NFC/RFID Security Hands-on RMLL 2013

Philippe Teuwen
SECurity REsearch Team – Leuven
NXP Semiconductors
09/07/2013

Agenda

- Standards
- Readers
- Tools
- Security aspects
- Hands-on
- Demos

Highly non-linear...

NFC/RFID LiveCD

GNU/Linux
- Debian Wheezy

Hybrid: ISO & dd of=/dev/sdx

"upgradable" via apt-get

Can also run under VirtualBox

RFID-related software, drivers & docs (cf readme.txt)

Not just for one-day experience!

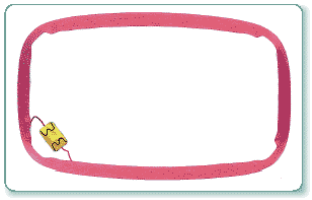
http://live.debian.net (3.x)
http://nfc-live.googlecode.com

RFID Zoo

Frequency	Standards	Applications
LF (125–134 kHz)	ISO 11784/85 ISO 18000-2	Animal ID, Car immobilizer
HF (13.56 MHz)	ISO 14443	AFC, banking, eGov
	ISO 15693 ISO 18000-3 HF EPC Gen2	Supply chain track & trace, Item level tagging
UHF (840 – 960 MHz)	ISO 18000-6 UHF EPC Gen2	Supply chain track & trace

Incomplete picture:

- More frequency bands
- Many more standards



RFID hacking on a PC

Commercially available readers:

- No standard reader API
- Same as pre-PC/SC era
- Some hooks on PC/SC

Let's focus on readers & tools with open-source support

ACG LF aka OMNIKEY 5534

125 & 134.2 kHz

EM4x02

EM4x50

EM4x05
(ISO 11784/5 FDX-B)

Hitag 1 / 2 / S

Q5

TI 64 bit R/O & R/W

TI 1088 bit Multipage

Module available at http://www.rfidiot.org

ACG LF aka OMNIKEY 5534

- `readlfx.py -R READER_ACG -s <baudrate>`
- `rfdump` (File / Prefs / ACG / baudrate then Reader / Start scan)
- `screen /dev/ttyUSB0 <baudrate>`

```
!     test continuous read -> ! if active, F if not
c     continuous read      -> poll, any key to stop -> S
dX    set tag settings     -> dH80 gain=2 sampling_time=0
l     login                -> lMIKR -> L=ok X=fail N=no_tag
oX    set tag type         -> oH
o+X   include tag type
o-X   exclude tag type
poff  antenna power off
pon   antenna power on
rb    read block           -> rb00  -> 4 bytes
wb    write block          -> wb0011223344
rp    read EEPROM
wp    write EEPROM
s     select               -> poll once
v     get version
x     reset
y     field reset          -> y8080 off time (ms) + recovery time (ms)
```

Omnikey CardMan 5321

Based on CL RC632

ISO 14443

ISO 15693

Also contact interface

Linux: PC/SC vendor driver
(libccid only supports contact interface)

Danger: be careful with dual-interface cards!

PC/SC

Personal Computer / SmartCard

1996

Goal: interoperability through common API

http://en.wikipedia.org/wiki/PC/SC
http://www.pcscworkgroup.com

PC/SC

Offers reader vendor independent API
→ reader independent applications

Controls shared access

Supports transaction management primitives

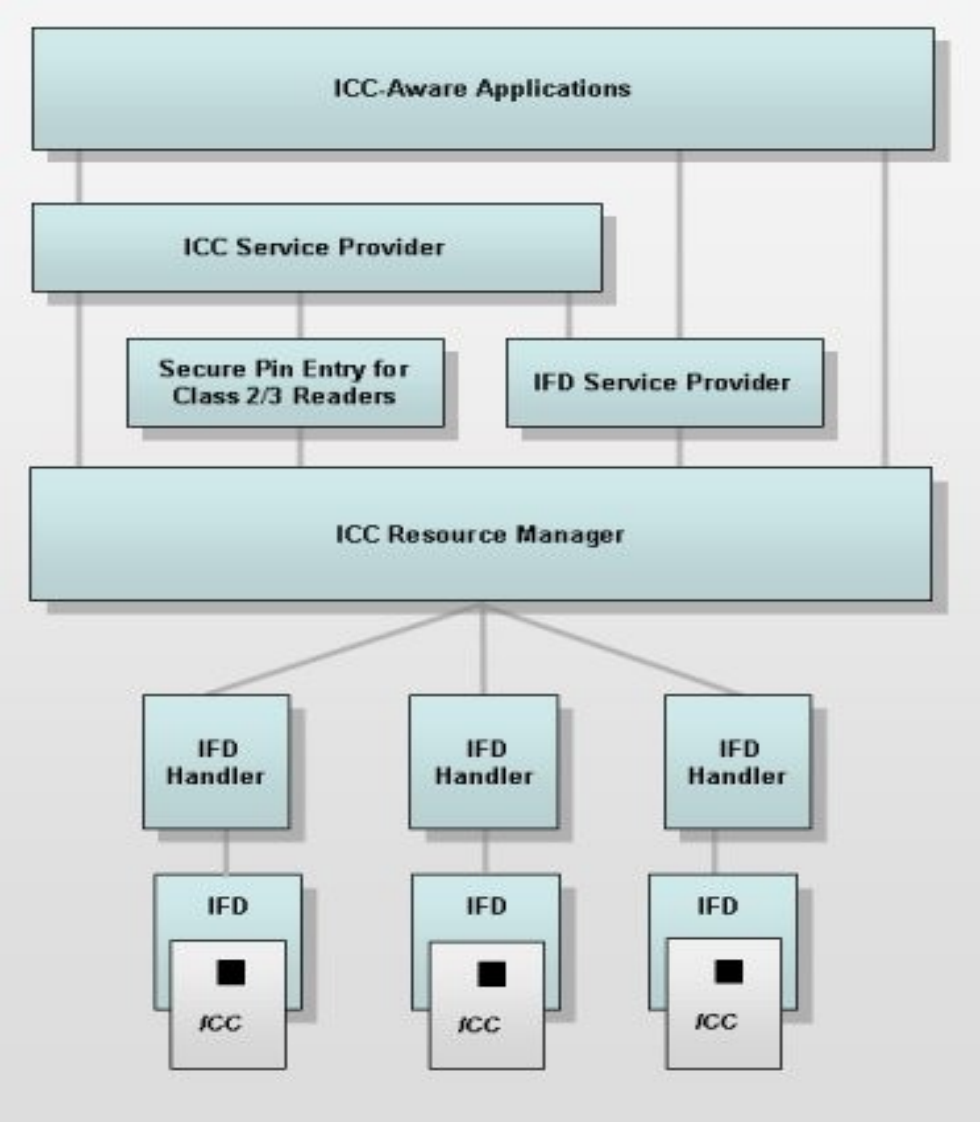
OS provides PC/SC service

Vendor provides PC/SC driver

PC/SC

IFD Handler
= device driver
- RS232
- PS/2 (kbd)
- PCMCIA
- USB
- USB-CCID
- …

Linux & Mac OS X:
pcsc-tools



IFD Handler USB-CCID

For Chip/SmartCard Interface Devices

One common driver

- USB device <> PC/SC
- Microsoft: usbccid.sys
- Linux & Mac OS X: libccid

PC/SC API

SCardEstablishContext(...)
- First called, to talk to PC/SC service

SCardListReaders(...)

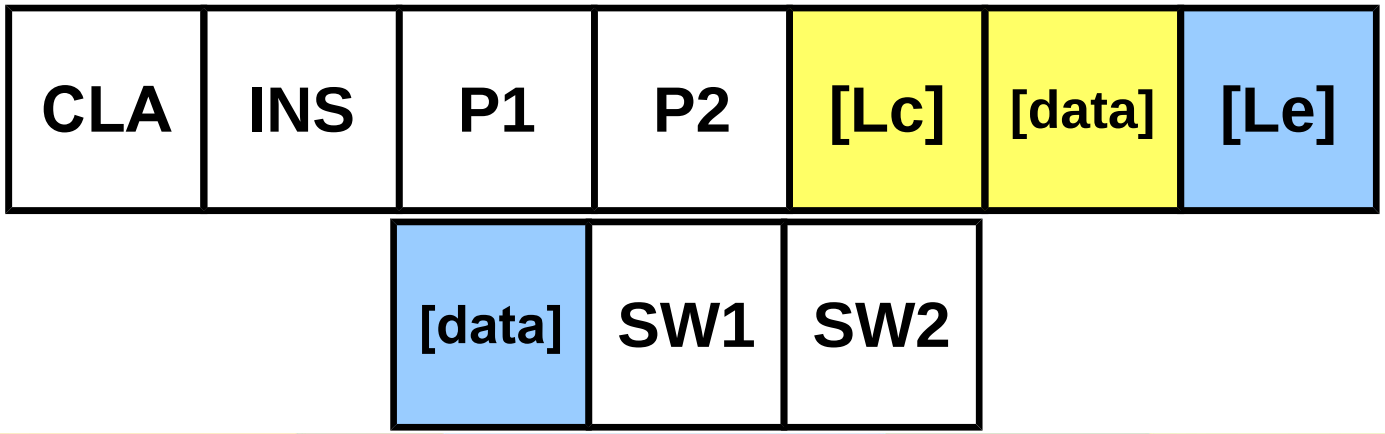
SCardConnect(reader, shared mode, proto,...)
- Card power-up & reset, ATR

ScardTransmit(APDU, pbRecvBuffer,...)

SmartCard → ATR & APDU

- Answer To Reset (ISO 7816-3)
  http://en.wikipedia.org/wiki/Answer_to_reset
  http://ludovic.rousseau.free.fr/softwares/pcsc-tools/smartcard_list.txt

- Application Protocol Data Unit (ISO 7816-4)
  http://en.wikipedia.org/wiki/APDU



Contactless → ATR??

Reader generates a PC/SC compliant ATR according to PC/SC v2.01, Part 3, 3.1.3.2: “Contactless Protocol Support”

- Smartcards (ISO14443-4)
  - ATS to ATR mapping, cf table 3.5 in 3.1.3.2.3.1
- Storage cards
  - cf table 3.6 in 3.1.3.2.3.1. Standard and card name mapped according to Part 3 Supplemental Document of PCSC 2.01

pcsc-tools

- Ludovic Rousseau
- `pcsc_scan`
- `scriptor` / `gscriptor`

  http://ludovic.rousseau.free.fr/softwares

Your reader: SCL3711

3 ways to use it...
helper scripts on the ISO are available

- With its proprietary PCSC driver

  `scl3711-pcsc_proprio`

- With the new ifdnfc
  opensource PCSC driver (still beta)

  `scl3711-pcsc_ifdnfc`

  `ifdnfc-activate`

- Without driver, via libnfc

  `scl3711-libnfc`

You might have to re-plug the reader if unresponsive

Example: ATR for MIFare 1K = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A}

ATR											
Initial Header	T0	TD1	TD2	T1	Tk	Length	RID	Standard	Card Name	RFU	TCK
3B	8F	80	01	80	4F	0C	A0 00 00 03 06	03	00 01	00 00 00 00	6A

Where:

Length (YY) = 0C

RID = A0 00 00 03 06 (PC/SC Workgroup)

Standard (SS) = 03 (ISO14443A, Part 3)

Card Name (C0 .. C1) = [00 01] (MIFare 1K)

Where, Card Name (C0 .. C1)

- 00 01: Mifare 1K
- 00 02: Mifare 4K
- 00 03: Mifare Ultralight
- 00 26: MiFare Mini
- ....
- F0 04: Topaz and Jewel
- F0 11: FeliCa 212K
- F0 12: Felica 424K
- ...

pcsc_scan

Contactless → APDU??

Smartcards

- ISO7816-4 APDU support, so just pass-thru
- GetData UID: FF CA 00 00 Le

  `scriptor` → `ffca000000`
- GetData ATS: FF CA 01 00 Le

Storage cards

- Transform APDU into specific command(s)
  = filter/map requests & responses
- GetUID, Read Binary, Update Binary, Load Keys, General Authenticate, (Verify)
- Other vendor-specific mappings

PC/SC 2.0 Part 3 sup 2

- New extension covering all you dreamed about for contactless readers
  - Raw modes, modulations, etc
- So it will be possible to write contactless-oriented applications agnostic to the type of reader you have
- NOT covering NFC

cf nfc-doc/technology/PCSC/pcsc3_v2.02.00_sup2.pdf

First(?) compliant reader chip: NXP PR533

PN53x family

NFC (ISO18092 NFCIP-1)

ISO14443-A Tag Read/Write

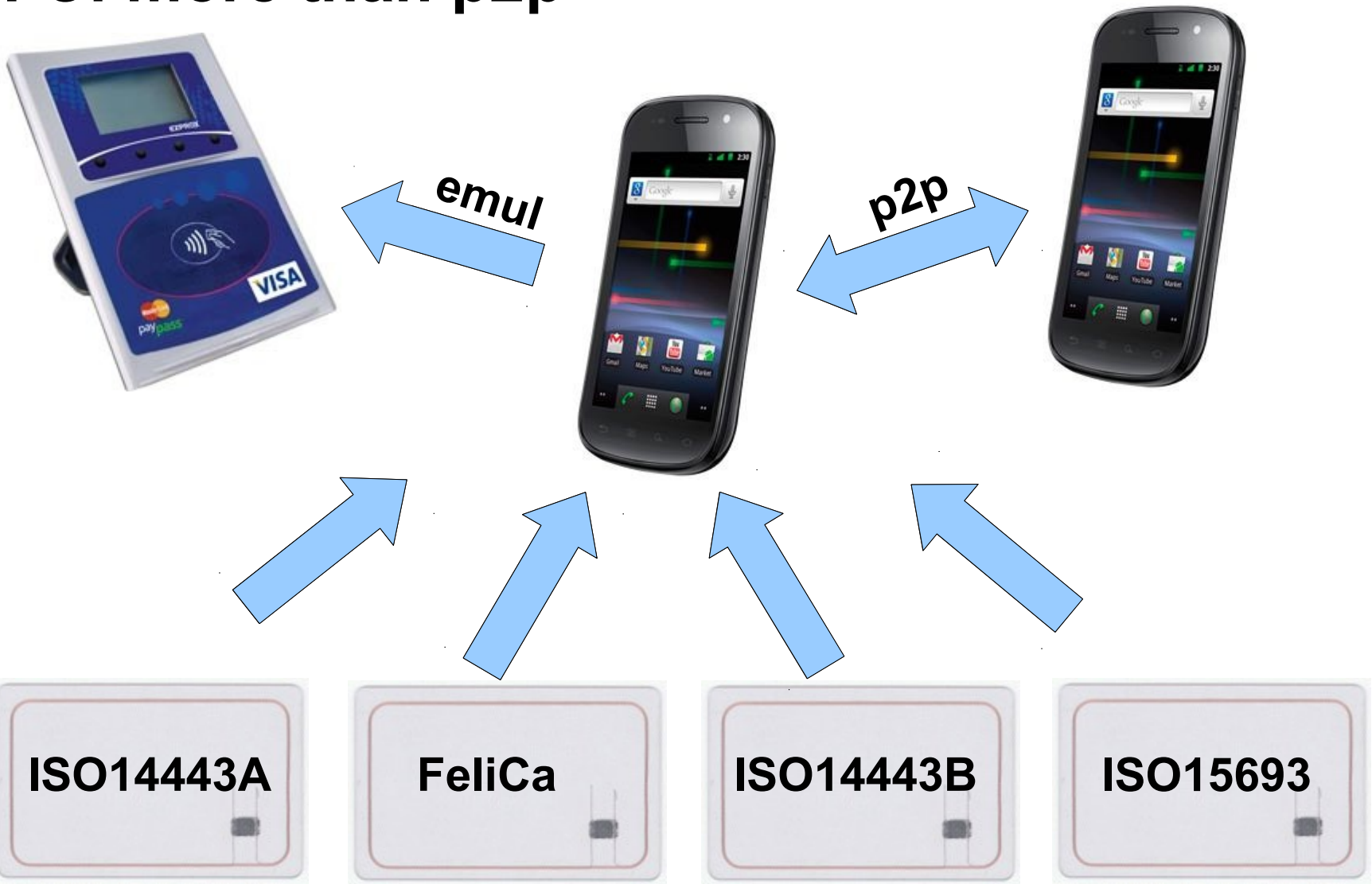
ISO14443-B Tag Read/Write

ISO14443-3A (Mifare®) Tag Emulate

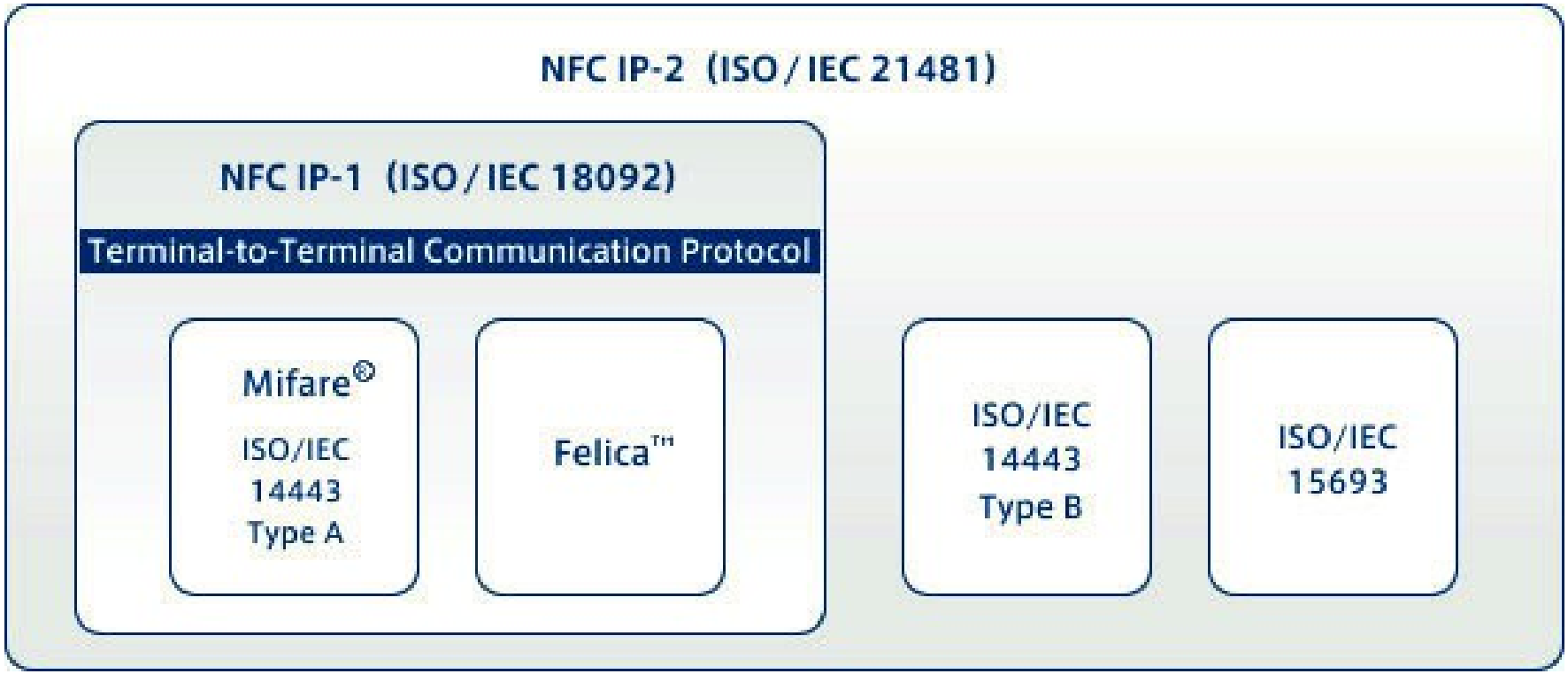
FeliCa™ Tag Read/Write/Emulate

- **PN532** (SPI / I2C / UART)
  - Automatic Polling Sequence
  - ISO14443-4A (T=CL) Tag Emulate
- **PN533** (USB 2.0)
  - NFC-SEC, PayPass

NFC: more than p2p



NFC: NFCIP + NFC-Forum



NFC-Forum

NDEF (NFC Data Exchange Format)
LLCP (Logical Link Control Protocol)
SNEP (Simple NDEF Exchange Protocol)
Android NPP (NDEF Push Protocol)

NFC-Forum Tags:

- Type 1: Innovision Topaz/Jewel (ISO14443-3A)
- Type 2: NXP Mifare Ultralight (ISO14443-3A)
- Type 3: Sony FeliCa
- Type 4: ISO7816-4 on ISO14443-4 A or B

PN53x family

TAMA language

D4	**CC**	**[data]**
D5	**CC+1**	**[data]**

- Ex: GetFirmware()
  D4 02
- **PN531** response
  D5 03 04 02
- **PN532** response
  D5 03 32 01 06 07
- **PN533** response
  D5 03 33 02 07 07

cf nfc-doc/products/NXP/{PN532|PN533}/

PN533-based: SCL3711 & ASK LoGO

- SCL3711
  - PCSC driver proprio
  - PCSC driver opensource
  - Direct libnfc support

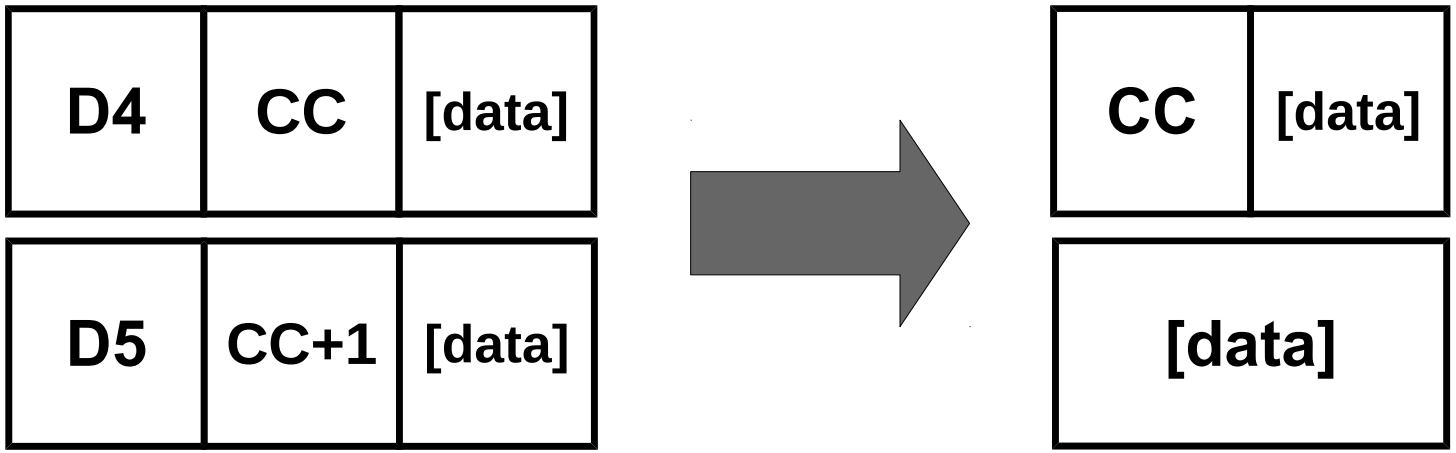
- ASK LoGO
  - Supports ISO14443-B' (*)
  - Progressive field for ISO14443-B
  - PCSC driver opensource
  - Direct libnfc support

(*) now in all libnfc supported readers

SCL3711 through libusb + libnfc

scl3711-libnfc

pn53x-tamashell



GetFirmwareVersion

02

33 02 07 07

IC=33 (PN533)

Ver=02

Rev=07

Support=07 (ISO18092+ISO14443B+ISO14443A)

libnfc

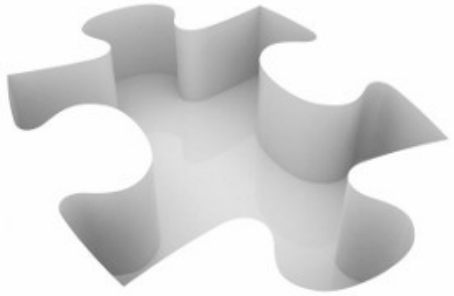
- Initiated by Roel Verdult
  Now mainly Romuald Conty, I & +10 developers
- Library to support PN53x readers + tools & examples
- via libusb, PC/SC, UART, SPI, I2C
- http://www.libnfc.org

```
nfc-<TAB><TAB>
```

libnfc-related projects

- libfreefare (MIFARE Classic, DESFire, UltralightC,...)
- ifdnfc
- nfc-tools:
  lsnfc, libnfc-llcp, pam_nfc, NfcEventD, DeskNFC,...
- qnfcd, pynfc, nfosc, libfm1208, micmd, mtools
- RFIDIOt
- mfoc, mfcuk, readnfccc
- mfocuino, nfcdoorlock

ifdnfc: bringing the missing piece



IFD Handler based on libnfc

Goal: make libnfc-supported devices PC/SC part 3 sup 2 compliant

Current status:

- PC/SC support (ATR & APDU) for ISO14443A-4
- FFCA000000 & FFCA010000
- Supports UART & USB devices
- Handles transparently USB libnfc devices,
  in the same way libccid supports USB CCID devices
- Handles multiple devices at once
- Can replace SCL3711 proprietary driver

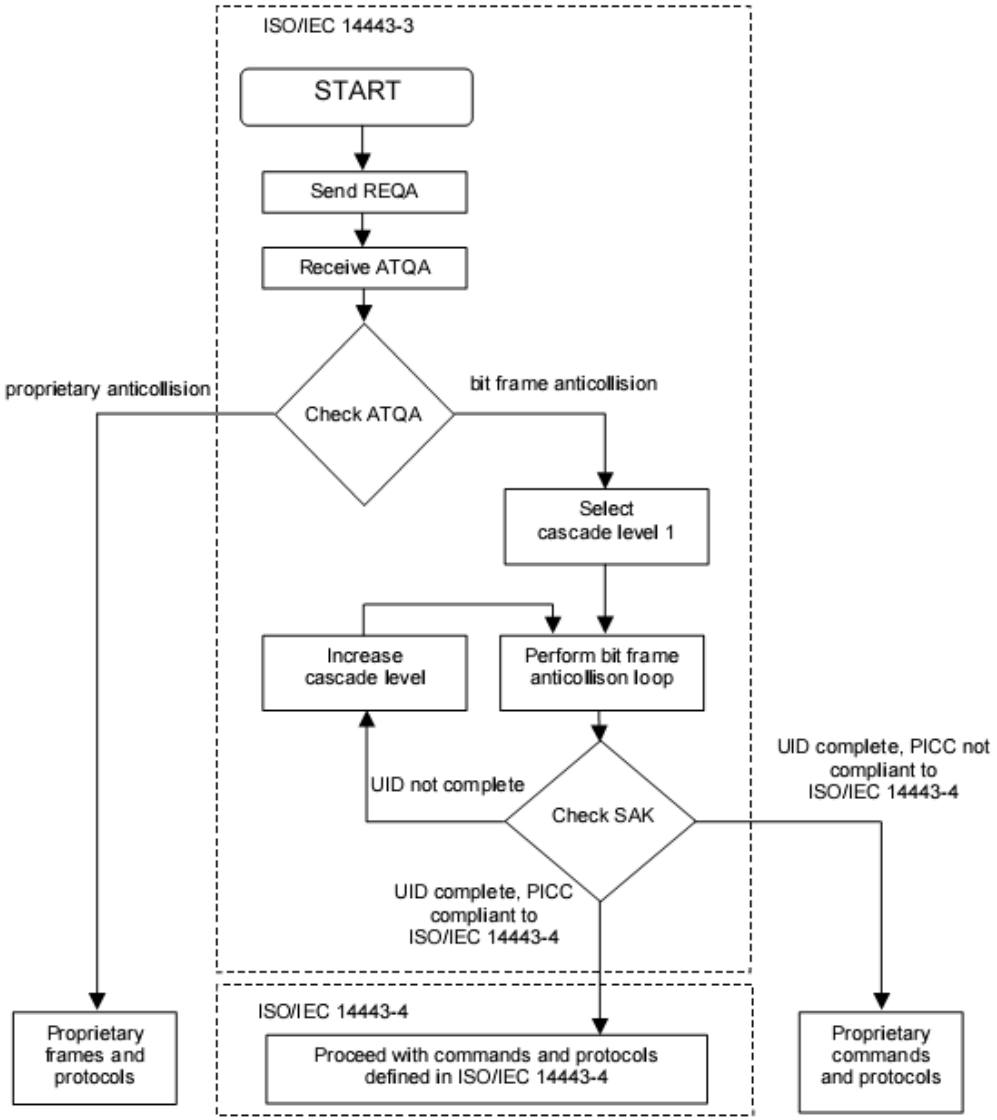
https://code.google.com/p/ifdnfc

Did I say it's still beta??

Anticollision

```
nfc-list
nfc-list -v
nfc-anticol
```

(only TypeA)



Anticollision

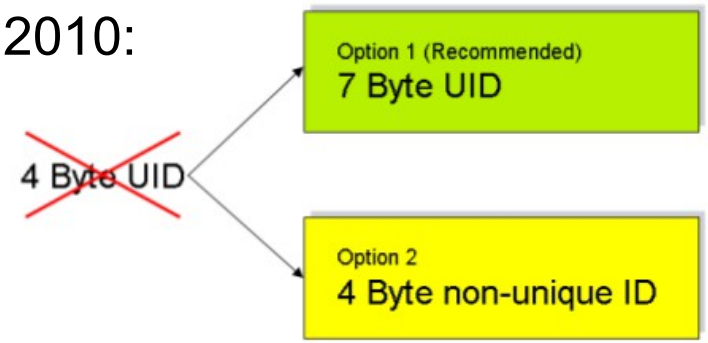
```
R: 26                                                  => REQA (7-bit)
T: 44   03                                             => ATQA (+anticol, double UID)
R: 93   20                                             => SEL (cascade level 1)
T: 88   04   34   74   cc                              => CT, UID(byte 1,2,3), BCC
R: 93   70   88   04   34   74   cc   0e   05          => SEL
T: 24   d8   36                                        => SAK (+cascade bit)
R: 95   20                                             => SEL (cascade level 2)
T: e1   e3   1c   80   9e                              => UID(byte 4,5,6,7), BCC
R: 95   70   e1   e3   1c   80   9e   b9   e1          => SEL
T: 20   fc   70                                        => SAK (14443-4 compliant)
R: e0   50   bc   a5                                   => RATS
T: 06   75   77   81   02   80   02   f0              => ATS
R: 50   00   57   cd                                   => HALT
```

UID

- 7-byte:
  Cascade Level 1: **88** **u1 u2 u3** (**u1=04** → NXP; **05** → Infineon,...)
  Cascade Level 2: **u4 u5 u6 u7**
- 4-byte:
  Cascade Level 1: **u1 u2 u3 u4**

  u1=08 → Random ID (used in card emulation, ePassports,...)

  u1=xF → FNUID = “F” Non-Unique ID
- MIFARE Classic since 2010:
- 11-byte foreseen in the standards



Reading/Writing raw tags & NDEF tags

```
nfc-mfultralight r foo
nfc-mfclassic r a foo
mifare-ultralight-info
mifare-classic-format
mifare-classic-write-ndef
mifare-classic-read-ndef -o foo
mifare-desfire-format
mifare-desfire-create-ndef
mifare-desfire-write-ndef
mifare-desfire-read-ndef -o foo
mifare-desfire-info
```

NFC: NFCIP1 p2p

Bring two readers against each other

On the first machine:

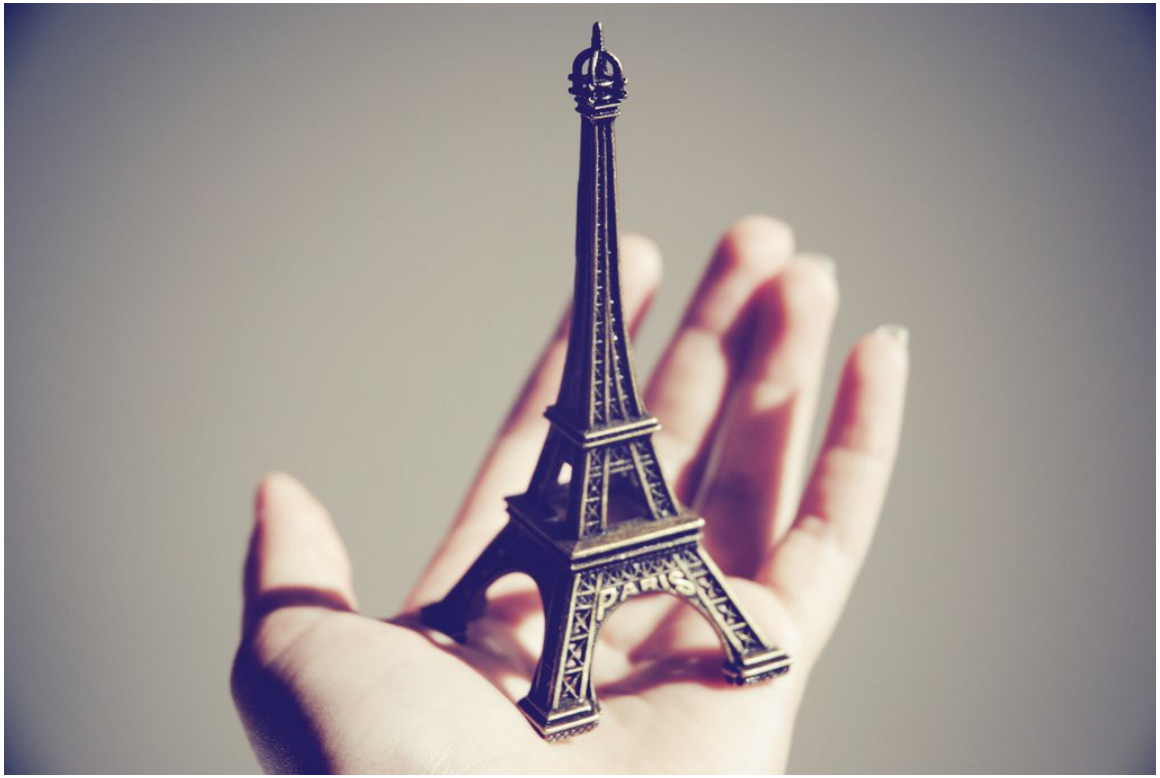
```
nfc-dep-target
```

On the second machine:

```
nfc-dep-initiator
```

NFC Security

NFC is intrinsically secure because it's short range



“about 15cm”

Seriously?

source: xaurorartx.deviantart.com

Short range?

Best results so far:

- Reader to card communication sniffed **22m** away
  - office environment, ISO14443 type A&B
- Card to reader communication sniffed **3.5m** away
  - office environment, typeB
- Reader to card communication sniffed **4m** away ***with an electric antenna***
  - so sniffing E rather than H

Pierre-Henri Thevenon's PhD thesis, 2011

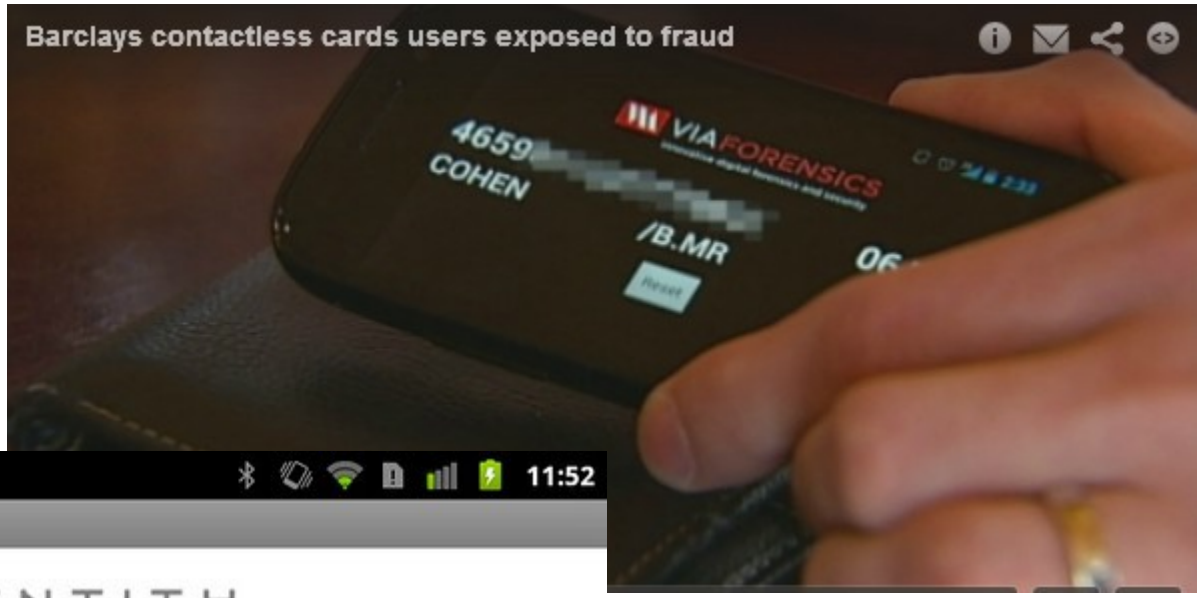
NFC “touch” & implicit user consent

Appealing but… dangerous!

- Privacy-leaking RFID tags
- Relay-attacks on tags & NFC devices
- Exploiting implicit intents on NFC devices

Privacy-leaking RFID tags

Contactless credit cards (US, UK)



14443A-4 relay attack via TCP/Bluetooth

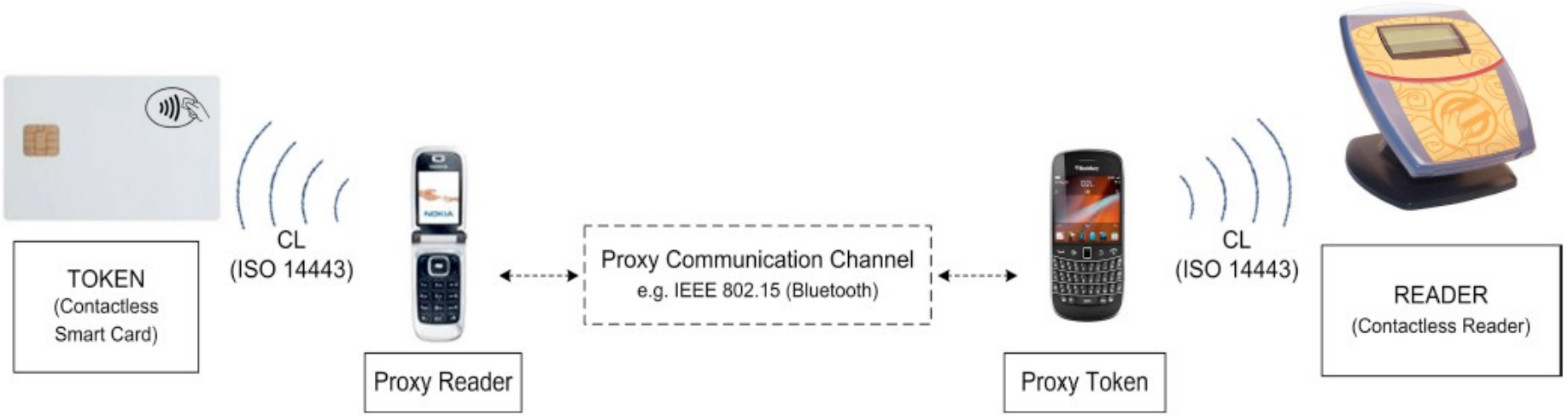
by Michael Weiß (2010)

http://www.sec.in.tum.de/student-work/publication/157

Idem with off-the-shelve NFC phones

by Lishoy Francis, Gerhard Hancke et al.

Using BlackBerry 9900 as proxy token



http://eprint.iacr.org/2011/618.pdf

CyanogenMod allowing sw card emulation

by Blackwing, Eddie Lee, 2012

http://sourceforge.net/p/nfcproxy

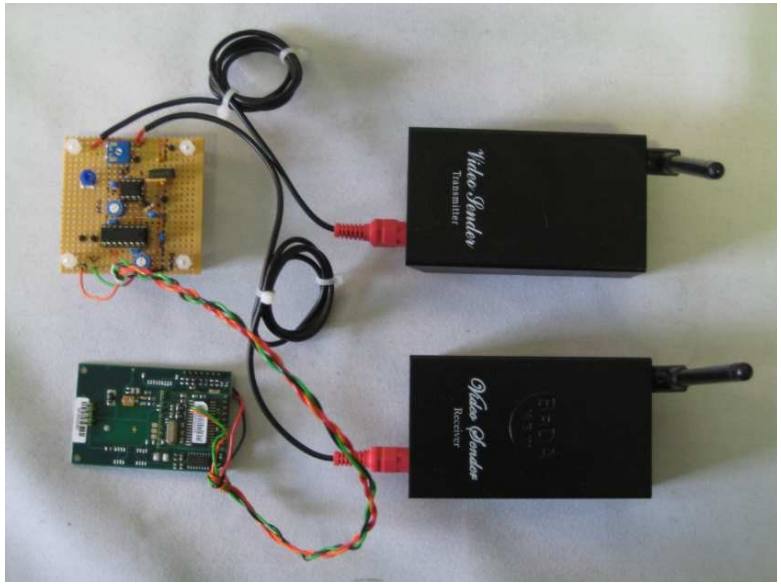
Libnfc:
Relay attacks

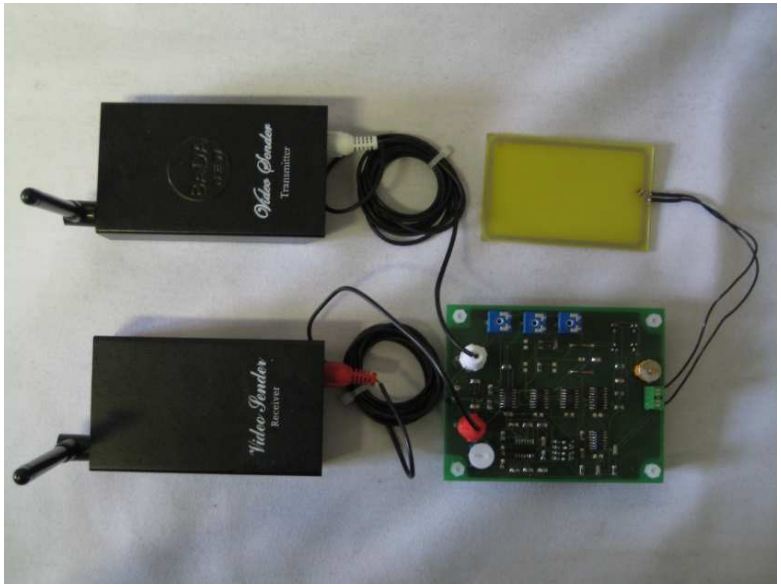
nfc-relay
- Sw card emulation

nfc-relay-picc
- PN532 only
- Works over TCP/IP

14443A relay attack via 2.4GHz video TX

by Gerhard P. Hancke (2009)

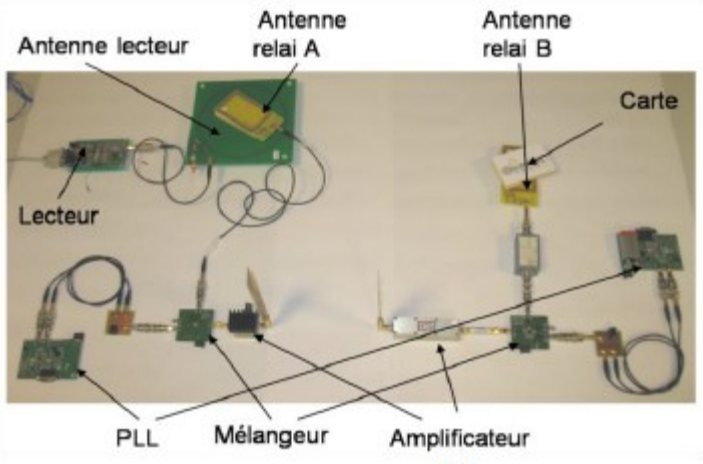


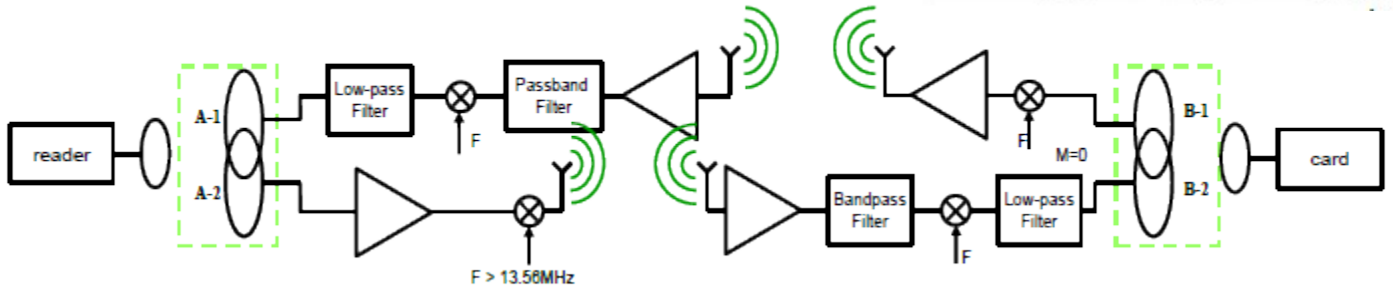


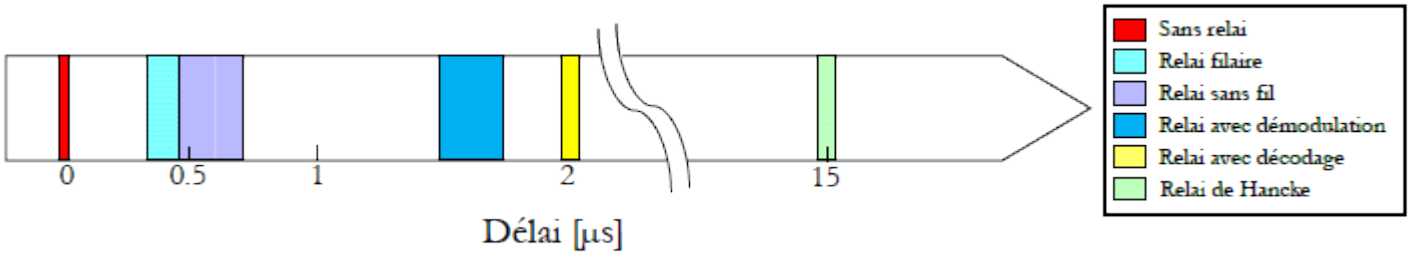
http://www.rfidblog.org.uk/Hancke-RelayOverview-2009.pdf

Fastest wireless relay so far

By Pierre-Henri Thevenon
about 700ns ~ 200m





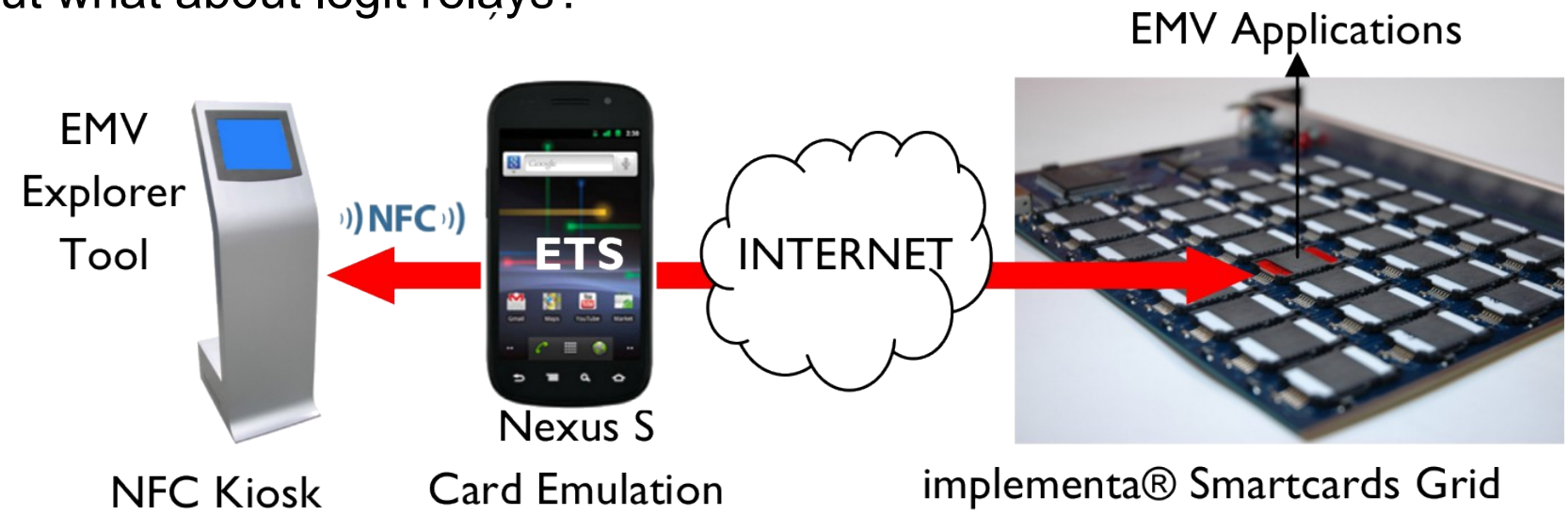


http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6064449

Defense against Relay Attacks

Distance bounding protocols
- Guarantee that the tag/device is not further away than X meters
- Based on timing of authentified unpredictable messages
- MIFARE Plus proximity check

But what about legit relays?



Source: Ethertrust

Exploiting implicit intents on NFC devices

Much cooler without user confirmation

But be careful…

E.g. Roel Verdult attack against Nokia 6212:

Fake smartposter (actually a p2p device) initiating BT
→ content sharing via OBEX
→ pushing malicious app
→ privilege escalation up to manufacturer/operator domain

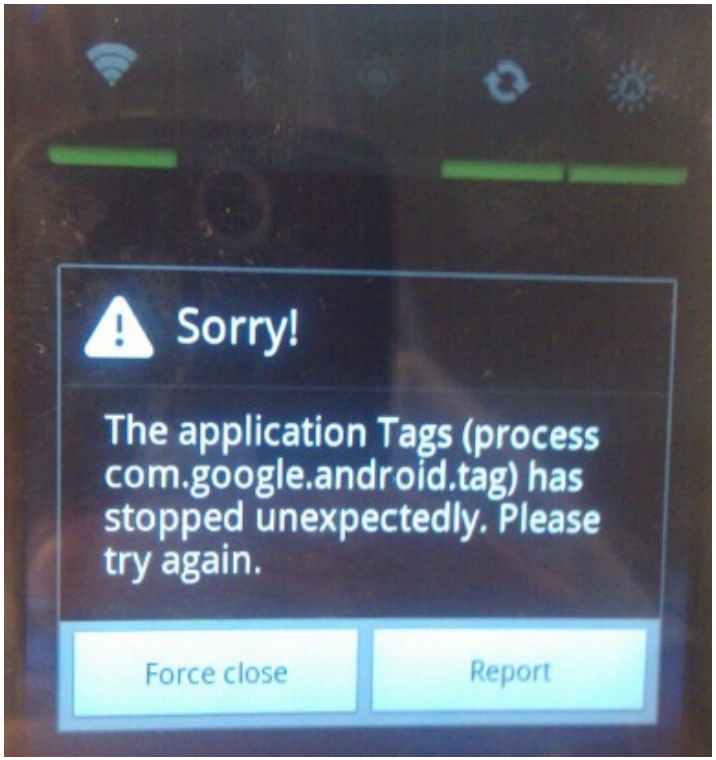
http://www.cs.ru.nl/~rverdult/Practical_attacks_on_NFC_enabled_cell_phones-NFC11.pdf

Today, still no confirmation to accept data from Android Beam...

But before asking for user intentions…

Fuzzing attacks with malformed NDEF attacks, exploiting NFC stack before any chance for user to accept or not.

NDEF fuzzing library by Colin Mulliner
http://www.mulliner.org/nfc



NFC “touch” attacks, what for?

- Crash system and/or app
  - Some could lead to successful exploits
- System targets: browser / dialer / sms handler
- Hijack phone by installing malicious apps
- Bugs & design issues => fraud?

Resources:

http://www.mulliner.org/nfc/feed/nfc_ndef_security_ninjacon_2011.pdf

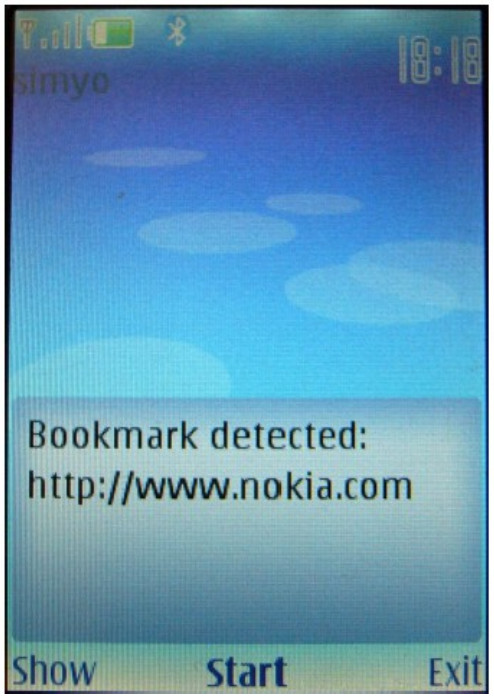
http://www.mulliner.org/collin/academic/publications/vulnanalysisattacknfcmobilephones_mulliner_2009.pdf

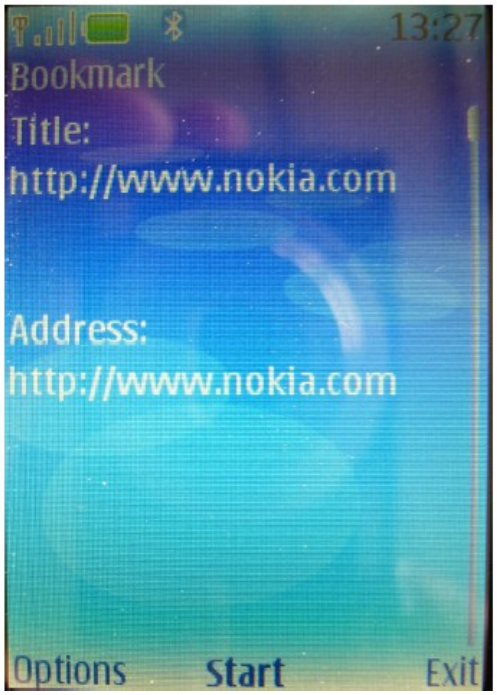
“NFC phishing” attacks on Smartposter

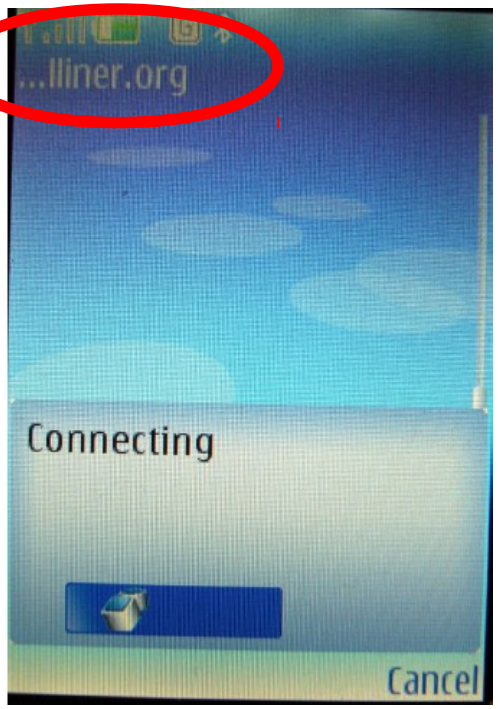
In this scenario the user needs to give his explicit consent

But… for what?

Smartposter URL: Abusing title field







Same attack on phone calls & SMS => redirect to surcharged call/SMS

Smartposter URL: Man-in-the-Middle Proxy

Transparent for the user as URL not displayed on mobile browsers

- Inject malicious content (e.g. auto-install trojan JAR bug in Nokia)
- Steal credentials
- etc

Smartposter URL: Attacking Selecta vendor machines

Make tags pointing to machine A and stick them on machine B, C, D, ...

Wait at machine A and pull out your free snack

Source: Colin Mulliner

More issues & mitigations

Problem of shortened URLs:

- Handy to store long URL on cheap NFC Tag1
- Is http://bit.ly/FHYSq safe or not??

Want to deploy smartposters?

- Use signed NDEF if possible (in its new version...)
- Turn tags physically read-only

NFC security

UIDs are unique, you can rely on them

seriously?

Well, not anymore

NFC technology barrier has fallen since a while for hackers…

- “One-shot” cheap designs
- Open-source dedicated designs
- Off-the-shelf NFC chips with open-source software
- Off-the-shelf readers & phones
- Industrial hacking products

Chinese MFC clone with R/W “UID”

About 25€

Can be fully reprogrammed, even if ACL data is corrupted

Quickly acquired by the “usual suspects” (nfc security researchers)

Quickly reversed and now supported by open-source tools

Libnfc:
Tag emulation

`nfc-emulate-uid`

Then from another machine try `nfc-anticol`

Try a second time `nfc-anticol`

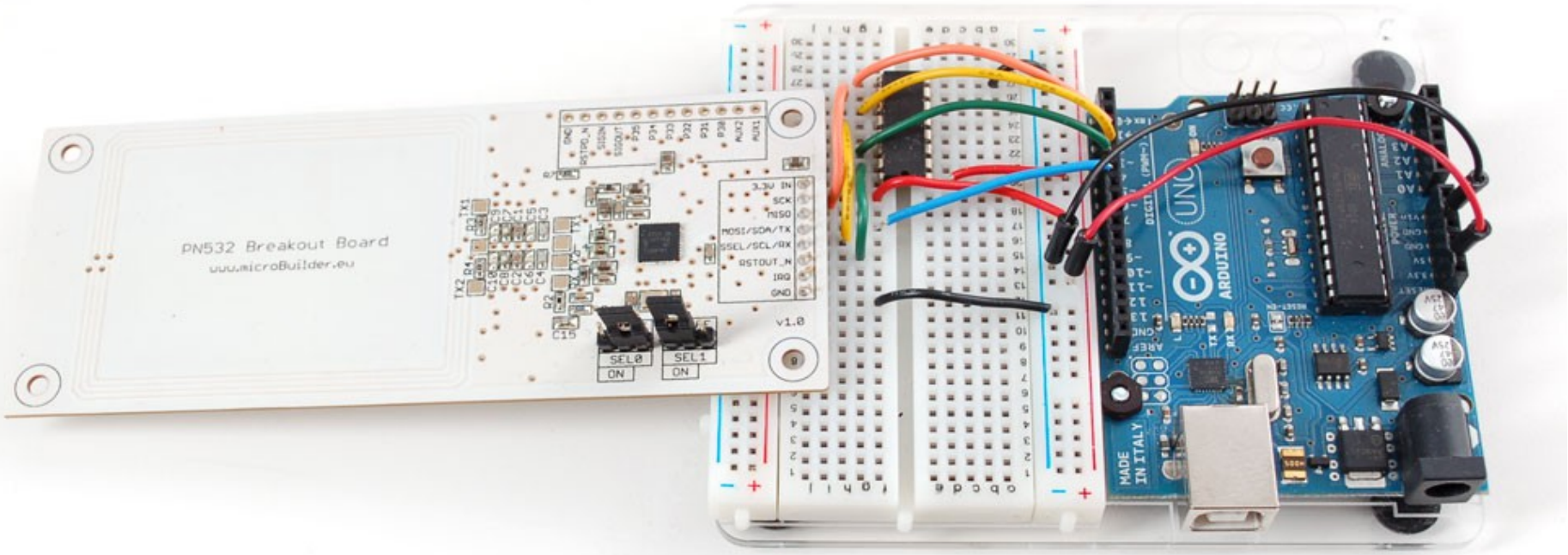
`nfc-emulate-forum-tag2`

`nfc-emulate-forum-tag4` (pn532 only)

PN532 breakout board

https://www.adafruit.com/products/364

$40 SPI, UART, I2C

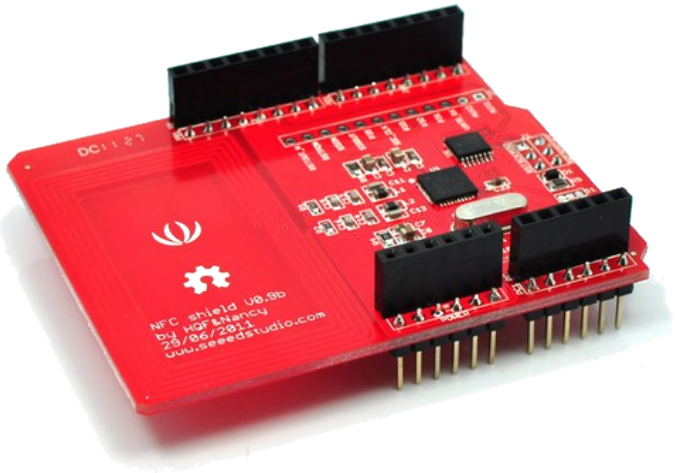


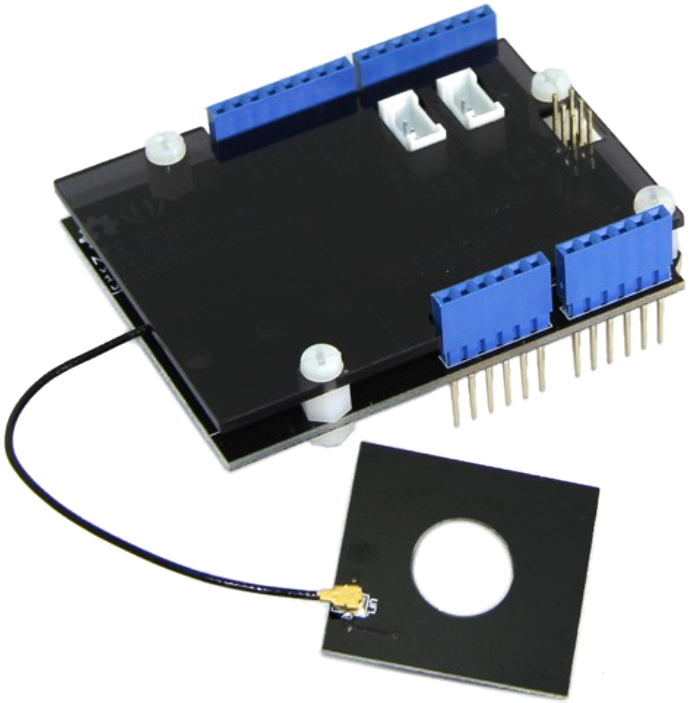
UID forgery still possible towards some readers

PN532 NFC shield for Arduino

http://seeedstudio.com/wiki/index.php?title=NFC_Shield

$29.50 SPI





PN532-based OpenPCD2

http://www.openpcd.org/OpenPCD_2_RFID_Reader_for_13.56MHz

- ARM Cortex-M3
- ∃ libnfc-compatible FW
- Read/Sniff/Emulate

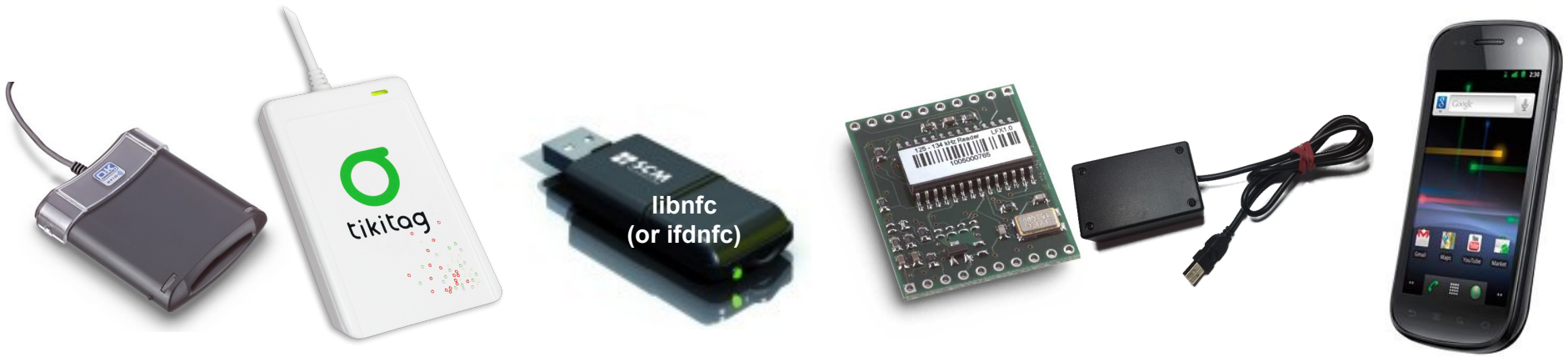
RFIDIOt

- Adam Laurie http://rfidiot.org
- many tools for LF & HF tags, some not much maintained

`isotype.py`

`multiselect.py`

  - try on epassport, then... `killall multiselect.py`



RFIDIOt
mrpkey.py

CHECK

<mrz|PLAIN> → copy in /tmp

- Supports short MRZ: 1-9;14-19;22-27
- Supports “???” in numeric part of document nr (demo)

Vonjeek/JMRTD applets:

<path> WRITE

<mrz|PLAIN> WRITE

SETBAC / UNSETBAC (Vonjeek only)

ePassport: Machine-Readable Zone & Basic Access Control

BAC key based on short MRZ: 1-9;14-19;22-27

ePassportViewer

UCL/GSI

`ePassportViewer`

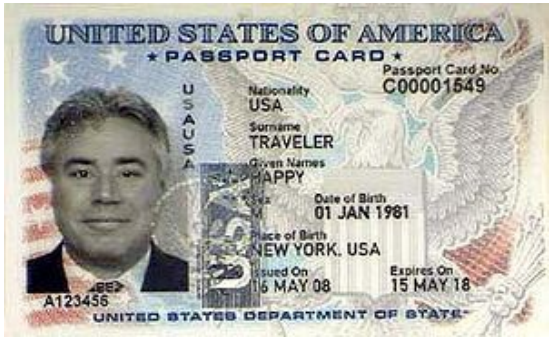
http://code.google.com/p/epassportviewer/ (not yet up-to-date)

Latest versions:

- CSCA
- Attacks
  - Err fingerprint
  - BAC Brute force
  - MAC traceability
  - Active Auth before BAC
- Forgery

ePassport != US Passport Card

- UHF



Chris Paget:

- $250 on eBay
  - Symbol XR400 RFID reader
  - Motorola AN400 patch antenna

cardpeek

- “L1L1”
- Extensible with LUA scripts
- EMV, Navigo, Calypso, Vitale, Moneo, ePP

`cardpeek`

http://code.google.com/p/cardpeek/

MOBIB Extractor

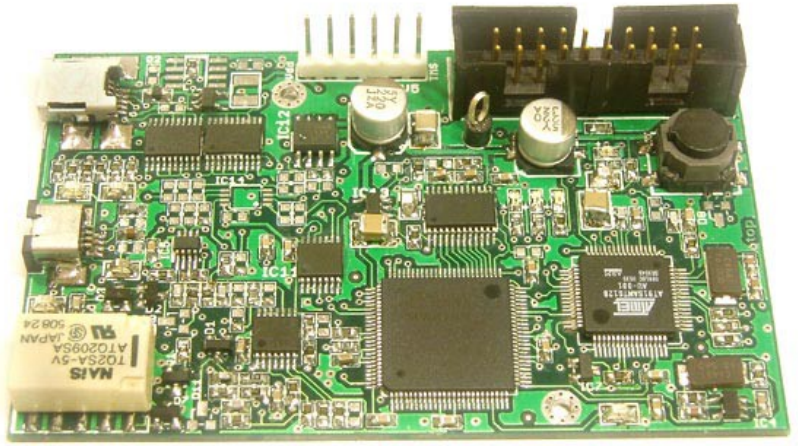
UCL/GSI

`MOBIB-Extractor`

Open /usr/local/lib/mobib-extractor/dump-sample.txt

http://sites.uclouvain.be/security/mobib.html

Proxmark III



- Jonathan Westhues
- 160€ / 188$ / 229$
- ARM7 + FPGA
- Opensource design & software (OS/ARM/FPGA)
- LF (125kHz / 132KHz) & HF (13.56MHz)
- Read, sniff (both directions), emulate & more

http://www.proxmark.org
https://code.google.com/p/proxmark3/wiki/RunningPM3

Proxmark III

~130 commands, half of them offline, readline support
- cf nfc-doc/applications/proxmark3/proxmark3-help*.txt
- Readers / sniffers / emulators / …

- **LF**:
  FlexPass, Indala, VeriChip, EM410x, EM4x50,
  HID Proxcard(*), TI, T55xx, Hitag
  (*) works standalone
- **HF**:
  ISO14443A, ISO14443B, SRI, ISO15693,
  Legic, iClass, MFC

PM3: LF analog trace demo

```
proxmark3
pm3> data load
/usr/local/share/proxmark3/traces/indala<TAB>
pm3> data plot
pm3> data dec
pm3> data load... (↑ to go back in history)
pm3> lf indalademod
```

PM3: Flashing latest firmware

```
cd /usr/local/share/proxmark3/firmware_r708
```

Proxmarks with still an old bootloader (SVN rev < 674):

```
flasher-old -b bootrom.elf fullimage.elf
```

Proxmarks with already new bootloader:

```
flasher fullimage.elf
```

Identifying unknown tag, first step:

```
pm3> hw tune
```

PM3: ISO14443A sniffing

```
pm3> hf 14a snoop
```

```
$ pn53x-tamashell
> 4a 01 00
```

```
$ pcsc_scan
```

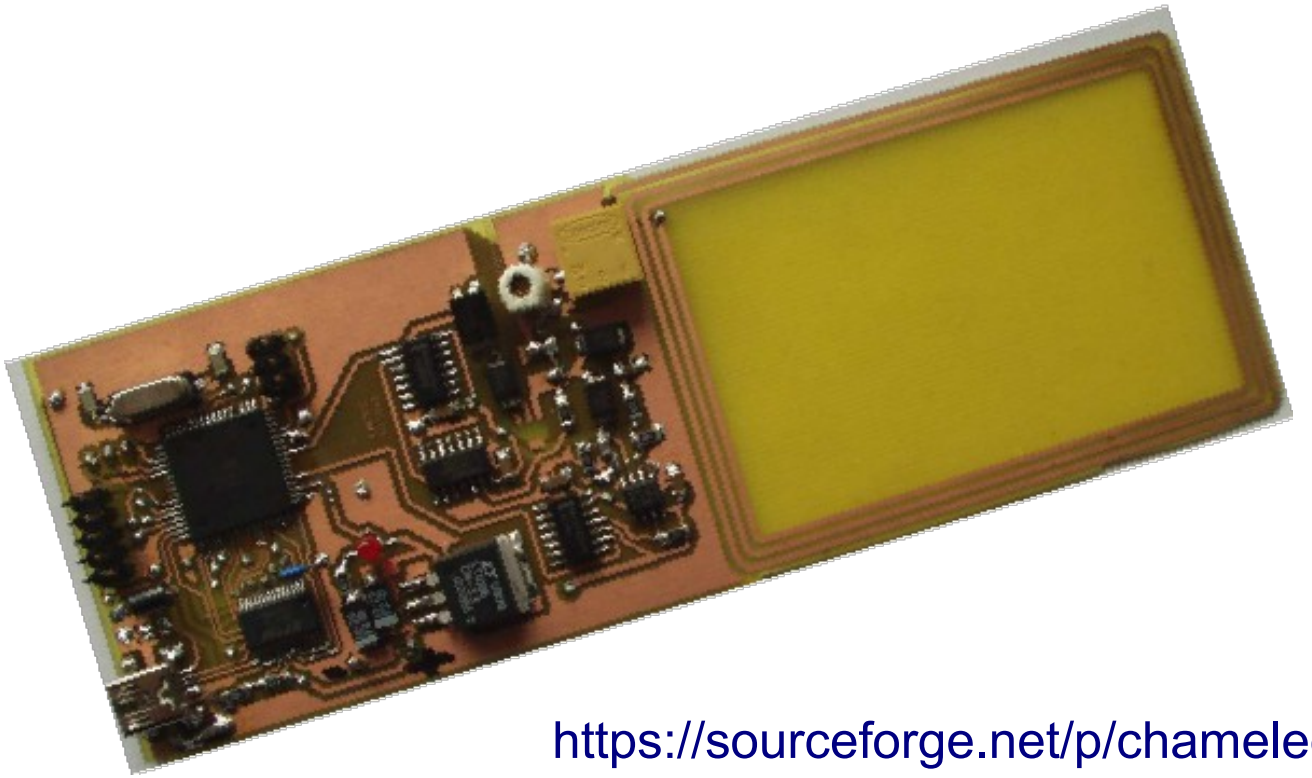
```
!! PRESS PM3 BUTTON TO STOP SNIFFING
```

```
pm3> hf 14a list
```

Chameleon: cloning MFC/DF for 25$

Mifare Classic, Desfire & DesfireEV1 emulation

Powered by battery, ATxmega192A3



https://sourceforge.net/p/chameleon14443/

Thanks!

- Want more?
  - http://nfc-live.googlecode.com
  - http://nfc-tools.org
  - http://wiki.yobi.be/wiki/RFID
- Feedback / Questions?
  - Now, or:
  - Jabber/GTalk: phil@jabber.reseaucitoyen.be
  - Email: phil@teuwen.org