



# ST25TB series NFC tags for fun in French\* public transports

Benjamin `gentilkiwi` DELPY

\*maybe elsewhere too



# Mandatory notice

- ❖ This presentation is not intended to help to fraud in public transports ;
  - ❖ There is no hack, no CVE, no bypass of intended security features ;
  - ❖ All this presentation is about normal behavior of ST25TB and SRx cards and their reading system ;
  - ❖ Research about ST25TB usages was not exhaustive, even in France



# Reloadable convenience tickets

- ◊ This is NOT about season tickets
  - ◊ Usually, in France, Calypso (Innovatron) or Desfire (NXP) related technologies are used
  - ◊ but not only!
  - ◊ The card/chip is far more secure, but also more expensive
    - ◊ Not always usable for convenience tickets, even reloadable
- ◊ This is NOT about magnetic tickets (still exist!), QR code or payment at usage with debit cards
  - ◊ Good news, debit cards for ticket transport validation are more and more common
- ◊ This is about convenience tickets, usually:
  - ◊ NFC chip in ‘flexible’ paper tickets
  - ◊ Reloadable multiple times (4, 10, etc..)
  - ◊ Multiple offers depending on cities, but often seen:
    - ◊ 1 ride ;
    - ◊ 2 rides ;
    - ◊ 10 rides ;
    - ◊ Unlimited rides for 1 day ;
    - ◊ Unlimited rides for the week ;
    - ◊ etc.
  - ◊ Supports are valid for some years



# ST25TB series NFC tags

- ❖ The ST25TB series of RFID tags are compatible with the ISO14443 standard, so support applications such as public transport and event ticketing.
- ❖ They provide state-of-the-art RF performance and include a counter able to count more than 4 billion events.
  - ❖ ISO14443-2 Type B with Proprietary Protocol
  - ❖ 512-, 2K-, and 4K-bit EEPROM with write protect
  - ❖ Two 32-bit counters with anti-tearing feature
  - ❖ OTP bytes with conditional erased features

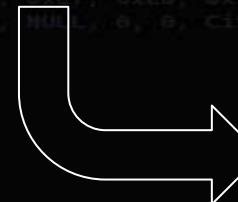




# ST25TB512-AT

The ST25TB series, and its ancestor SRx, has multiple products available

- ◊ SR\* – obsolete (~2002/2003...)
- ◊ SR176 (176 bits) - *the most basic one (never seen)*
- ◊ memory area with configurable bits locks
- ◊ SRI512, SRIX512, SRI2K, SRI4K, SRIX4K (for 512 bits, 2 Kbits and 4Kbits)
  - ◊ memory area with configurable locks bits
  - ◊ resettable one-time programming areas
    - ◊ 2 x count down counters, with anti tear down, one counter can be used to reload some OTP part
    - ◊ *SRIX\* have « France Telecom proprietary anti-clone function », only cards of the series supporting authentication (challenge/response)*
- ◊ **SRT512** (for 512 bits)
  - ◊ memory area with configurable locks bits
  - ◊ 2 x count down counters, with anti tear down
- ◊ **Designed for transports**
- ◊ ST25TB\* (~2016...)
  - ◊ Replacement for SRI\*
  - ◊ No (public ?) traces for a SRIX replacement
- ◊ **ST25TB512-AT**
  - ◊ Replacement for SRT\*



Addr	32 bits block	Description
0	User area	Lockable EEPROM
1		
2		
3		
4		
5		
6	32 bits binary counter	Count down counter
7	User area	Lockable EEPROM
8		
9		
10		
11		
12		
13		
14		
15		
255	OTP_Lock_Reg, 1, ST Reserved	System OTP bits
UID0	64 bits UID area	ROM
UID1		



# How to read them?



- ❖ Nearly all 14B capable readers can read (and write) to them, this include:

- ❖ The infamous ACS - ACR122U

- ❖ Proxmark3

- ❖ (now that we understand some PCB problems, even the RDV4 ;))

- ❖ DL533N

- ❖ etc.

- ❖ Usually, all NXP PN53x based readers are compatible (but ST and TI chipsets are also in the game)





# How to read them?

## libnfc : nfc-st25tb

```
c:\security>nfc-st25tb
| mode   : info
Reader  : NXP / PN533 - via pn53x_usb:bus-
0:\.\libusb0-0255--0x04cc-0x2533
    ...wait for card...
Target  : ISO/IEC 14443-2B ST SRx (106 kbps)
UID     : d5 b4 fb 7d 78 33 02 d0
Manuf   : 0x02 - STMicroelectronics
ChipId  : 0x33 - ST25TB512-AT
Serial  : 0x787dfbb4d5
|blk sz : 32 bits
|nb blks: 16
|sys idx: 255
[0x00] 11 01 00 00
[0x01] 24 00 00 25
[0x02] 8e 06 80 15
[0x03] 00 01 43 39
[0x04] 00 00 c0 85
[0x05] 02 00 00 0a
[0x06] fe ff ff ff
[0x07] 00 00 00 00
[0x08] 00 00 00 00
[0x09] 00 00 00 00
[0x0a] 5a 17 00 00
[0x0b] 00 00 00 00
[0x0c] 00 00 00 00
[0x0d] 00 00 00 00
[0x0e] 00 00 00 00
[0x0f] d4 4e bc e4
[0xff] ff ff ff ff
| ST reserved   : 111111111111111
| b15           : 1 - not OTP (?)
| OTP_Lock_Reg : 111111111111111
```



# How to read them?

## libnfc : nfc-st25tb

```
c:\security>nfc-st25tb -h
```

Usage:

```
nfc-st25tb [-i]
nfc-st25tb -b N -r
nfc-st25tb -b N [-r] -w ABCD[EF01]
nfc-st25tb -h
```

Options:

```
-i          (default) information mode - will try to dump the tag content and display informations
-b N        specify block number to operate on (tag dependent), needed for read (-r) and write (-w) modes
-r          read mode - will try to read block (specified with -b N parameter)
-w ABCD[EF01] write mode - will try to write speciced data (2 or 4 bytes depending on tag) to block (specified
with -b N parameter)
-h          this help
```

Examples:

```
nfc-st25tb -i
    Display all tag informations
nfc-st25tb -b 0x0e -r
    Read block 0x0e (14) of the tag
nfc-st25tb -b 0x0d -w 0123abcd
    Write block 0x0d (13) of the tag with hexadecimal value '01 23 ab cd'
nfc-st25tb -b 0x0c -r -w 0123abcd
    Read, then write block 0x0c (12) of the tag with hexadecimal value '01 23 ab cd'
```

Warnings:

Be careful with: system area, counters & otp, bytes order.



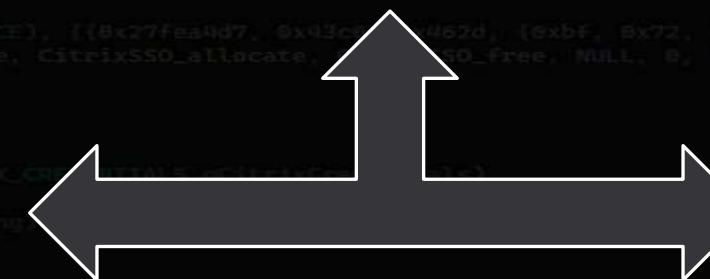
# How to read them?

## Proxmark3

```
[usb] pm3 --> hf 14b info
[+] UID: D0 02 33 78 7D FB B4 D5
[+] MFG: 02, ST Microelectronics SA France
[+] Chip: 0C, SRT512
[usb] pm3 --> hf 14b dump --ns
[+] found a SRT512 tag
[=] reading tag memory from UID
D00233787DFBB4D5
[=] .....
[+] SRT512 tag
[=] block# | data          | lck | ascii
[=] -----+-----+-----+
[=] 0/0x00 | 11 01 00 00 |   | ....
[=] 1/0x01 | 24 00 00 25 |   | $..%
[=] 2/0x02 | 8E 06 80 15 |   | ....
[=] 3/0x03 | 00 01 43 39 |   | ..C9
[=] 4/0x04 | 00 00 C0 85 |   | ....
```



# How to understand them?



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

*Example in Lille, other layouts are available, and used*



# How to understand them'

- ◆ Intercode & Intertic @ AFNOR

◆ *Intercode*

◆ *NF P99-405-1, NF P99-405-2, NF P99-405-3, NF P99-405-4, NF P99-405-5 & XP P99-405-6*

◆ Intertic

◆ NF P99-410

❖ FD P99-416 - E-ticketing for the transportation sector - Interoperability rules for the codification of e-ticketing data - Contactless tickets - FD INTERTIC - 597,12 € HT

❖ Chapitre 9. SRT512 - ST25TB512-AT



# How to understand them?

(a very old draft is available on the internet)

## [DISTRIBUTION DATA]

0x250 = FR

Company/City, Version, End of  
ticket validity, Tariff, etc.

COUNTER1(CNT1)  
0x000001 ride left

COUNTER2(CNT2)  
?

? CERTIFICATE ?

DATA CERTIFICATE

[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	01	00	00	0a
[0x06]	fd	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	b8	40	d2	b6
[0x0c]	64	80	b3	c2
[0x0d]	24	85	bc	04
[0x0e]	11	12	00	41
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff

## [SEDENTARY BLOCK]

ProductID, Key information, etc.

RELOADING1(RLD1)  
0xa reloading left

RELOADING2(RLD2)  
?

[USAGE DATA]  
Validation information,  
timestamp, etc.

DISTRIBUTION CERTIFICATE

# How to understand them?

## Included in Proxmark3: proxmark3/client/pyscripts



[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	01	00	00	0a
[0x06]	fd	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	b8	40	d2	b6
[0x0c]	64	80	b3	c2
[0x0d]	24	85	bc	04
[0x0e]	11	12	00	41
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff



# How to understand them?

- ❖ CERTIFICATE parts are ~a signature on the card data with an external key

- ❖ usually on an external SAM card in vending & validating machines

- ❖ 2 to 4 keys can exist:

- ❖ DISTRIBUTION

- ❖ Create signature – (normally) only on vending/loading machines

- ❖ Validate signature

- ❖ USAGE

- ❖ Create signature

- ❖ Validate signature

- ❖ CERTIFICATE are UID dependent

- ❖ An exact copy on another (full) card will not have valid CERTIFICATE validation – UID is not the same



# What can we do to play with?

- ❖ Reload a dump of a full card on the original card?

Saved dump  
before usage

```
[0x00] 11 01 00 00
[0x01] 24 00 00 25
[0x02] 8e 06 80 15
[0x03] 00 01 43 39
[0x04] 00 00 c0 85
[0x05] 02 00 00 0a
[0x06] fe ff ff ff
[0x07] 00 00 00 00
[0x08] 00 00 00 00
[0x09] 00 00 00 00
[0x0a] 5a 17 00 00
[0x0b] 00 00 00 00
[0x0c] 00 00 00 00
[0x0d] 00 00 00 00
[0x0e] 00 00 00 00
[0x0f] d4 4e bc e4
[0xff] ff ff ff ff
```



```
[0x00] 11 01 00 00
[0x01] 24 00 00 25
[0x02] 8e 06 80 15
[0x03] 00 01 43 39
[0x04] 00 00 c0 85
[0x05] 01 00 00 0a
[0x06] fd ff ff ff
[0x07] 00 00 00 00
[0x08] 00 00 00 00
[0x09] 00 00 00 00
[0x0a] 5a 17 00 00
[0x0b] 00 00 00 00
[0x0c] 00 00 00 00
[0x0d] 00 00 00 00
[0x0e] 00 00 00 00
[0x0f] d4 4e bc e4
[0xff] ff ff ff ff
```

- ❖ **No:** counters cannot be refilled at same values (new value is > to current one)
  - ❖ 0xa000002 is > to 0xa000001
  - ❖ 0xfffffff is > to 0xffffffff



# What can we do to play with?

- ❖ Reload a dump of a fcc11 card, the original card?  
**Tear-off attacks exist!**

See the wonderful work of Philippe Teuwen & Christian Herrmann with the Proxmark3

- <https://blog.quarkslab.com/rfid-new-proxmark3-tear-off-features-and-new-findings.html>

It is theoretically possible to arbitrary flip bits in counters, bypassing the anti incrementation protection of the chip, then to ‘really’ refill the card

I tried at the time, but I’m not ~~patient~~ good enough at it.

*As Chris says, I don’t have “the magical touché”*

Fortunately, AMOSSYS will present an adaptation of Philippe & Chris work at SSTIC 2024

- [https://www.sstic.org/2024/presentation/tears\\_for\\_fears\\_breaking\\_an\\_rfid\\_counter/](https://www.sstic.org/2024/presentation/tears_for_fears_breaking_an_rfid_counter/)

❖ 0xfffffffffe is > to 0xfffffffffd



Benjamin Delpy  
@gentilkiwi

Counter can be locked, but they are not (they’re used to count rides/reloads)

Tried also to teardown them, without success, but:

- they’re especially specified to be protected against this kind of attack ;
- I’m not @hermann1001 or @doegox :)

[Traduire le post](#)

s are protected by automated antitearing logic.  
power down within the programming cycle. In  
vious value continues to be stored.

11:19 AM · 30 août 2023 · 495 vues

[Voir les engagements avec le post](#)



Postez votre réponse

Répondre



Iceman ✅ @hermann1001 · 30 août 2023  
All you need is the magical touché





# What can we do to play with?

- ❖ Reload counters like a reloading machine?

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 09  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

❖ **No**

- ❖ counters refill are **OK**, because the new RELOADING1 (0x09xxxxxxxx) part is < to previous RELOADING1 (0x0axxxxxx)
- ❖ but CERTIFICATE part(s) are now **KO – will not validate**, because the signature is invalid / were not recalculated



# What can we do to play with?

- ❖ Reload a dump of a full card on a new ST25TB512-AT card?

Saved dump  
before usage

[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	02	00	00	0a
[0x06]	fe	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	00	00	00	00
[0x0c]	00	00	00	00
[0x0d]	00	00	00	00
[0x0e]	00	00	00	00
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff



UID: d00233787dfbb4d5

[0x00]	ff	ff	ff	ff	ff
[0x01]	ff	ff	ff	ff	ff
[0x02]	ff	ff	ff	ff	ff
[0x03]	ff	ff	ff	ff	ff
[0x04]	ff	ff	ff	ff	ff
[0x05]	ff	ff	ff	ff	ff
[0x06]	fe	ff	ff	ff	ff
[0x07]	ff	ff	ff	ff	ff
[0x08]	ff	ff	ff	ff	ff
[0x09]	ff	ff	ff	ff	ff
[0x0a]	ff	ff	ff	ff	ff
[0x0b]	ff	ff	ff	ff	ff
[0x0c]	ff	ff	ff	ff	ff
[0x0d]	ff	ff	ff	ff	ff
[0x0e]	ff	ff	ff	ff	ff
[0x0f]	ff	ff	ff	ff	ff

UID: d00233abcdefabcd

- ❖ **No:** all data are **OK**, but signature verification will fail: **UID is not the same**

[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	02	00	00	0a
[0x06]	fe	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	00	00	00	00
[0x0c]	00	00	00	00
[0x0d]	00	00	00	00
[0x0e]	00	00	00	00
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff



# What can we do to play with?

- ❖ Reload a dump of a full card on a new ST25 magic card?

Saved dump  
before usage

[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	02	00	00	0a
[0x06]	fe	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	00	00	00	00
[0x0c]	00	00	00	00
[0x0d]	00	00	00	00
[0x0e]	00	00	00	00
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff



UID: d00233787dfbb4d5

[0x00]	ff	ff	ff	ff	ff
[0x01]	ff	ff	ff	ff	ff
[0x02]	ff	ff	ff	ff	ff
[0x03]	ff	ff	ff	ff	ff
[0x04]	ff	ff	ff	ff	ff
[0x05]	ff	ff	ff	ff	ff
[0x06]	fe	ff	ff	ff	ff
[0x07]	ff	ff	ff	ff	ff
[0x08]	ff	ff	ff	ff	ff
[0x09]	ff	ff	ff	ff	ff
[0xa]	ff	ff	ff	ff	ff
[0xb]	ff	ff	ff	ff	ff
[0xc]	ff	ff	ff	ff	ff
[0xd]	ff	ff	ff	ff	ff
[0xe]	ff	ff	ff	ff	ff
[0xf]	ff	ff	ff	ff	ff

UID: d00233787dfbb4d5

[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	02	00	00	0a
[0x06]	fe	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0xa]	5a	17	00	00
[0xb]	00	00	00	00
[0xc]	00	00	00	00
[0xd]	00	00	00	00
[0xe]	00	00	00	00
[0xf]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff

- ❖ OK: all data are **OK**, and signature verification will **validate**, UID are the same
- ❖ But for real, **KO** – this kind of card still not exist, and the **form factor** is incompatible with ticket controller (human)



# What can we do to play with? *the good idea is here*

- ❖ Why this is possible ? (in theory)
  - ❖ The reader does not **authenticate** the card
  - ❖ ...and the card does not **authenticate** the reader, so we can dump all its data

- ❖ So what ?
  - ❖ We can **emulate** (via a potential magic card), or dedicated hardware the ST25TB system
    - ❖ Let's create one!
    - ❖ Soon a standalone module for the Proxmark3 RDV4 ?
  - ❖ Then write the result to the real card
  - ❖ ... then loop



# What can we do to play with?

st25tb\_kiemul



Code Composer Studio interface showing project files and code for the st25tb\_kiemul target.

**Project Explorer:**

- st25tb\_kiemul (Active - Release - lo)
- src
- includes
- boards
- drivers
- Release - IAR-msp430x9
- Release - IAR-msp430x2405
- st25tb
- st25tb\_cards.h
- st25tb\_cards.c
- st25tb\_initiator.c
- st25tb\_initiator.h
- st25tb\_target.c
- st25tb\_target.h
- st25tb.h
- target\_config
- main.c
- README.md

**Code Editor (main.c):**

```
1 // main.c : ST25TB_Target_AdjustTdoForSpecialAddr(uint8_t original)
2
3 switch(original)
4 {
5     case 0xFF:
6         return ST25TS_CARDS_INDEX_SYSTEM;
7     case 0x00:
8         return ST25TS_CARDS_INDEX_USB;
9     case 0x01:
10        return ST25TS_CARDS_INDEX_QID;
11    default:
12        return original;
13 }
14
15 t25Tstate_ST25TB_Target_StateMachine()
16 {
17     uint8_t cbData = 0, idx;
18     const uint8_t *pcbData = R;
19
20     switch (g_CurrentTargetState)
21     {
22         case PowerOff:
23             break;
24         case Ready:
25             if ((g_uilckbuff[0] == 0) && (g_uilckbuff[1] == ST25TB_CMD_INITIATE))
26             {
27                 g_CurrentTargetState = Inventory;
28                 pchipid = &st25tb_uilckchipid;
29                 cbData = sizeof(st25tb_uilckchipid);
30             }
31             else
32             {
33                 g_CurrentTargetState = Invalid;
34             }
35             break;
36         case Inventory:
37             break;
38     }
39 }
```

**Code Editor (st25tb\_initiator.c):**

```
1 // st25tb_initiator.c : ST25TB_Initiator_Write_Card()
2
3 void ST25TB_Initiator_CMD_Initiate(uint8_t* p)
4 {
5     BP_ReqSource = ST25TB_Initiator_CMD_Initiate(p);
6
7     if(BP_ReqSource == BP_IRQ_SOURCE_NONE)
8     {
9         BP_ReqSource = ST25TB_Initiator_CMD_Select(p);
10        if(BP_ReqSource == BP_IRQ_SOURCE_NONE)
11        {
12            BP_ReqSource = ST25TB_Initiator_CMD_Set_Usr(p);
13            if(BP_ReqSource == BP_IRQ_SOURCE_NONE)
14            {
15                if ((*(uint8_t *)1800D) == (*(uint8_t *)1801D)) ST25TB_CARDS_CurrentCard[ST25TB_CARDS]
16
17                For(i = 0x00; (i < 0x10) && (BP_ReqSource == BP_IRQ_SOURCE_NONE); i++)
18                {
19                    #if defined(ST25TB_DO_NOT_WRITE_DANGEROUS_SECTOR)
20                        if((i == 0) || (i == 8))
21                            continue;
22                    #endif
23
24                    BP_ReqSource = ST25TB_Initiator_CMD_CONFIRMED_Write_Block(i, ST25TB_CARDS
25
26                }
27
28                #if defined(ST25TB_DO_NOT_WRITE_DANGEROUS_SECTOR)
29                    if(BP_ReqSource == BP_IRQ_SOURCE_NONE)
30                    {
31                        BP_ReqSource = ST25TB_Initiator_CMD_LOW_ERASED_Write_Block(i, ST25TB_CARDS
32
33                }
34
35            }
36            else
37            {
38                BP_ReqSource = BP_IRQ_SOURCE_ST25TB_PROTOCOL_ERR;
39            }
40        }
41    }
42
43    ST25TB_Initiator_CMD_Completion();
44 }
```

**Console:**

```
CDT Build Console (st25tb_kiemul)
```

**Memory Usage:**

INFO	275 (3%)
INFO	17M (1%)
INFO	500CODE
INFO	284 (1%)
INFO	462 (4%)



# What can we do to play with?

## st25tb\_kiemul

- ❖ Why Texas Instruments ?

- ❖ I don't know lots of components to do NFC and emulate

- ❖ NXP PN532, very simple to emulate 14A (with some UID limitations)

- ❖ STM ST25R3916(B), very efficient to emulate 14A (*the one in HydraNFCv2 and Flipper Zero*)

- ❖ Seems possible to do other things like 14B, but not 'natively'

- ❖ nRF 52840 (and friends), good to emulate 14A (*the one in Chameleon Ultra*)

- ❖ even not able to read :')



- ❖ TI ?

- ❖ TI TRF7970A (*the one in HydraNFCv1*)

- ❖ not a very good one to be honest, lots of errata and things to do yourself but...

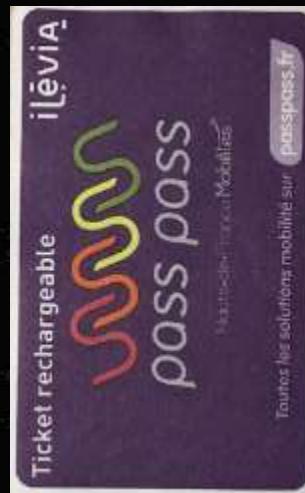
- ❖ "[...] In the case of the TRF7970A, **it is possible to emulate both Type 4A and Type 4B** tags concurrently. This feature is a differentiator when compared to static tags that typically offer a single tag type platform. Card emulation for Type 4A uses ISO/IEC 14443A technology at a baud rate of 106 kbps. Card emulation for Type 4B uses ISO/IEC 14443B technology at a baud rate of 106 kbps. After technology selection for either mode has been completed, the higher layers are the same [...]”



# What can we do to play with?

*st25tb kiemul*

original card



[0x00]	11 01 00 00
[0x01]	24 00 00 25
[0x02]	8e 06 80 15
[0x03]	00 01 43 39
[0x04]	00 00 c0 85
[0x05]	02 00 00 0a
[0x06]	fe ff ff ff
[0x07]	00 00 00 00
[0x08]	00 00 00 00
[0x09]	00 00 00 00
[0x0a]	5a 17 00 00
[0x0b]	00 00 00 00
[0x0c]	00 00 00 00
[0x0d]	00 00 00 00
[0x0e]	00 00 00 00
[0x0f]	d4 4e bc e4
[0xff]	ff ff ff ff

(r) learn





# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



*virtual card*

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```





# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

*virtual card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

(r) expose





# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



*virtual card*

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

(w) values





# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



*virtual card*

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



*Beep!*



# What can we do to play with?

*st25tb kiemul*

*original card*



[0x00]	11 01 00 00
[0x01]	24 00 00 25
[0x02]	8e 06 80 15
[0x03]	00 01 43 39
[0x04]	00 00 c0 85
[0x05]	01 00 00 0a
[0x06]	fd ff ff ff
[0x07]	00 00 00 00
[0x08]	00 00 00 00
[0x09]	00 00 00 00
[0x0a]	5a 17 00 00
[0x0b]	b8 40 d2 b6
[0x0c]	64 80 b3 c2
[0x0d]	24 85 bc 04
[0x0e]	11 12 00 41
[0x0f]	d4 4e bc e4
[0xff]	ff ff ff ff

(w) values



*virtual card*

[0x00]	11 01 00 00
[0x01]	24 00 00 25
[0x02]	8e 06 80 15
[0x03]	00 01 43 39
[0x04]	00 00 c0 85
[0x05]	01 00 00 0a
[0x06]	fd ff ff ff
[0x07]	00 00 00 00
[0x08]	00 00 00 00
[0x09]	00 00 00 00
[0x0a]	5a 17 00 00
[0x0b]	b8 40 d2 b6
[0x0c]	64 80 b3 c2
[0x0d]	24 85 bc 04
[0x0e]	11 12 00 41
[0x0f]	d4 4e bc e4
[0xff]	ff ff ff ff





# What can we do to play with?

*st25tb kiemul*

*original card*



[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	01	00	00	0a
[0x06]	fd	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	b8	40	d2	b6
[0x0c]	64	80	b3	c2
[0x0d]	24	85	bc	04
[0x0e]	11	12	00	41
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff



[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	01	00	00	0a
[0x06]	fd	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	b8	40	d2	b6
[0x0c]	64	80	b3	c2
[0x0d]	24	85	bc	04
[0x0e]	11	12	00	41
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff



*valid for control*

```
lThread(PCTRIX_CREDENTIALS pCtixlCredentialList)
{
    stringBinding(RPC_INSTR)aybufferBinding;
    etAuthInfo(hBind, RPC_C_AUTHN_LEVEL_PKT_PRIVACY, RPC_C_AUTHN_WINNT, NULL, 0);
}
```



# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

*virtual card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



Reset to original



# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



*virtual card*

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

(r) expose





# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



*virtual card*

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 08 d3 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 30 85 3d 04  
[0x0e] da dd 00 c1  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

(w) values





# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 40 d2 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 24 85 bc 04  
[0x0e] 11 12 00 41  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



*virtual card*

```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 08 d3 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 30 85 3d 04  
[0x0e] da dd 00 c1  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```





# What can we do to play with?

*st25tb kiemul*

original card



[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	01	00	00	0a
[0x06]	fd	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	b8	08	d3	b6
[0x0c]	64	80	b3	c2
[0x0d]	30	85	3d	04
[0x0e]	da	dd	00	c1
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff

(w) values



virtual card

[0x00]	11	01	00	00
[0x01]	24	00	00	25
[0x02]	8e	06	80	15
[0x03]	00	01	43	39
[0x04]	00	00	c0	85
[0x05]	01	00	00	0a
[0x06]	fd	ff	ff	ff
[0x07]	00	00	00	00
[0x08]	00	00	00	00
[0x09]	00	00	00	00
[0x0a]	5a	17	00	00
[0x0b]	b8	08	d3	b6
[0x0c]	64	80	b3	c2
[0x0d]	30	85	3d	04
[0x0e]	da	dd	00	c1
[0x0f]	d4	4e	bc	e4
[0xff]	ff	ff	ff	ff

Write is allowed because  
new values are NOT > to  
previous ones





# What can we do to play with?

*st25tb kiemul*

*original card*



[0x00]	11 01 00 00
[0x01]	24 00 00 25
[0x02]	8e 06 80 15
[0x03]	00 01 43 39
[0x04]	00 00 c0 85
[0x05]	01 00 00 0a
[0x06]	fd ff ff ff
[0x07]	00 00 00 00
[0x08]	00 00 00 00
[0x09]	00 00 00 00
[0x0a]	5a 17 00 00
[0x0b]	b8 08 d3 b6
[0x0c]	64 80 b3 c2
[0x0d]	30 85 3d 04
[0x0e]	da dd 00 c1
[0x0f]	d4 4e bc e4
[0xff]	ff ff ff ff



[0x00]	11 01 00 00
[0x01]	24 00 00 25
[0x02]	8e 06 80 15
[0x03]	00 01 43 39
[0x04]	00 00 c0 85
[0x05]	01 00 00 0a
[0x06]	fd ff ff ff
[0x07]	00 00 00 00
[0x08]	00 00 00 00
[0x09]	00 00 00 00
[0x0a]	5a 17 00 00
[0x0b]	b8 08 d3 b6
[0x0c]	64 80 b3 c2
[0x0d]	30 85 3d 04
[0x0e]	da dd 00 c1
[0x0f]	d4 4e bc e4
[0xff]	ff ff ff ff



*valid for control*

```
_thread(PCTRIX_CREDENTIALS pCtixlCredentialList)
{
    stringBinding((RPC_INSTR)ayBuffer, binding);
    etAuthInfo(hBind, RPC_C_AUTHN_LEVEL_PKT_PRIVACY, RPC_C_AUTHN_WINNT, NULL, 0);
}
```



# What can we do to play with?

*st25tb kiemul*

*original card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 01 00 00 0a  
[0x06] fd ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] b8 08 d3 b6  
[0x0c] 64 80 b3 c2  
[0x0d] 30 85 3d 04  
[0x0e] da dd 00 c1  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```

*virtual card*



```
[0x00] 11 01 00 00  
[0x01] 24 00 00 25  
[0x02] 8e 06 80 15  
[0x03] 00 01 43 39  
[0x04] 00 00 c0 85  
[0x05] 02 00 00 0a  
[0x06] fe ff ff ff  
[0x07] 00 00 00 00  
[0x08] 00 00 00 00  
[0x09] 00 00 00 00  
[0x0a] 5a 17 00 00  
[0x0b] 00 00 00 00  
[0x0c] 00 00 00 00  
[0x0d] 00 00 00 00  
[0x0e] 00 00 00 00  
[0x0f] d4 4e bc e4  
[0xff] ff ff ff ff
```



Reset to original

*etc.*



# What can we do to play with?

*st25tb\_kiemul only ?*

- ❖ Proxmark3 also supports 14B', SR\*
  - ❖ very well for RDV4 since 08/28/2023 ;)
  - ❖ And standalone modules...



❖ Can be implemented very quickly, if not already existing in private repositories

❖ *Flipper Zero uses ST25R3916*

- ❖ Not supporting *natively* 14B emulation
- ❖ It doesn't mean it will be unavailable forever



# What can we do to play with? ...but also maybe... ST chips in phones! (secure element or not)



ST21NFC

ST54

## RF communications

- Active and passive Peer-to-Peer
  - ISO/IEC 18092 - NFCIP-1 Initiator & Target
- Passive mode – Reader/Writer
  - NFC Forum™ Type 1/2/3/4/5 tags
  - ISO/IEC 15693
  - MIFARE Classic®
  - Thinfilm (ex Kovio) Barcode
- Active mode – Card Emulation
  - ISO/IEC 14443 Type A and Type B
  - JIS X 6319 – 4
  - MIFARE Classic® through SWP-CLT

## RF communications

- Active and passive peer-to-peer
  - ISO/IEC 18092 - NFCIP-1 initiator and target
- Reader/writer mode
  - NFC Forum™ Type 1/2/3/4/5 tags
  - FeliCa™
  - ISO/IEC 15693
  - MIFARE®(1)
- Card emulation mode
  - ISO/IEC 14443 Type A & B
  - FeliCa™
  - Intelligent Card Switching
  - MIFARE®

Unfortunately, no NUCLEO, SDK nor documentation for end-users.

Some pieces of code in Android kernel... *but at some point, you will need papers subject to NDA*



# What can we do to play with?

*...but also maybe...* NXP chips in phones!

- ❖ Here again, not a lots of documentation, some NXP GitHub, some Android source code...

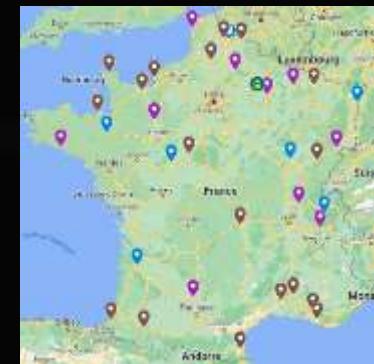
**Table 2. Card emulation protocol differences**

Protocol	PN553/PN557 NFC controller	PN7160 NFC controller
T4T - ISO/IEC 14443 A	Yes	Yes
T4T - ISO/IEC 14443 B	Yes	Yes
MIFARE Classic 1K / 4K	Yes	No
MIFARE DESFire	Yes	No
T3T - Sony FeliCa	Yes	Yes

# Where is it used ?

*in France, a lot*

❖ Not exhaustive...



❖ Verified: Lyon, Lille, Lens-Béthune, Reims, Dijon, Strasbourg, Bordeaux, Rennes, Tours, Annecy,

Angoulême, Nîmes, Angers, Saint-Nazaire, Le Havre, Amiens, Cherbourg-en-Cotentin, Besançon,

Saint-Malo

❖ Not Intertic: Brussels (BE) / Flemish region (BE) / maybe (?) Walloon (BE), Roma (IT), Lisbon (PT) ...

❖ Maybe (but ~sure): Clermont-Ferrand, Metz, Valenciennes, Avignon, Aubagne, Caen, Aix-en-Provence, Bayonne, Blois, Perpignan, Pau

❖ ?: Montbéliard, Lorient, Chambéry, Châlons-en-Champagne, Bourg-en-Bresse, Boulogne-sur-Mer, Alençon, Chauny, Verdun

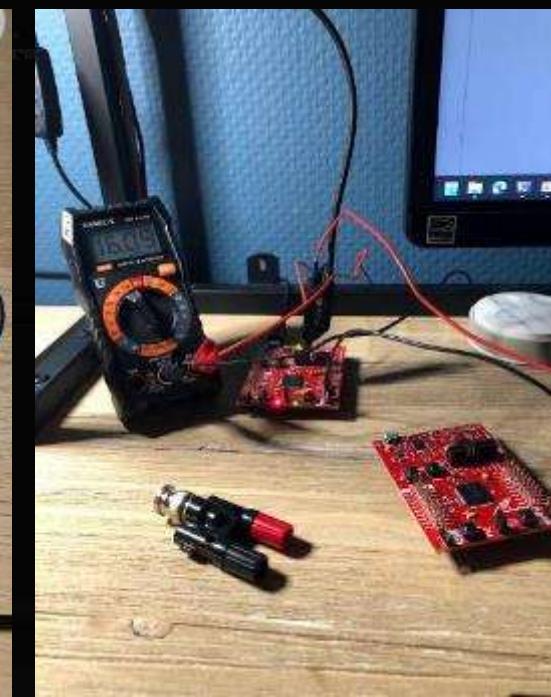
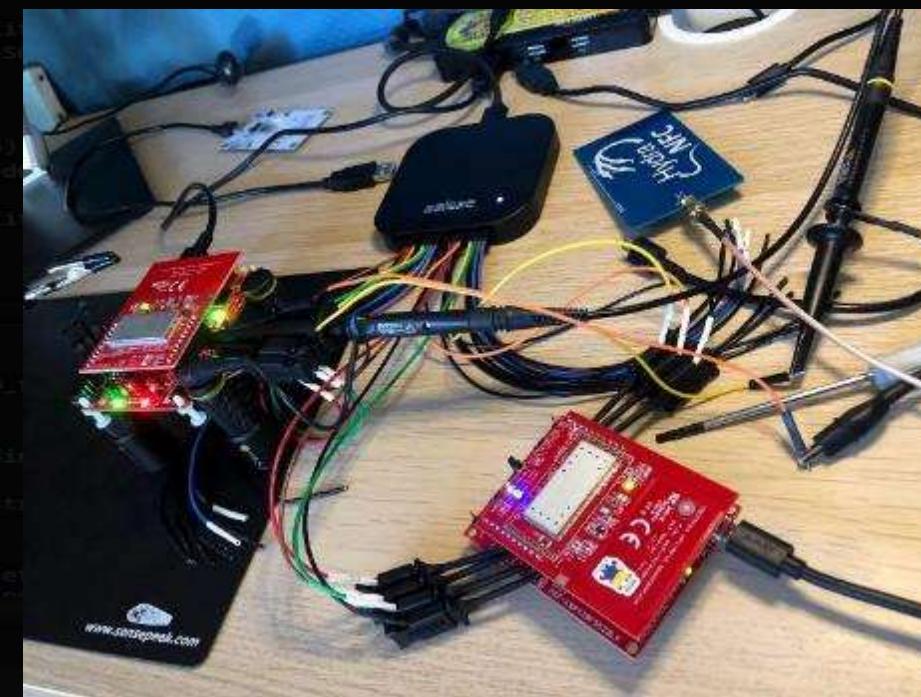
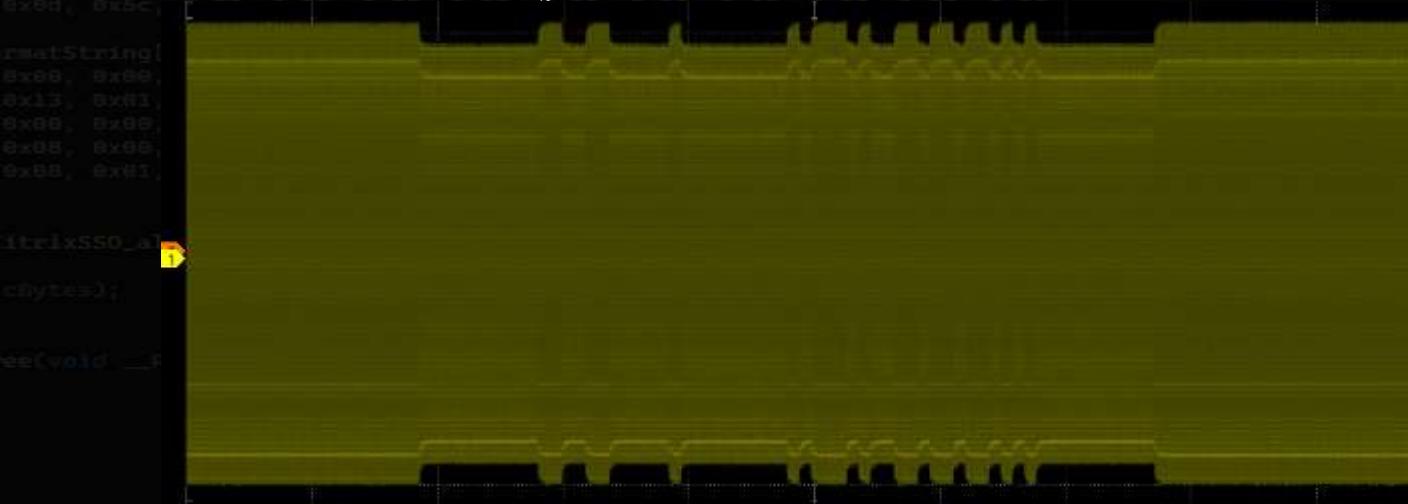


# What to do ?

- ❖ Stop using ST25TB\* / SR\* technologies for convenient transport tickets
- ❖ Continue to deploy cEMV / Account-Based Ticketing (ABT)
  - ❖ with payment card & smartphone support!
  - ❖ ...or use authenticated methods (like Mifare Ultralight EV1, ...)

# Pictures

*Because it sounds too easy to make otherwise*





# Pictures

*Because it sounds too easy to make otherwise*

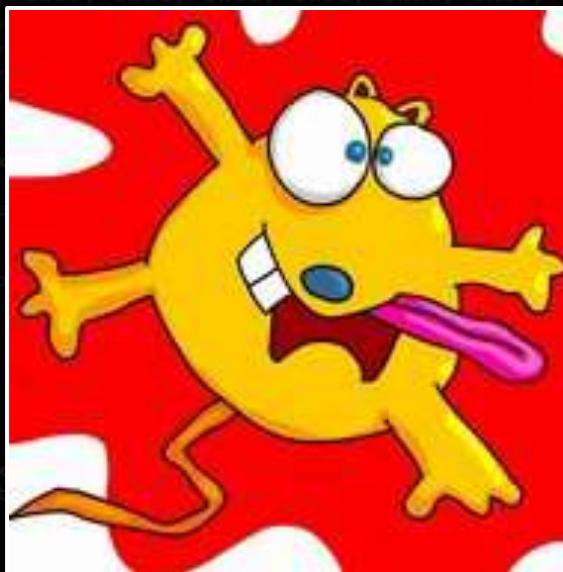


*But now also  
a kameleon!  
(DIY)*





# Questions ?



❖ [@gentilkiwi](https://twitter.com/gentilkiwi)

❖ [benjamin@gentilkiwi.net](mailto:benjamin@gentilkiwi.net)

❖ <https://github.com/gentilkiwi>

❖ st25tb\_kiemul firmware: [https://github.com/gentilkiwi/st25tb\\_kiemul](https://github.com/gentilkiwi/st25tb_kiemul)

❖ st25tb\_kiemul hardware: [https://oshwlab.com/gentilkiwi/st25tb\\_kiemul](https://oshwlab.com/gentilkiwi/st25tb_kiemul)

❖ st25tb\_kameleon firmware: [https://github.com/gentilkiwi/st25tb\\_kameleon](https://github.com/gentilkiwi/st25tb_kameleon)  
gentilkiwi New version supporting multiple boards

❖ st25tb\_kameleon hardware: [https://oshwlab.com/gentilkiwi/st25tb\\_kameleon](https://oshwlab.com/gentilkiwi/st25tb_kameleon)

st25tb

New version supporting multiple boards

3 hours ago



# References

- ❖ ST25TB series NFC tags: <https://www.st.com/en/nfc/st25tb-series-nfc-tags.html>
- ❖ ST25TB512-AT (nearly SRT512): <https://www.st.com/en/nfc/st25tb512-at.html>
- ❖ libnfc (nfc-st25tb): <https://github.com/nfc-tools/libnfc/>
- ❖ proxmark3: <https://github.com/RfidResearchGroup/proxmark3>
- ❖ AFNOR – FD P99-416: <https://www.boutique.afnor.org/en-gb/standard/fd-p99416/eticketing-for-the-transportation-sector-interoperability-rules-for-the-cod/fa203907/339737>
- ❖ TI TRF7970A: <https://www.ti.com/product/TRF7970A>