Who we are

- bughardy

  (aka Matteo Beccaro)

  bughardy@cryptolab.net

  Italian student with passion of IT, networking and pentesting. In 2013 ended his studies in high school and apply for Politecnico of Turin at Computer Engineering.

- Eagle1753

  (aka Matteo Collura)

  eagle1753@onenetbeyond.org

  Italian student, applied for Politecnico of Turin, Electronic Engineering. Has a great passion for Physics. He is studying with bughardy on WiFi networks and security. Loves to solve challenges.

History of NFC hacks

- 2008 NFC MIFARE CLASSIC exploit, further in following years.
- 2011 first hack of NFC ULTRALIGHT transport system by U.S. Researchers using the RESET ATTACK
- 2013 a new hack of NFC ULTRALIGHT transport system made by us. We called it LOCK ATTACK.

What is MIFARE chip?

RFID chip designed to work at 13.56MHz.There are millions of MIFARE chip cards worldwide and they belong to several variants:

- MIFARE CLASSIC
- MIFARE ULTRALIGHT
- MIFARE ULTRALIGHT C
- MIFARE DESFIRE
- etc

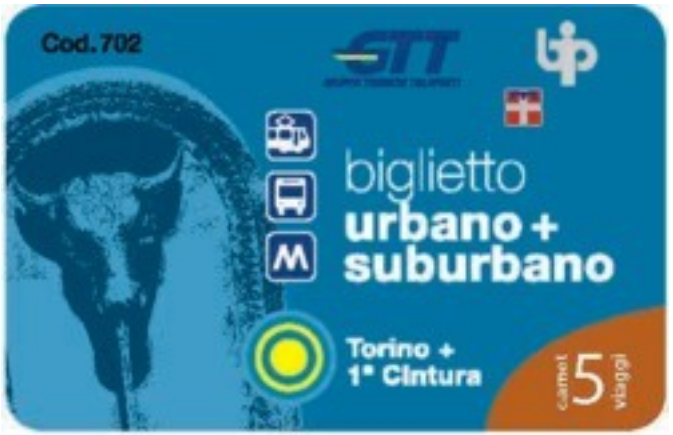
More details of: MIFARE CLASSIC vulnerabilities

- Security through obscurity is not security; algorithm has been reversed.
- By eavesdropping the communication an attacker might recover keys in few minutes.
- Default keys let an attacker recover all the other keys in few seconds without the need of eavesdropping any communication.

More details of: MIFARE ULTRALIGHT vulnerabilities

- Do not have encryption to keep them low cost
- Often transport companies use a bad implementation
- They often don't use OTP data to store rides but instead they do on a r/w memory: DATA sector.
- That is the RESET ATTACK

Yes, but what is MIFARE ULTRALIGHT?



How is it composed?

Page Address	**Byte number**				
Decimal	Hex	0	1	2	3
0	0x00	**UID**			
1	0x01	**UID**			
2	0x02	**UID**	**INTERNAL**	**LOCK BYTE**	**LOCK BYTE**
3	0x03	**OTP**	**OTP**	**OTP**	**OTP**
4 to 15	0x04 to 0x0F	**DATA**			

What is UID?

- 7 bytes serial number
- 2 check bytes obtained by XORing the previous bytes in this way:

  1st byte: CT ⊕ SN0 ⊕ SN1 ⊕ SN2

  2nd byte: SN3 ⊕ SN4 ⊕ SN5 ⊕ SN6

- Programmed by manufacturer, so they're read only

What is OTP?

- Only security function in MIFARE ULTRALIGHT tickets
- 4 bytes, all 00 at first (by default)
- OR operation prevents from turning a bit from 1 to 0 again
- Used for storing rides (just need to turn a bit from 0 into 1). The stamping machine checks the number of “0” left.

What is LOCK sector?

- 2 bytes

L - 7	L - 6	L - 5	L - 4	L - OTP	BL – 10 to 15	BL – 4 to 9	BL – OTP
L – 15	L – 14	L – 13	L – 12	L – 11	L – 10	L – 9	L – 8

- Each bit can turn 1 page (4 bytes) into read-only mode
- The last 3 bits of first lock byte freeze the bits of the lock bytes themselves

What is DATA sector?

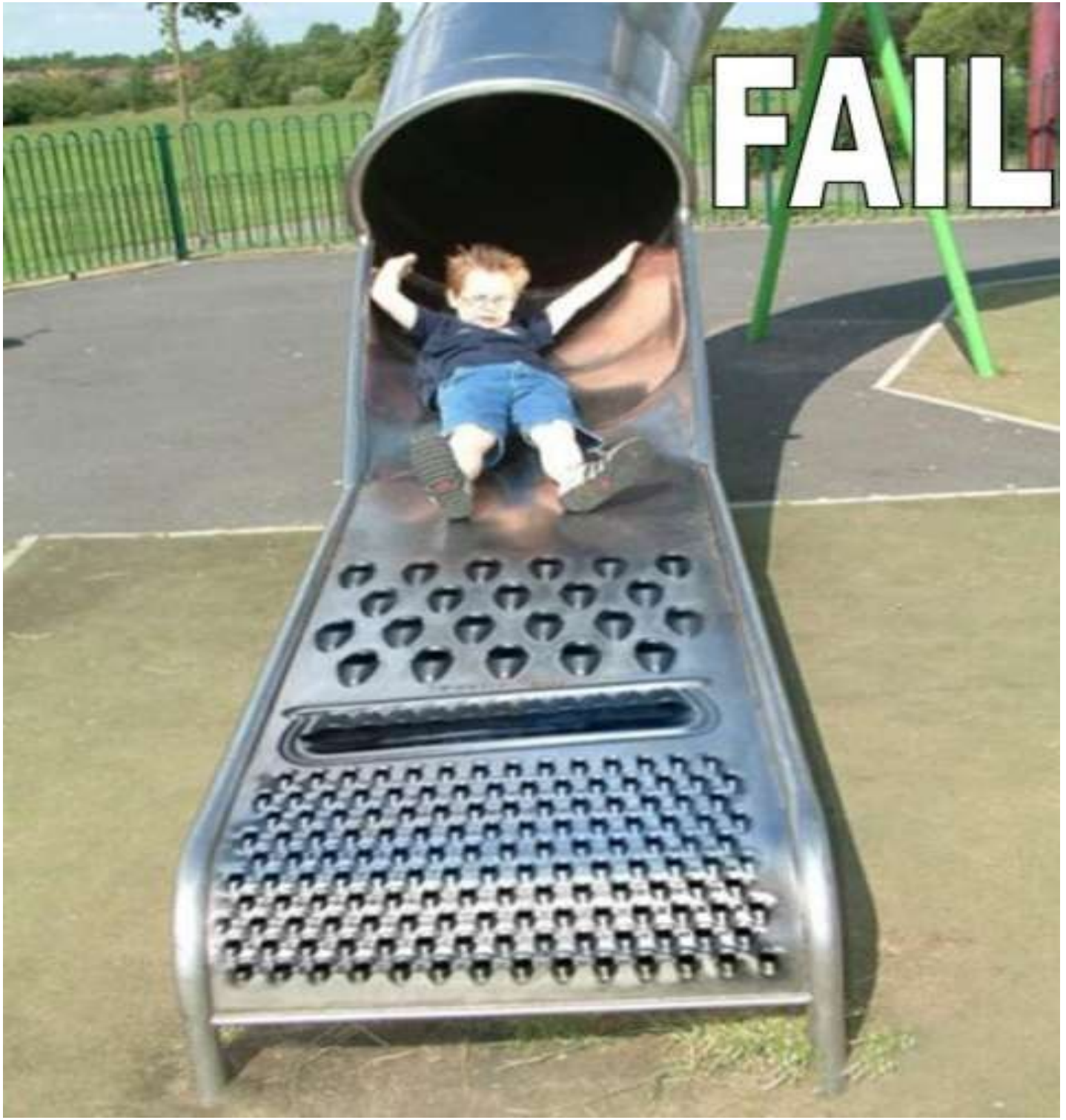
- Biggest sector, 48 bytes
- It stores details like time (of last stamp), date, station ID, etc
- In the reset attack, it is used to store the number of rides left.

Regarding DATA sector

- Working still in progress.
- Decoding how and which data are encoded to the ticket.
- We will provide dumps and info (in the Q&A session) if you would like to help us.

The history of an hack

- First tests, without knowing how OTP was working.
- OTP contains the number of rides left!!
- Attempt to write something over OTP.


FAIL

There is still a long way

- “One the roa.. Er.. On the bus” test!
- Stamping more tickets one after the other and looking and comparing their dumps
- Empiric results about how data is stored on tickets

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

“On the road” tests..

- Some empirical results in DATA sector decoding:

BYTES	DESCRIPTION	EXAMPLE
0-24 bytes	Locked DATA	01 04 00 00 02 01 02 BE 40 05 AF 00 00 AE 10 A0 61 03 1C 1C B2 2B 61 8E
25-28	Stamping progressive number	43 3B (7B 00)
29-32	Validator ID (guessed) / or Ticket type	04 F8 00 00
33-36	Stamping progressive numer	43 3B (7B 00)
37-38	Still not guessed	00 3B 00 04
39-40	Ticket type (guessed) / or data	F8 AE
41-48	Time data (guessed)	10 7B B3 02 E6 56

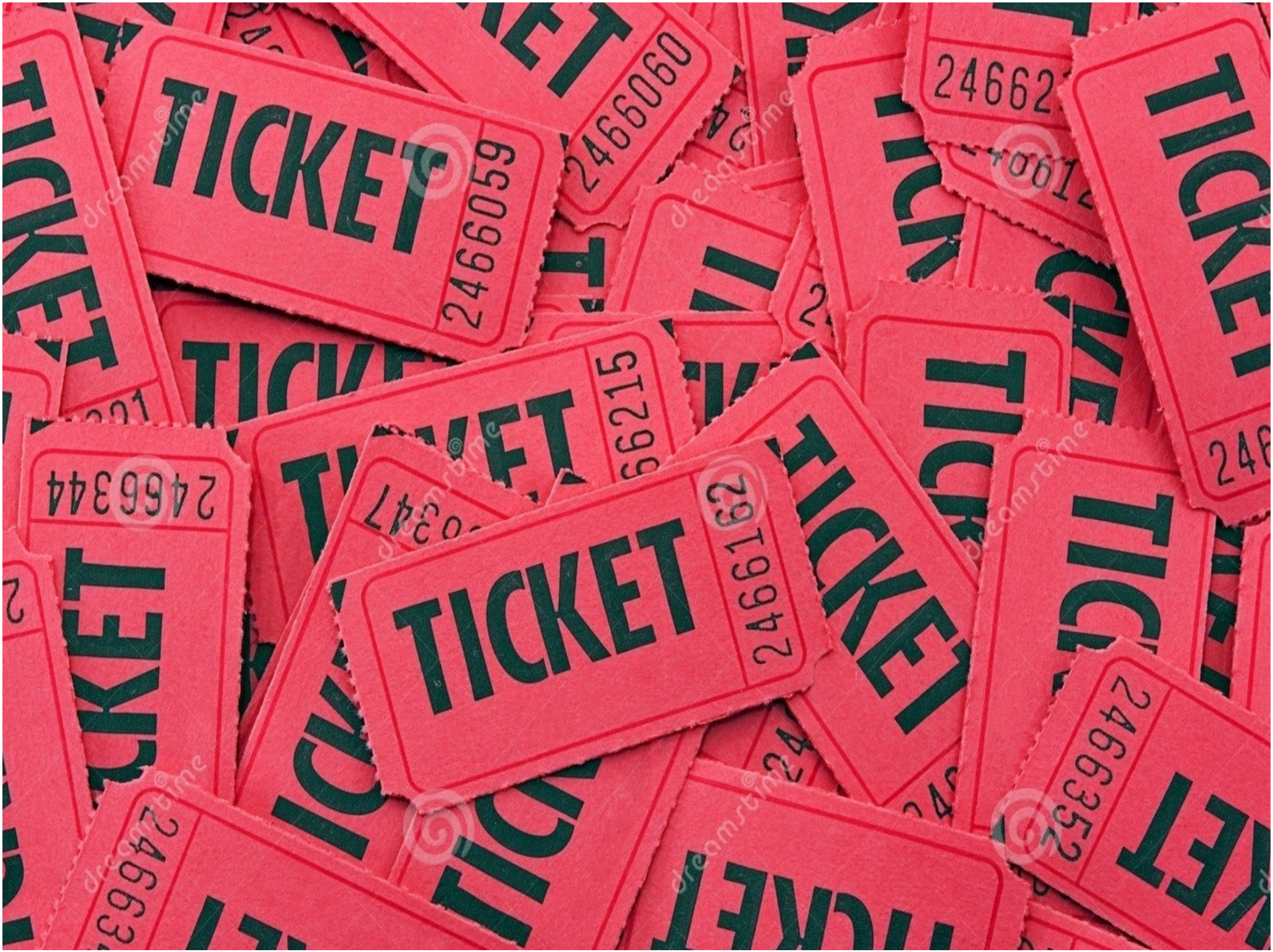
Seize the day

- Assume that you know where the time (of the last stamp) is stored and how
- Use a NFC phone / NFC reader to change that field (it is in the data field so there are no problems)
- It isn’t so reliable and now we aren’t able to deal with this.

Mission Completed

- Preventing the machine to write the number of rides left would turn the ticket into an unlimited one.

- The answer is: LOCK BYTES


TICKET
2466059
2466060
TICKET
2466215
TICKET
2466344
TICKET
2469162
TICKET
2466352

The LOCK ATTACK: Why?

- Locking the OTP sector we prevent the stamping machine from removing rides stored on our ticket.
- Each time we stamp the ticket the validator checks if we have rides left
- If so it writes on DATA sector data time, etc and tries, without success, to turn bit from 0 to 1 in OTP sector.
- However...

Ops...

Yes, it is not okay to have always 5 rides on a 5 rides-ticket...

LOL

How to fix it?

- LOCK ATTACK would be easy to be fixed.
  - Firmware update: check whether OTP sector is locked or not, if so, just refuse to validate the ticket.
  - Firmware update: try to unlock the sector, but only if block bits are not enabled.

- TIME ATTACK isn't really easy to be fixed.
  - Communication between validator and ticket is not encrypted: easy to be sniffed.
  - Solution: Implementing an encrypted communication

Future works...

We are actually working on:

- Rewrite the tool in C/C++ without using external tools
- Decoding DATA sector: dumps and infos are available in Q&A section to anyone who would like to help us.
- NFC-enabled phone or a proxymark for further studying.

Questions?