EEPROM

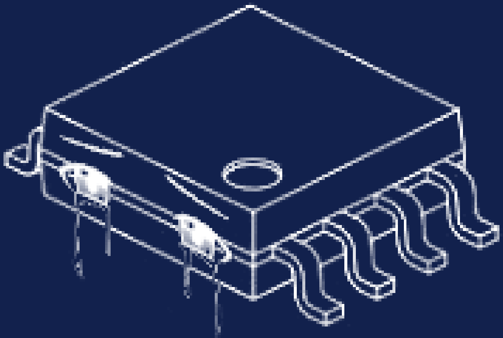
It Will All End in Tears

Philippe Teuwen
Christian Herrmann

29 October 2021, Den Haag

hardwear.io
Hardware Security Conference and Training

Quarkslab



Iceman

- .NET Developer with for 17+ Years
- Certified MCPD Enterprise Architect
- Founder of IceSQL AB
- RFID Evangelist
  - Major Proxmark3 Contributor since 2013
  - RFID Forum & Discord Admin
- Co-Founder of RFID Research Group
  - Co-Designer of Proxmark3 RDV4

Doegox

- A Team Leader of
  Crypto & Embedded Security at Quarkslab
- White-box crypto “grey box” attacks
- Hardware-oriented CTFs
- International Journal of PoC||GTFO
- Libnfc Maintainer
- Proxmark3 Contributor & RFID Discord Admin
- +20 NFC/RFID Security & Privacy Trainings

Iceman + Doegox ?

Happy consequence of Nullcon 2019

Thank you Antriksh & friends 💖

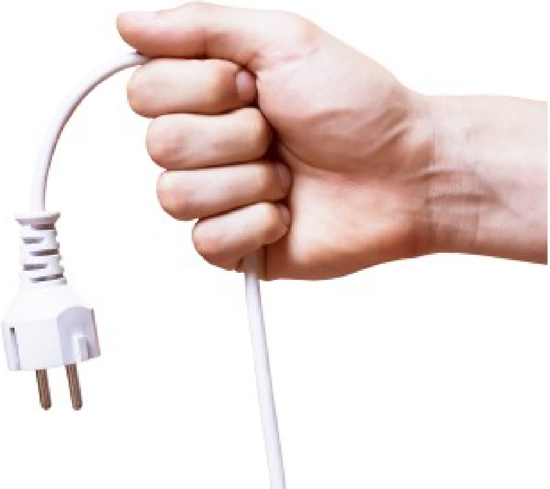
Physical tearing

Plan

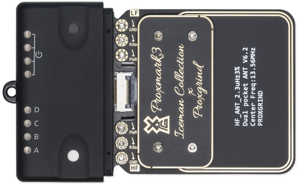
- Exploring RFID tearing events
- EEPROM physics
- How to control tearing effects
- Which security features to target
- Attack examples
- Tooling

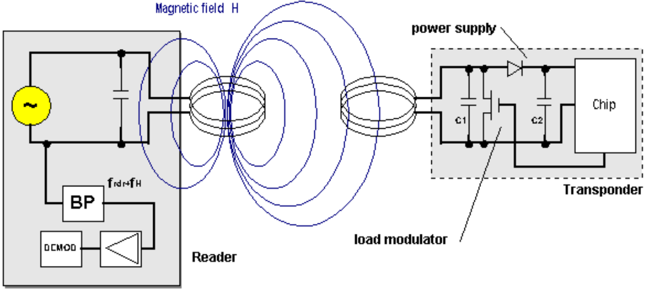
Approximative order...

Electrical tearing



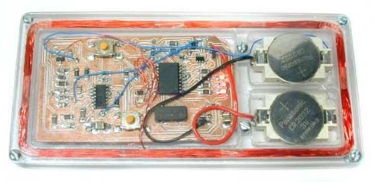
Toolbox: Proxmark3





http://RFID-handbook.com

Proxmark Evolution



Interrupting a WRITE command

Example: a MIFARE Ultralight

1. Choose a user memory address, e.g. block 4
2. Set an initial value
   - `WRITE(4, 0xFFFFFFFF)`
3. Launch a second write and interrupt it
   - `WRITE(4, 0xFFFFFFFF)`
   - Shutdown reader field after N μs
4. Read memory block
   - `READ(4)`
5. Adjust timings, goto step 2

Interrupting a WRITE command (0..3 ms)

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 200 μs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 400 μs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 600 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 800 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1000 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1200 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1400 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1600 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1800 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2000 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2200 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2400 μs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2600 μs → READ → FFFFFFFF

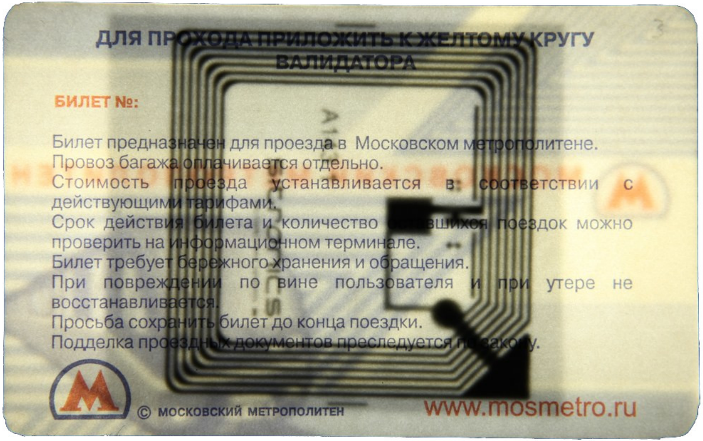
WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2800 μs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 3000 μs → READ → FFFFFFFF

Finding interesting targets

- Security features involving EEPROM *erase* and/or *write*
- That can be triggered by attacker
- But final result supposedly not under attacker control

Example



ДЛЯ ПРОХОДА ПРИЛОЖИТЬ К ЖЕЛТОМУ КРУГУ ВАЛИДАТОРА

БИЛЕТ №:

Билет предназначен для проезда в Московском метрополитене.
Провоз багажа оплачивается отдельно.
Стоимость проезда устанавливается в соответствии с действующими тарифами.
Срок действия билета и количество оставшихся поездок можно проверить на информационном терминале.
Билет требует бережного хранения и обращения.
При повреждении по вине пользователя и при утере не восстанавливается.
Просьба сохранить билет до конца поездки.
Подделка проездных документов преследуется по закону.

© МОСКОВСКИЙ МЕТРОПОЛИТЕН

www.mosmetro.ru

MIK640M2D, ”Ultralight” by Mikron

Example

Page address	
Decimal	Hexadecimal
0	0x00
1	0x01
2	0x02
3	0x03
4	0x04
5	0x05
...	...

Byte number inside page			
0	1	2	3
UID			
UID			
UID	Internally used data	Lock bytes 0 and 1	
OTP	OTP	OTP	OTP
User memory			

OTP = *One-Time Programmable* bits

Example: poorly implemented OTP

READ(3) → 0x12345678

WRITE(3, 0x00000001)

- *read*(3) = 0x12345678
- 0x12345678 OR 0x00000001 = 0x12345679
- *write*(3, 0x12345679)

READ(3) → 0x12345679

Example: poorly implemented OTP

READ(3) → 0x12345678

WRITE(3, 0x00000001)

- *read*(3) = 0x12345678
- 0x12345678 OR 0x00000001 = 0x12345679
- *erase*(3)
- *write*(3, 0x12345679)

READ(3) → 0x12345679

Example: poorly implemented OTP

READ(3) → 0x12345678

WRITE(3, 0x00000001)

- *read*(3) = 0x12345678
- 0x12345678 OR 0x00000001 = 0x12345679
- *erase*(3)
- TEAR-OFF before *write*(3, 0x12345679)

READ(3) → 0x00000000

Attack published by Grisolìa and Ukmar in 2020 in their student thesis

Demo

Interrupting a WRITE command (0..3 ms)

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 200 µs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 400 µs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 600 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 800 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1000 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1200 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1400 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1600 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 1800 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2000 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2200 µs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2400 µs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2600 µs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2800 µs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 3000 µs → READ → FFFFFFFF

Tear-off during first transition phase

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 546 μs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 548 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 550 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 552 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 554 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 556 μs → READ → F3DFF7FB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 558 μs → READ → F1CFF6FB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 560 μs → READ → F0CF76FB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 562 μs → READ → E0CF42DB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 564 μs → READ → E0010003

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 566 μs → READ → 60010003

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 568 μs → READ → 60010001

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 570 μs → READ → 60000001

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 572 μs → READ → 20000001

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 574 μs → READ → 20000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 576 μs → READ → 00000000

Tear-off during second transition phase

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2356 μs → READ → 00000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2358 μs → READ → 0235005B

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2360 μs → READ → 8275007B

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2362 μs → READ → 8AFD00FB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2364 μs → READ → 8EFD00FB

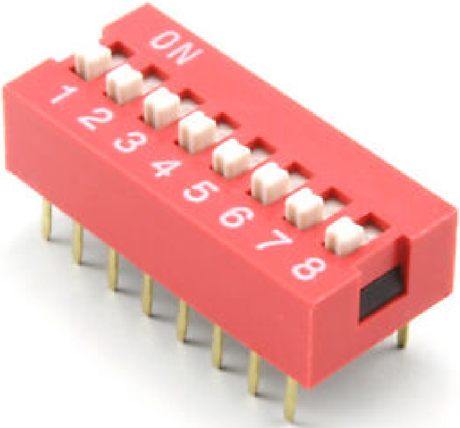
WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2366 μs → READ → AFFD3DFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2368 μs → READ → AFFFBFFF

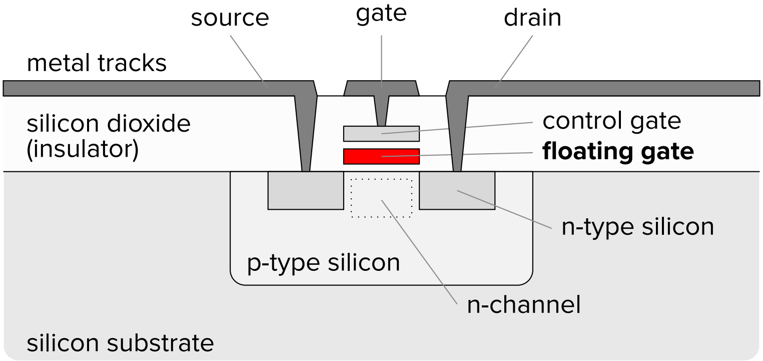
WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2370 μs → READ → EFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 2372 μs → READ → FFFFFFFF

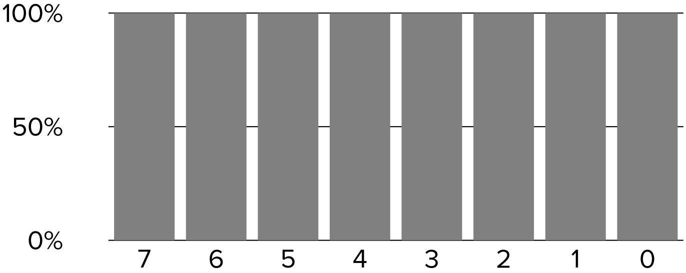
Bad analogy



EEPROM Transistor



Erasing an EEPROM Byte

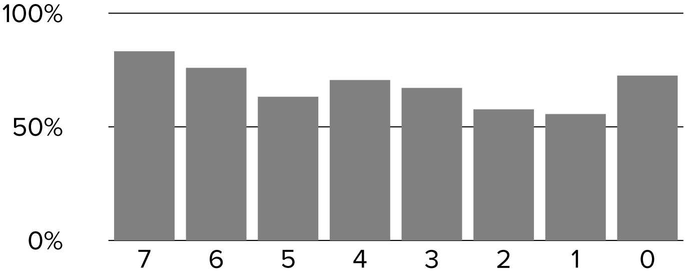


readout at 50%:

`11111111`

`= 0xFF`

Erasing an EEPROM Byte

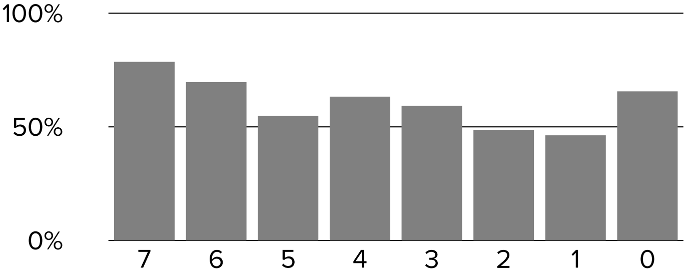


readout at 50%:

`11111111`

`= 0xFF`

Erasing an EEPROM Byte

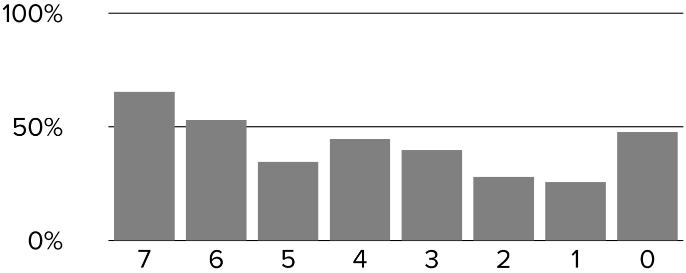


readout at 50%:

`11111001`

`= 0xF9`

Erasing an EEPROM Byte

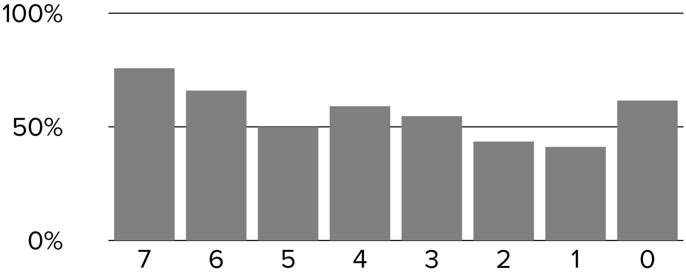


readout at 50%:

`11000000`

`= 0xC0`

Erasing an EEPROM Byte: weak bits

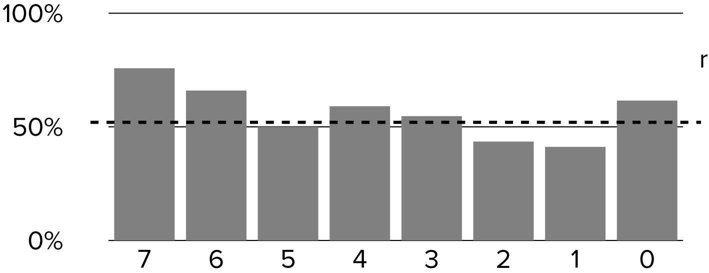


readout at 50%:

`11?11001`

`= ??`

Erasing an EEPROM Byte: weak bits

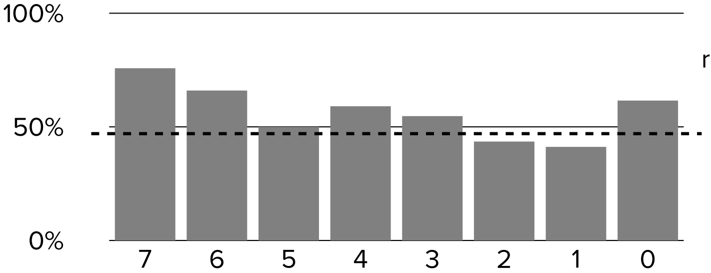


readout at 52%:

```
11011001

= 0xD9
```

Erasing an EEPROM Byte: weak bits



readout at 47%:

`11111001`

`= 0xF9`

Tear-off during first transition phase

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 546 μs → READ → FFFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 548 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 550 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 552 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 554 μs → READ → FBFFFFFF

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 556 μs → READ → F3DFF7FB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 558 μs → READ → F1CFF6FB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 560 μs → READ → F0CF76FB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 562 μs → READ → E0CF42DB

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 564 μs → READ → E0010003

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 566 μs → READ → 60010003

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 568 μs → READ → 60010001

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 570 μs → READ → 60000001

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 572 μs → READ → 20000001

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 574 μs → READ → 20000000

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 576 μs → READ → 00000000

Progressive Tear-off during first phase

WRITE FFFFFFFF → WRITE FFFFFFFF with *tearing* at 500 μs → READ → FFFFFFFF

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → FFFFFFFF

repeated 20 times, still no visible change, then...

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → FBFFFFFF

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → FBFFFFFF

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → FBFFFFFF

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → FBFFF7FF

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → FBEFF7FB

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → FACFF7FB

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → F8CDF7FB

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → E8CD76FB

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → E00540FB

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → E00140FB

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → E00140D9

→ WRITE FFFFFFFF with *tearing* at 300 μs → READ → ...

Controlling EEPROM erase/write

- Tear-off between *erase* & *write* operations
  - Check logic of erased word: all *zeroes* or all *ones*
- Tear-off during *erase* or *write* operations
  - Statistic bias across bits
  - Possibility of fingerprinting
- Progressive tear-off during first operation for finer control

Controlling EEPROM erase/write

- Tear-off between *erase* & *write* operations
  - Check logic of erased word: all *zeroes* or all *ones*
- Tear-off during *erase* or *write* operations
  - Statistic bias across bits
  - Possibility of fingerprinting
- Progressive tear-off during first operation for finer control
- Timings influenced by
  - Distance to the reader
  - Temperature
  - Content to be erased/written

Controlling EEPROM read of weak bits

- Distance to the reader, e.g.
  - 1 close to the reader
  - 0 far away

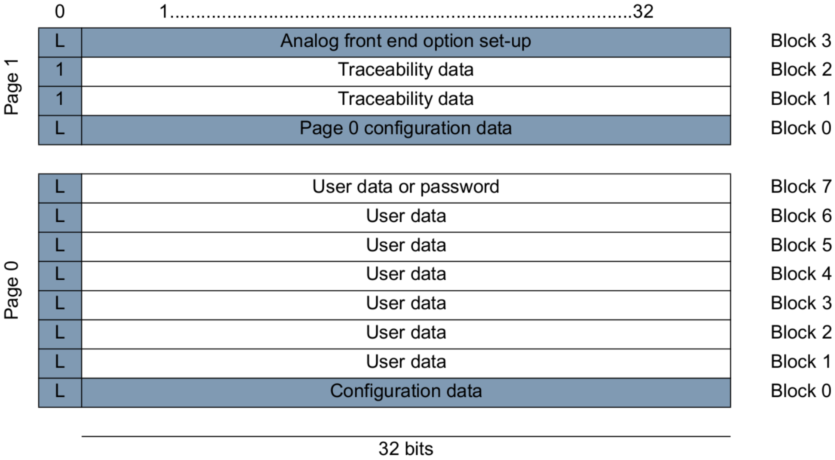
Controlling EEPROM read of weak bits

- Distance to the reader, e.g.
  - 1 close to the reader
  - 0 far away
- Time since powering, e.g.
  - 0 if read immediately after the card gets powered
  - 1 if read later
- $\Rightarrow$ Combine controls, e.g.
  - 0 if far away and read immediately
  - 1 if close and read later

Second example

- LF tag
- ATA5577C: generic read/write IDentification IC
- Lockable with a password

ATA5577C

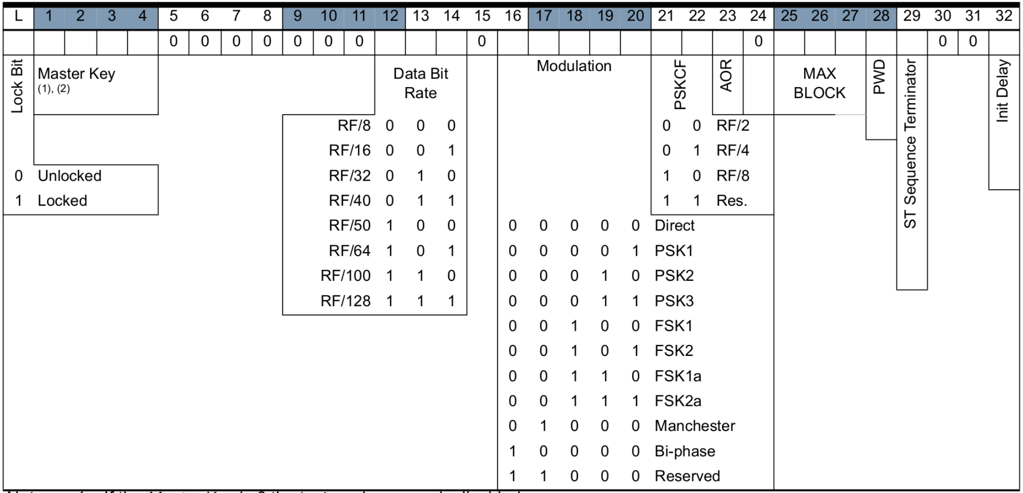


	0	1...32	
Page 1	L	Analog front end option set-up	Block 3
	1	Traceability data	Block 2
	1	Traceability data	Block 1
	L	Page 0 configuration data	Block 0
Page 0	L	User data or password	Block 7
	L	User data	Block 6
	L	User data	Block 5
	L	User data	Block 4
	L	User data	Block 3
	L	User data	Block 2
	L	User data	Block 1
	L	Configuration data	Block 0

32 bits

▨ Not transmitted

ATA5577C Configuration Block



L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
					0	0	0	0	0	0	0				0									0						0	0	
Lock Bit	Master Key (1), (2)											Data Bit Rate				Modulation					PSKCF		AOR		MAX BLOCK			PWD	ST Sequence Terminator			Init Delay

Lock Bit	
0	Unlocked
1	Locked

Data Bit Rate	12	13	14
RF/8	0	0	0
RF/16	0	0	1
RF/32	0	1	0
RF/40	0	1	1
RF/50	1	0	0
RF/64	1	0	1
RF/100	1	1	0
RF/128	1	1	1

16	17	18	19	20	Modulation
0	0	0	0	0	Direct
0	0	0	0	1	PSK1
0	0	0	1	0	PSK2
0	0	0	1	1	PSK3
0	0	1	0	0	FSK1
0	0	1	0	1	FSK2
0	0	1	1	0	FSK1a
0	0	1	1	1	FSK2a
0	1	0	0	0	Manchester
1	0	0	0	0	Bi-phase
1	1	0	0	0	Reserved

21	22	PSKCF
0	0	RF/2
0	1	RF/4
1	0	RF/8
1	1	Res.

Notes: 1. If the *Master Key* is 6 the test mode access is disabled

2. If the *Master Key* is neither 6 nor 9, the extended function mode and *Init Delay* are disabled

ATA5577C Password Protection

- 1 bit in Configuration word → Block 7 data becomes a mandatory password
- *Test-mode* hidden command to write patterns in the whole memory

ATA5577C Password Protection

Quarkslab

- 1 bit in Configuration word → Block 7 data becomes a mandatory password
- *Test-mode* hidden command to write patterns in the whole memory

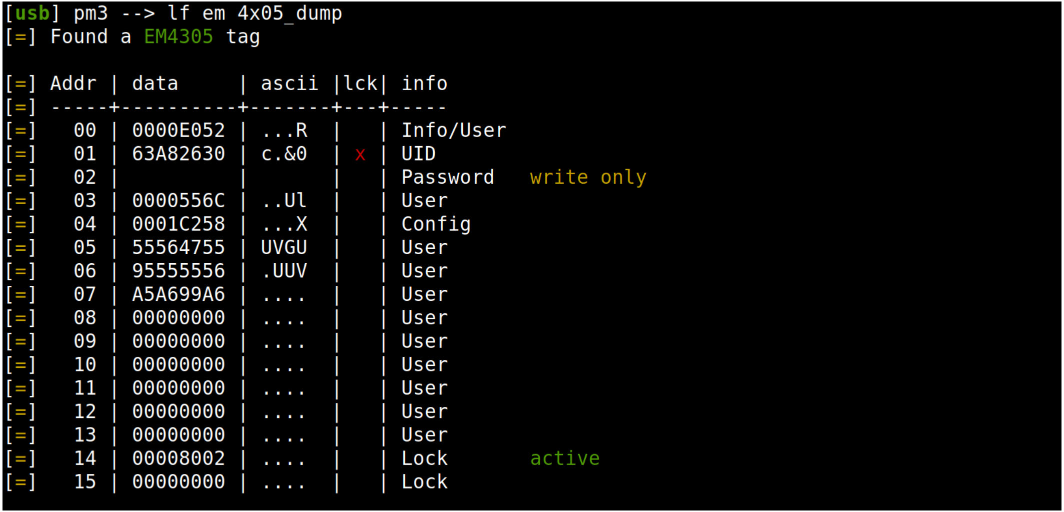
Strategy: (destructive, foresee a few cards with the same password)

- Tear a test-mode during *erase* phase → few bits cleared across memory
- Repeat with progressive tearing till password protection configuration bit is cleared
- Overwrite configuration to stabilize it
- Read partially erased password (use tips to force weak bits towards 1)
- Repeat on other cards, bruteforce the rest if needed

Third example

- Another LF tag
- EM4305: generic read/write IDentification IC
- Read-only UID

EM4305



```
[usb] pm3 --> lf em 4x05_dump
[=] Found a EM4305 tag

[=] Addr | data     | ascii |lck| info
[=] -----+----------+-------+---+-----
[=]   00 | 0000E052 | ...R  |   | Info/User
[=]   01 | 63A82630 | c.&0  | x | UID
[=]   02 |          |       |   | Password   write only
[=]   03 | 0000556C | ..Ul  |   | User
[=]   04 | 0001C258 | ...X  |   | Config
[=]   05 | 55564755 | UVGU  |   | User
[=]   06 | 95555556 | .UUV  |   | User
[=]   07 | A5A699A6 | ....  |   | User
[=]   08 | 00000000 | ....  |   | User
[=]   09 | 00000000 | ....  |   | User
[=]   10 | 00000000 | ....  |   | User
[=]   11 | 00000000 | ....  |   | User
[=]   12 | 00000000 | ....  |   | User
[=]   13 | 00000000 | ....  |   | User
[=]   14 | 00008002 | ....  |   | Lock       active
[=]   15 | 00000000 | ....  |   | Lock
```

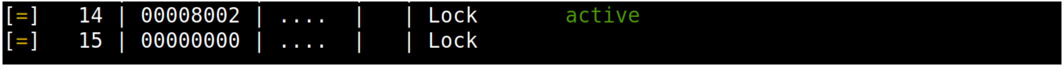
EM4305 Protection Words

- ”Write locking” configuration blocks
- When a bit is set, it locks the corresponding memory word
- Acts like OTP → a lock can’t be cleared
- Last bit indicates which Protection Word is active

EM4305 Protection Words

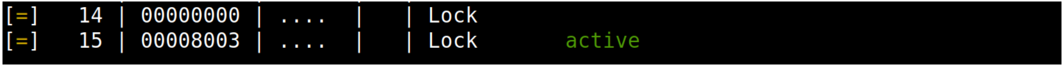
- ”Write locking” configuration blocks
- When a bit is set, it locks the corresponding memory word
- Acts like OTP → a lock can’t be cleared
- Last bit indicates which Protection Word is active

E.g. `PROTECT(0x00000001)` to lock first Word



```
[=]    14 | 00008002 | ....   |    | Lock         active
[=]    15 | 00000000 | ....   |    | Lock
```

⇓

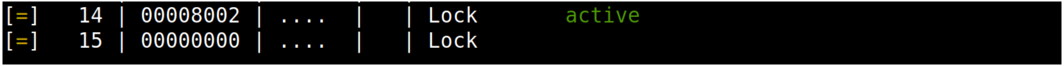


```
[=]    14 | 00000000 | ....   |    | Lock
[=]    15 | 00008003 | ....   |    | Lock         active
```

EM4305 Protection Words

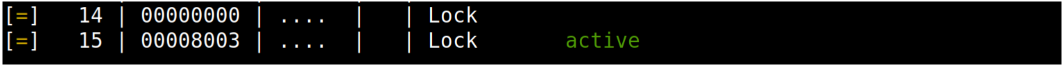
- ”Write locking” configuration blocks
- When a bit is set, it locks the corresponding memory word
- Acts like OTP → a lock can’t be cleared
- Last bit indicates which Protection Word is active

E.g. `PROTECT(0x00000001)` to lock first Word



```
[=]    14 | 00008002 | ....   |    | Lock        active
[=]    15 | 00000000 | ....   |    | Lock
```

⇓



```
[=]    14 | 00000000 | ....   |    | Lock
[=]    15 | 00008003 | ....   |    | Lock        active
```

Should the operation be interrupted for any reason (e.g. tag removal from the field) the double buffer scheme ensures that no unwanted ”0”-Protection Bits (i.e unprotected words) are introduced. – EM4305 datasheet

Defeating Protection Words

- Launch and interrupt a `PROTECT` command
- Hope for a Protection Word with `0x00008000`

Defeating Protection Words

- Launch and interrupt a `PROTECT` command
- Hope for a Protection Word with `0x00008000`

⇒ Both Protection Words become active

- The same one has always priority
- → Start with the other one being active

Defeating Protection Words

- Launch and interrupt a `PROTECT` command
- Hope for a Protection Word with `0x00008000`

⇒ Both Protection Words become active

- The same one has always priority
- → Start with the other one being active

⇒ Complex strategy loop

- Adjust timings
- Deal with all outcomes and corner cases (weak bits)
- Restart from stable situation

Defeating Protection Words

- Launch and interrupt a `PROTECT` command
- Hope for a Protection Word with `0x00008000`

⇒ Both Protection Words become active

- The same one has always priority
- → Start with the other one being active

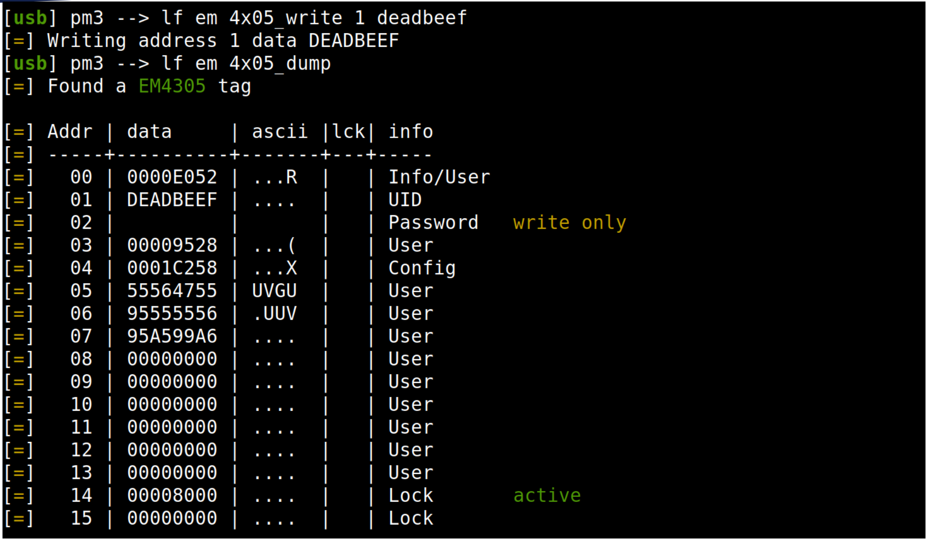
⇒ Complex strategy loop

- Adjust timings
- Deal with all outcomes and corner cases (weak bits)
- Restart from stable situation

⇒ Automated attack: few seconds to few minutes
Success rate: about 85%

Demo

Unlocked EM4305



```
[usb] pm3 --> lf em 4x05_write 1 deadbeef
[=] Writing address 1 data DEADBEEF
[usb] pm3 --> lf em 4x05_dump
[=] Found a EM4305 tag

[=] Addr | data     | ascii |lck| info
[=] -----+----------+-------+---+-----
[=]   00 | 0000E052 | ...R  |   | Info/User
[=]   01 | DEADBEEF | ....  |   | UID
[=]   02 |          |       |   | Password   write only
[=]   03 | 00009528 | ...(  |   | User
[=]   04 | 0001C258 | ...X  |   | Config
[=]   05 | 55564755 | UVGU  |   | User
[=]   06 | 95555556 | .UUV  |   | User
[=]   07 | 95A599A6 | ....  |   | User
[=]   08 | 00000000 | ....  |   | User
[=]   09 | 00000000 | ....  |   | User
[=]   10 | 00000000 | ....  |   | User
[=]   11 | 00000000 | ....  |   | User
[=]   12 | 00000000 | ....  |   | User
[=]   13 | 00000000 | ....  |   | User
[=]   14 | 00008000 | ....  |   | Lock       active
[=]   15 | 00000000 | ....  |   | Lock
```

Proxmark3 Tear-off Support

Chip/Standard	Command
MIK640M2D	`hf mfu otptear` (automated)
ATA5577C	`lf t55xx dangerraw`
EM4x05	`lf em 4x05_unlock` (automated)
EM4x05	`hw tearoff` combined with `lf em 4x05_write`
EM4x50	`hw tearoff` combined with `lf em 4x50_write`
ISO14443A	`hw tearoff` combined with `hf 14a raw`
ISO14443B	`hw tearoff` combined with `hf 14b raw`
ISO15693	`hw tearoff` combined with `hf 15 raw`
iClass	`hw tearoff` combined with `hf iclass wrbl`

Demo

Fourth example

- HF tag
- MIFARE Ultralight EV1 / NTAG Family
- Monotonic counters

MIFARE Ultralight EV1

Three 24-bit monotonic counters with anti-tearing support

- `INCR_CNT`
- `READ_CNT`
- `CHECK_TEARING_EVENT`

MIFARE Ultralight EV1

Three 24-bit monotonic counters with anti-tearing support

- `INCR_CNT`
- `READ_CNT`
- `CHECK_TEARING_EVENT`

⇒ Saved internally in 2 slots, a bit like EM4305 Protection Words, but:

- Slots: not readable directly
- Validity flag: a full byte (=`0xBD`)
- Priority: if both slots are valid, it returns the highest counter
- Evidence: Command to detect tearing event

MFUL EV1 Counter Examples

Slot	Flag	Value	Active

⟹

READ_CNT	CHECK_TEAR...

0x000123 + 1 in normal conditions

A	0xBD	0x000123	
B	0xBD	0x000124	★

	0xBD
0x000124	

0x000123 + 1 interrupted

A	0xBD	0x000123	★
B	0x98	¿0x000124?	

0x000123	
	0x98

MFUL EV1 Experiments

We need a valid flag byte (`0xBD`)
⇒ Testing some tearing on an `INCR_CNT` near the end of the operation
⇒ Got the following:

- `CHECK_TEARING_EVENT` returning `0xBD` but
- `READ_CNT` returning the **old** counter value

MFUL EV1 Experiments

Possible explanation:

Slot	Flag	Value	Active	⟹	READ_CNT	CHECK_TEAR...

0x000123 + 1 in normal conditions

A	0xBD	0x000123				0xBD
B	0xBD	0x000124	★		0x000124	

0x000123 + 1 interrupted

A	0xBD	0x000123	★		0x000123	
B	0xBD	¿0x000104?				0xBD

MFUL EV1 Counter Strategy

- Bump counter to next $2^N - 1$ (0x000123→0x0001FF)
- INCR_CNT(0) to copy it to the other slot
- INCR_CNT(1) and tear, hope for a weak bit

Slot	Flag	Value	Active	⟹	READ_CNT	CHECK_TEAR...

Initial values, B gets priority

A	0xBD	0x0001FF				0xBD
B	0xBD	0x0001FF	★		0x0001FF	

After $+1$ interrupted late

A	0xBD	0x000?00	??		??	0xBD
B	0xBD	0x0001FF				

MFUL EV1 Counter Strategy

Slot	Flag	Value	Active	$\Longrightarrow$	READ_CNT	CHECK_TEAR...

Weak bit in 2^N counter

A	0xBD	0x000?00	??		??	0xBD
B	0xBD	0x0001FF				

When read close to reader → weak bit $= 1$

A	0xBD	0x000?00	★		0x000200	
B	0xBD	0x0001FF				0xBD

When read far from reader → weak bit $= 0$

A	0xBD	0x000?00				0xBD
B	0xBD	0x0001FF	★		0x0001FF	

MFUL EV1 Counter Strategy

If no weak bit at 2^N

- Try again a few times
- Then try from $2^{N+1} - 1$: $\texttt{0x0003FF}$, $\texttt{0x0007FF}$, $\texttt{0x000FFF}$,...
- Reaching $2^N + 1$, $2^N + 2$?
  That's fine... $\texttt{0x00?002} \rightarrow \texttt{0x000002} \rightarrow \texttt{0x000FFF} \rightarrow \texttt{0x00?000} \rightarrow \texttt{0x000000}$

MFUL EV1 Counter Strategy

If no weak bit at 2^N

- Try again a few times
- Then try from $2^{N+1} - 1$: $\texttt{0x0003FF}$, $\texttt{0x0007FF}$, $\texttt{0x000FFF}$,...
- Reaching $2^N + 1$, $2^N + 2$?
  That's fine... $\texttt{0x00?002} \rightarrow \texttt{0x000002} \rightarrow \texttt{0x000FFF} \rightarrow \texttt{0x00?000} \rightarrow \texttt{0x000000}$

How to move

- from $\texttt{0x0001FF} \Leftrightarrow \texttt{0x000200}$
- to $\texttt{0x000000}$?

MFUL EV1 Counter Strategy

Slot	Flag	Value	Active	$\Longrightarrow$	READ_CNT	CHECK_TEAR...

Card close to reader → weak bit $= 1$

A	0xBD	0x000?00	★	0x000200	
B	0xBD	0x0001FF			0xBD

After $+0$ interrupted soon → other slot gets corrupted

A	0xBD	0x000?00	★	0x000200	
B	0x98	??			0x98

But when read far from reader → weak bit $= 0$

A	0xBD	0x000?00	★	0x000000	
B	0x98	??			0x98

MFUL EV1 Counter Strategy

Slot | Flag | Value | Active ⟹ READ_CNT | CHECK_TEAR...

Card far from reader → weak bit $= 0$

Slot	Flag	Value	Active	READ_CNT	CHECK_TEAR...
A	0xBD	0x000?00	★	0x000000	
B	0x98	??			0x98

After $+0$, B gets priority

Slot	Flag	Value	Active	READ_CNT	CHECK_TEAR...
A	0xBD	0x000?00			0xBD
B	0xBD	0x000000	★	0x000000	

But when read close to reader → weak bit $= 1$

Slot	Flag	Value	Active	READ_CNT	CHECK_TEAR...
A	0xBD	0x000?00	★	0x000200	
B	0xBD	0x000000			0xBD

MFUL EV1 Counter Strategy

Slot	Flag	Value	Active	$\Longrightarrow$	READ_CNT	CHECK_TEAR...

Card far from reader → weak bit $= 0$, B gets priority

A	0xBD	0x000?00				0xBD
B	0xBD	0x000000	★		0x000000	

After $+0$

A	0xBD	0x000000				0xBD
B	0xBD	0x000000	★		0x000000	

Counter is now fully reset!

Demo

CVE-2021-33881: affected products

- In MIFARE Ultralight family:
  - MIFARE Ultralight EV1, MF0UL;
  - MIFARE Ultralight C, MF0ICU;
  - MIFARE Ultralight NANO, MF0UN.
- In NTAG 21x family:
  - NTAG 210(μ)/212: NT2L1, NT2H10, NT2H12;
  - NTAG 213 (TT/F) /215 /216 (F): N2H13, NT2H15, NT2H16.

OTP & Lock bits security features potentially impacted too

Mitigations: see updated NXP *Application Note* AN11340 & new AN13089

Your turn!

- Large palette of EEPROM tearing effects
- Find other interesting targets
- EEPROM not only in RFID...
- You've now opensource tools
- You've now a paper: https://tinyurl.com/tearoff

Thank you

Let’s keep in touch

Discord:	https://discord.gg/iceman
Chris:	@herrmann1001
Phil:	@doegox