 24 40 45 52 45 7d 48 03 00 00 00 63
047: a3 f9 74 28 dd 01 7f 07 88 00 64 3f b6 de 22 17
048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
049: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
051: 63 f1 7a 44 9a f0 ff 07 80 00 82 f4 35 de df 01
052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
053: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
054: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
055: c4 65 2c 54 26 1c ff 07 80 00 02 63 de 12 78 f3
056: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
057: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
058: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
059: d4 9e 28 26 66 4f ff 07 80 00 51 28 4c 36 86 a6
060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
061: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
062: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
063: 3d f1 4c 80 00 a1 ff 07 80 00 6a 47 0d 54 12 7c
064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
065: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
066: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
067: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
071: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
072: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
073: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
075: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
076: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
077: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
079: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
```

```
080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
081: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
083: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
084: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
085: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
086: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
087: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
089: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
091: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
092: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
093: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
094: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
095: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
097: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
098: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
099: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
101: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
102: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
103: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
104: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
105: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
106: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
107: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
108: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
109: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
111: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
112: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
113: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
114: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
115: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
116: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
117: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
118: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
119: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
121: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
122: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
123: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
124: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
125: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
126: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
127: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
128: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
129: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
131: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
132: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
133: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
134: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
135: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
136: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
137: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
138: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
139: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
141: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
142: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
143: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
144: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
145: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
146: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
147: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
148: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
149: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
151: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
152: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
153: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
154: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
155: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
156: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
157: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
158: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
159: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
161: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
162: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
163: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
164: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
165: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
166: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
167: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
168: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
169: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
171: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
172: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
173: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
174: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
175: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
176: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
177: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
178: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
179: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
181: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
182: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
183: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
184: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
185: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
186: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
187: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
188: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
189: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
191: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
192: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
193: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
194: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
195: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
196: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
197: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
198: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
199: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
201: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
202: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
203: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
204: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
205: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
206: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
207: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
208: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
209: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
211: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
212: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
213: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
214: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
215: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
216: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
217: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
218: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
219: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
221: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
222: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
223: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
224: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
225: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
226: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
227: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
228: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
229: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
231: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
232: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
233: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
234: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
235: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
236: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
237: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
238: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
239: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
241: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
242: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
243: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
244: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
245: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
246: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
247: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
248: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
249: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
251: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
252: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
253: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
254: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
255: ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
```