Open sesame - or how secure is your stuff in electronic lockers

NULLCON Berlin 2024 – Dennis Giese

About me

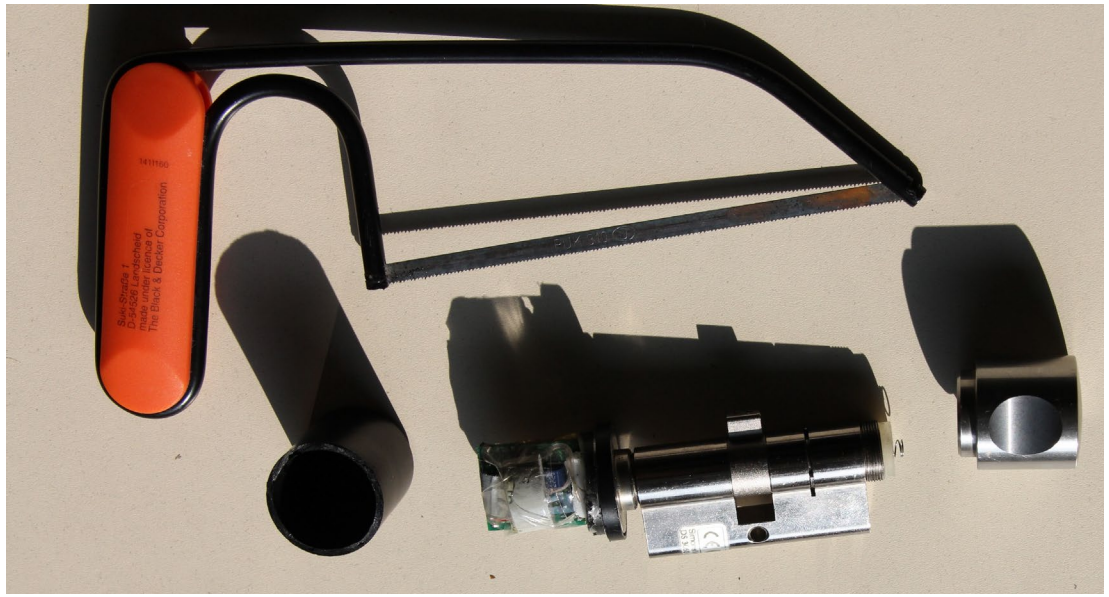
- “Security Researcher” aka Hardware Hacker
  - Research field: Wireless and embedded Security & Privacy
- Vacuum Robot (and IoT) collector
  - All brands: iRobot, Roborock, Dreame, Xiaomi, Shark, Narwal, Ecovacs, …
- Interests: Reverse engineering of interesting devices
  - Current research: Robots, Smart Speakers, Flash memory

Vacuum Robot hacking

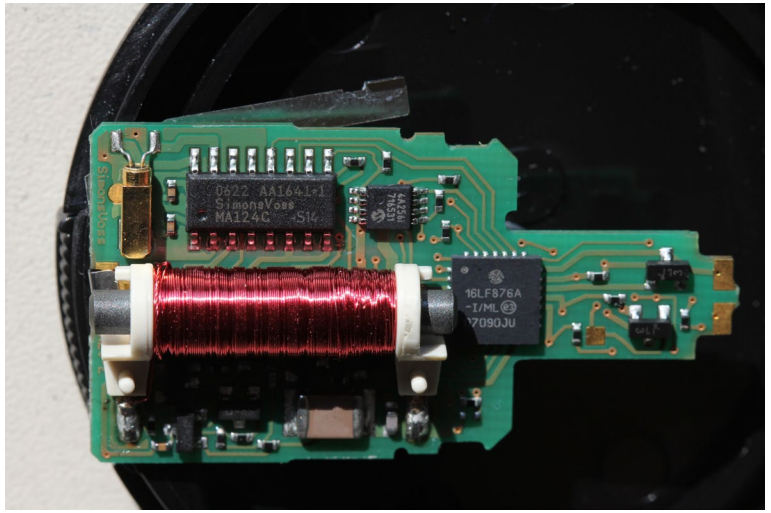
- Rooting of robots
- Analysis of Security&Privacy
- Using sensors
- Robotinfo.dev for documentation

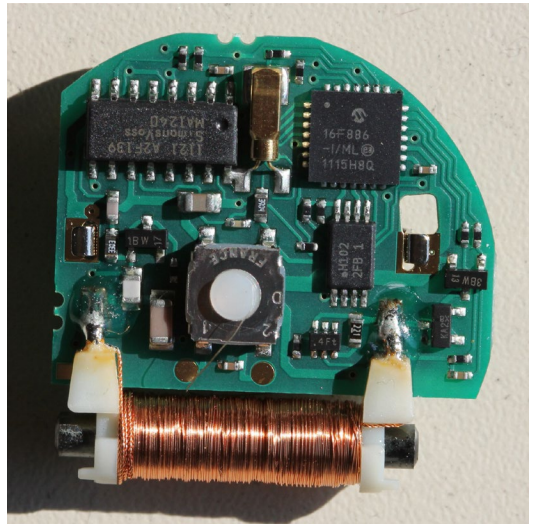
Previous work on locks

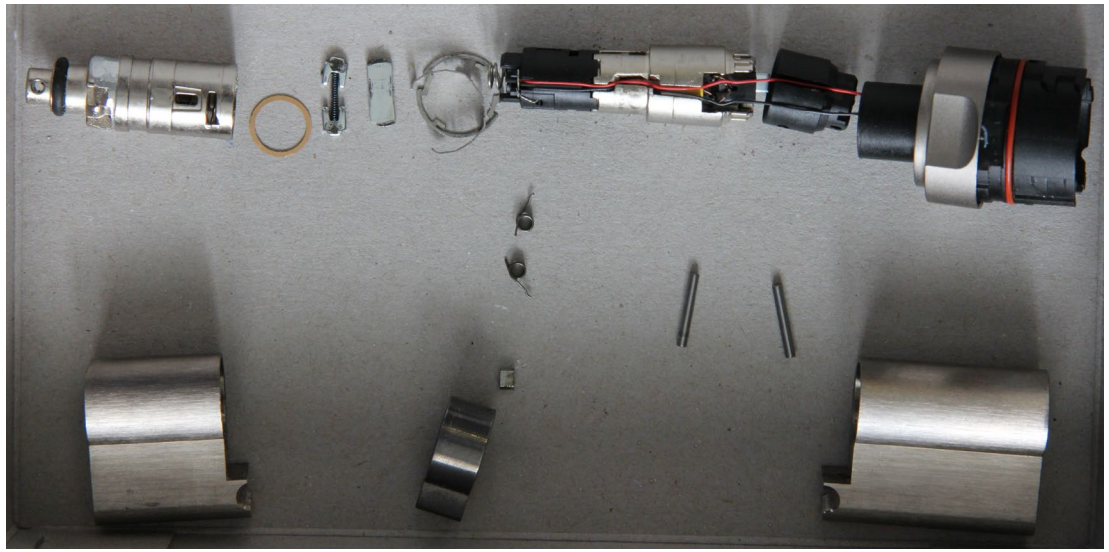
- Simons & Voss locks (2010-2013)
  - Published at ACM CCS 2013



Michael Weiner, Maurice Massar, Erik Tews, Dennis Giese, and Wolfgang Wieser. 2013. Security analysis of a widely deployed locking system. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13). Association for Computing Machinery, New York, NY, USA, 929–940. https://doi.org/10.1145/2508859.2516733



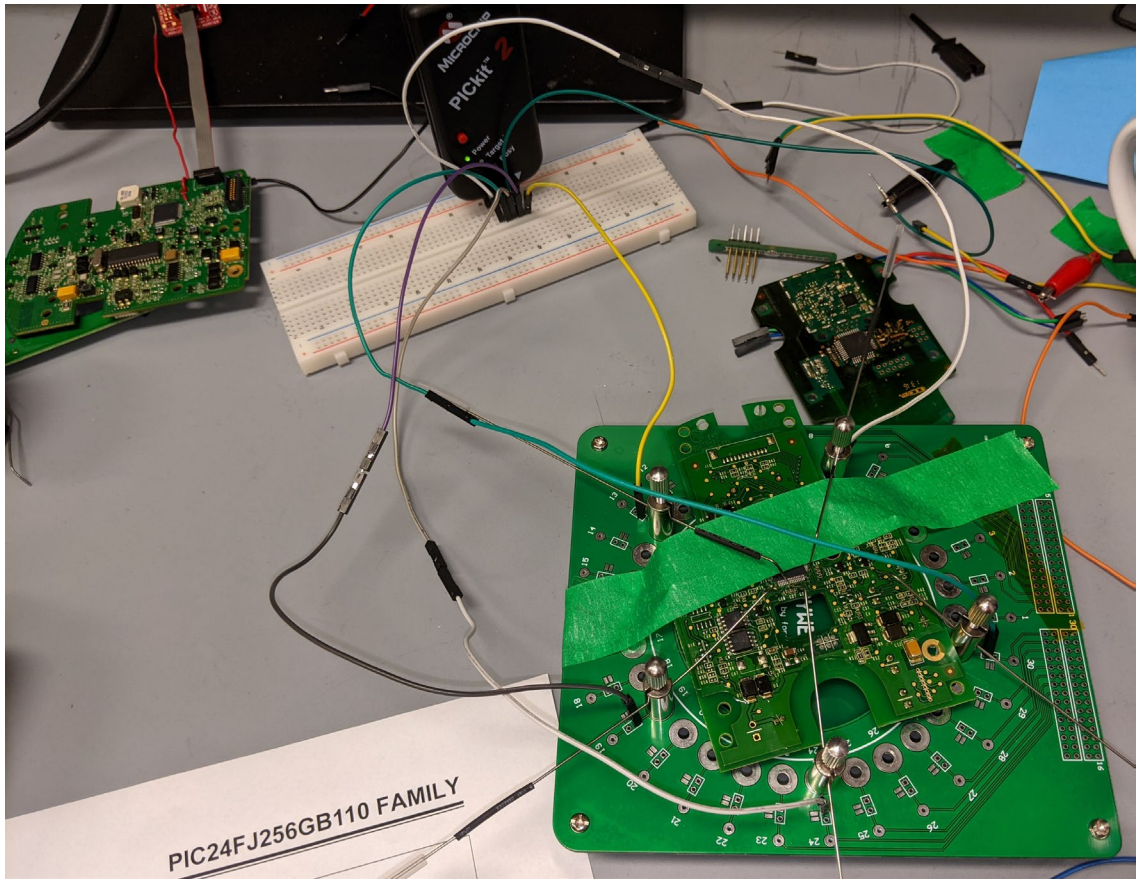




https://dontvacuum.me/papers/ACMCCS-2013/

Previous work on locks

- Schlage AD-400/401 Electronic Locks (2017/2018)



Goals of this talk

- Get an overview of the reverse-engineering of “Digilock” locks
- Learn about vulnerabilities
- Understand methods to extract firmware and config
- Raise awareness about PIN numbers
- Sidenote:
  - We use Digilock as an example and are not claiming that they are more/less secure than other companies
  - We chose Digilock due to the good reputation and quality of their products
  - The company is actively working on fixing issues

About this talk

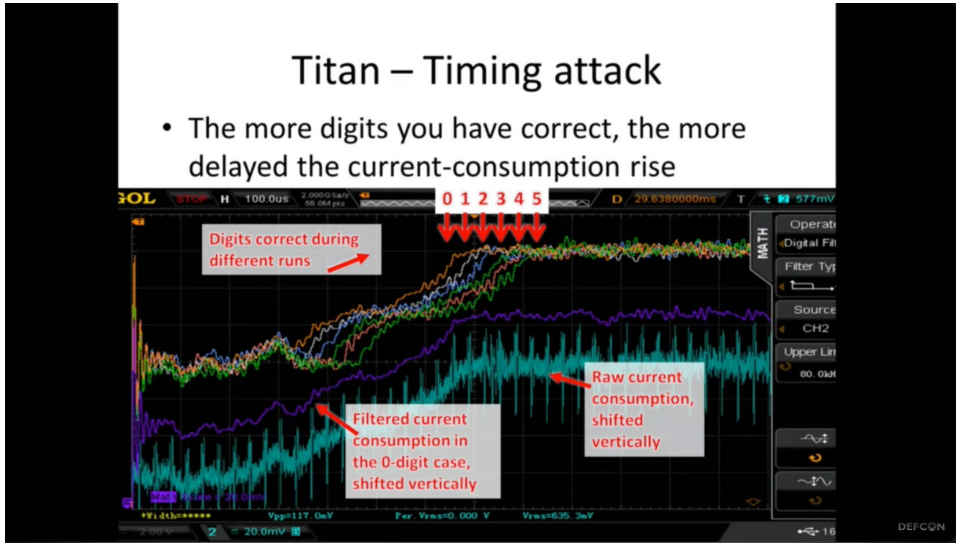
- Focus on one-wire based, offline locks
- Does not cover
  - last-gen locks
  - details about audit logs and management software
  - details about re-provisioning
  - destructive attacks (drilling, decapping, etc.)

MOTIVATION

Motivation

- Hacking electronic locks is not new
- Researchers focus on high security safe locks
  - Lots of research in side-channel
  - Safes contain expensive things
  - Big impact if insecure

Problem: It is hard to defend against physical attacks and motivated attackers



https://www.reuters.com/article/us-locks-cyber-exclusive/exclusive-high-security-locks-for-government-and-banks-hacked-by-researcher-idUSKCN1UW26Z/
https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEF%20CON%2024%20-%20Plore-Side-Channel-Attacks-On-High-Security-Electronic-Safe-Locks.pdf
https://www.youtube.com/watch?v=lXFpCV646E0

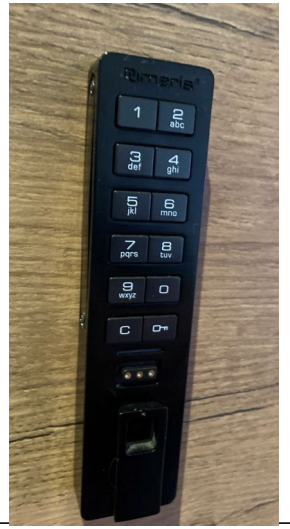
Motivation

- Consumer locks, safes and cabinets are known to be bad
  - Mechanical flaws
  - Trivial bypasses
  - Insecure software

LockPickingLawyer: [1571] The DUMBEST “Safe” Design I’ve Ever Seen! (Toriexon)
https://www.youtube.com/watch?v=gJrSWXFXvlE

Motivation

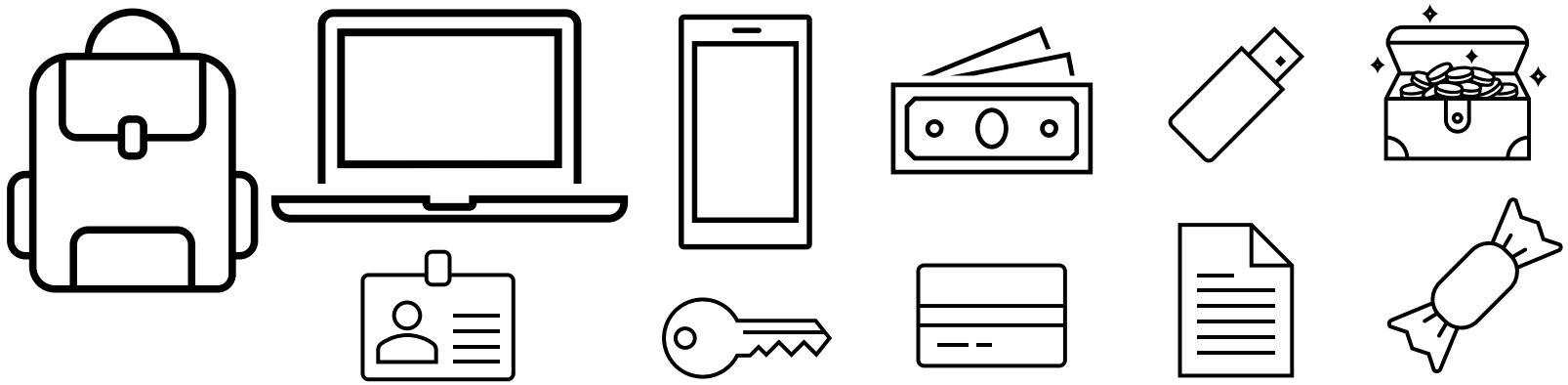
- Northeastern University (~2018)
  - Lockers introduced to labs
  - Use-case:
    - Put stuff in locker after work
    - Lock locker with PIN
- Seen in many co-working spaces
- Hotels



Top: Lockers in NEU EXP building
Bottom: Lockers in lobby of Moxy Hotel

Why hack lockers/cabinets?

- Lockers and cabinets are everywhere
- Used in public spaces or shared workspaces
- Forgotten PIN, lost keys, Red Team penetration tests
- ~~Tamper with~~ correct audit logs
- Might contain interesting stuff (including our own)

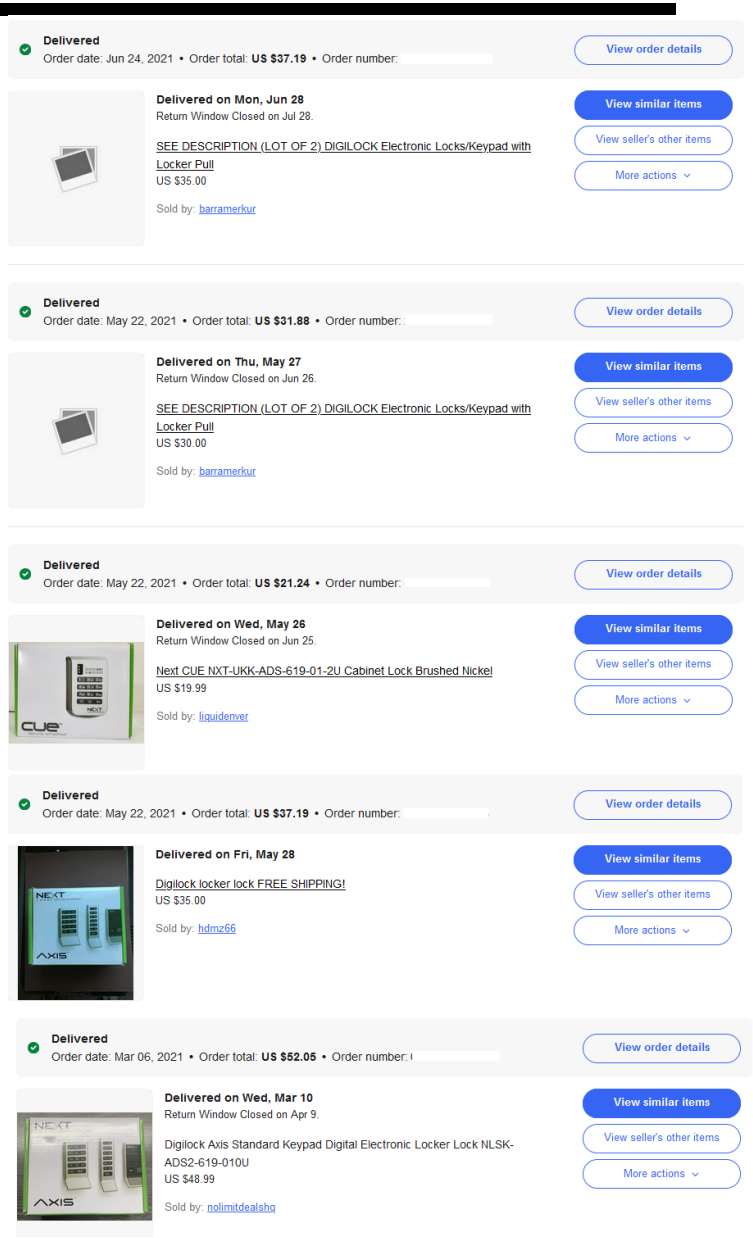


Is the same PIN used for the locker as for the Phone, Notebook or Credit card?

Quick survey:
Who knows “a friend” that is using a PIN in multiple places (e.g. Phone, Lockers, ATM cards)?

Procurement

- Experiments require multiple devices
  - Cannot use someone else's property ☹
- Locks and Keys are expensive
  - Locks ~USD $100
  - Keys ~USD $50-100
- COVID helped
  - Many gyms closed due to the pandemic
  - Surplus locks on eBay



THE DIGILOCK ECOSYSTEM

Digilock

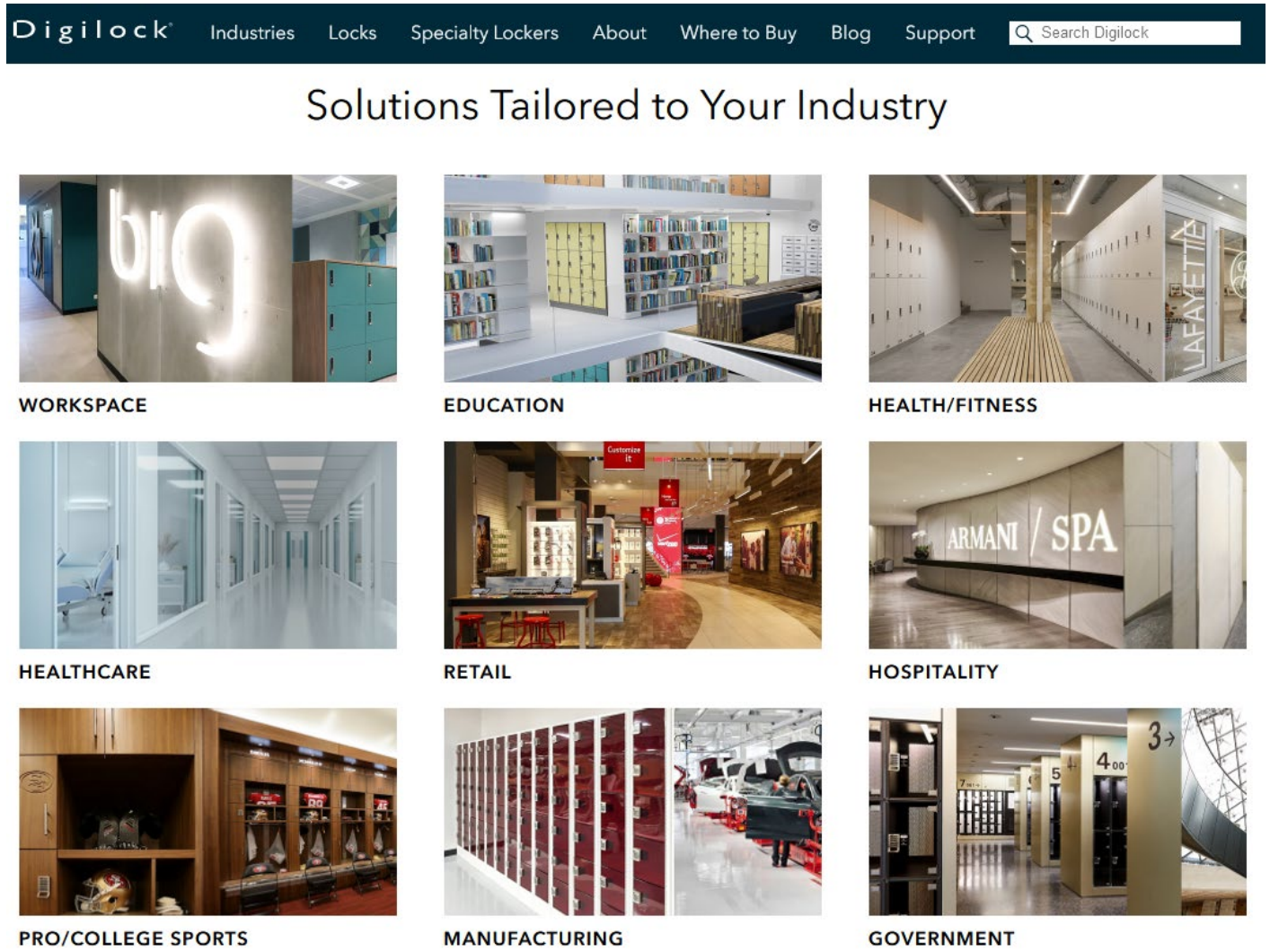
- dba of Security People Inc.
- US-based company
- Existed for more than 40 years
- Claims to be the “global leader in keyless lock solutions”
- Many different types and brands of locks
  - Connected locks, offline locks, mechanical locks
  - Access medium: RFID, PIN, Keys, Smartphone (BLE)
  - Brands (examples): “Digilock”, “NEXT”, “Numeris”

Source(March 2024): https://digilockeurope.com/en/

Examples


1 2 abc 3 def
4 ghi 5 jkl 6 mno
7 pqrs 8 tuv 9 wxyz
C 0
NEXT
1 2 abc 3 def
4 ghi 5 jkl 6 mno
7 pqrs 8 tuv 9 wxyz
C 0
NEXT
1 2 abc
3 def 4 ghi
5 jkl 6 mno
7 pqrs 8 tuv
9 wxyz 0
C
1 2 abc 3 def
4 ghi 5 jkl 6 mno
7 pqrs 8 tuv 9 wxyz
C 0
Digilock

Industries



Source/Screenshot(March 2024): https://www.digilock.com/

Digilock example customers

Electronic locks around the world

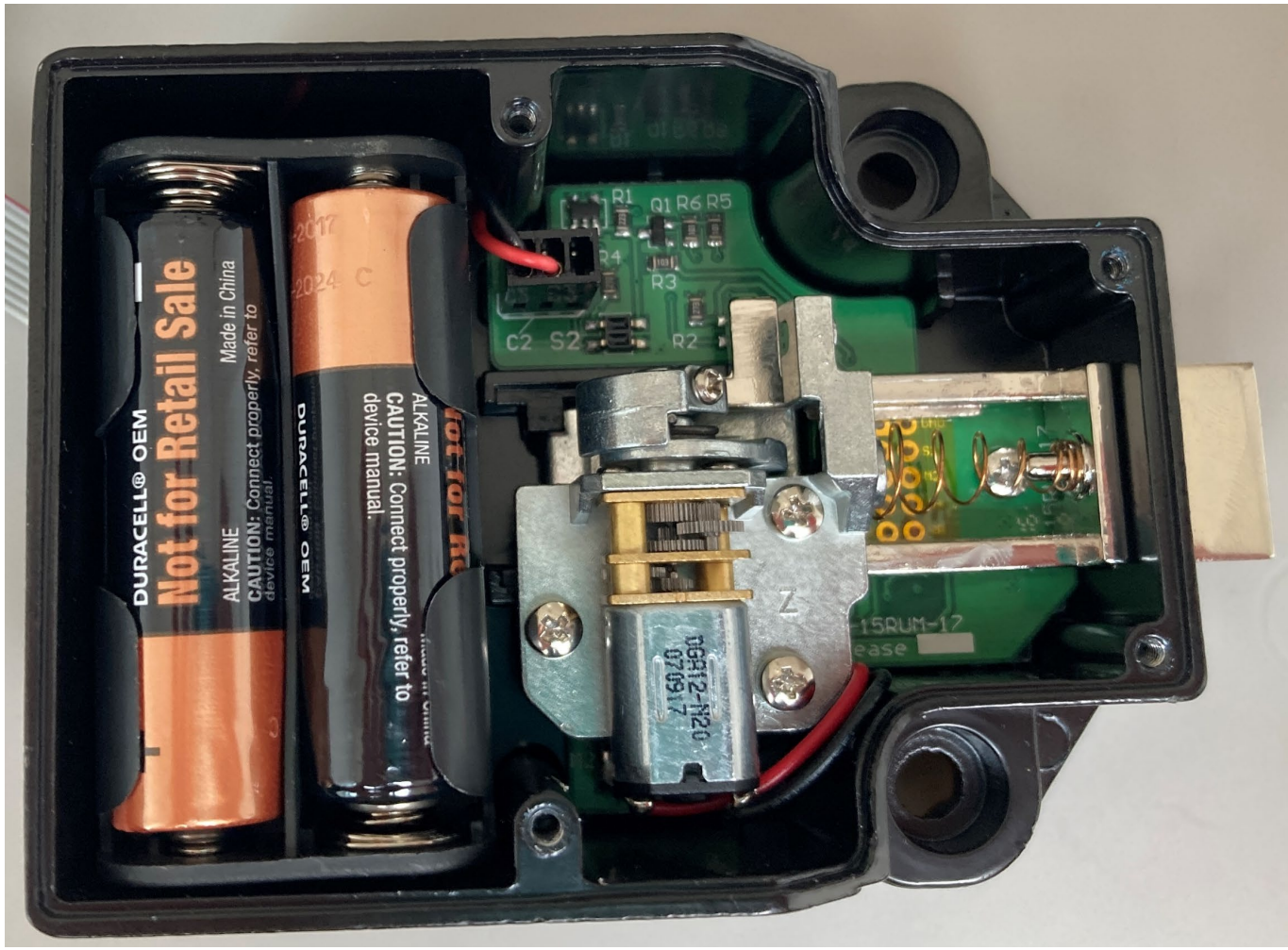
Countless users use Digilock locks.

View less customers

Source/Screenshot(March 2024): https://digilockeurope.com/en/

Lock Hardware

Digilock 4G (outside part) with latch unit (inside part)



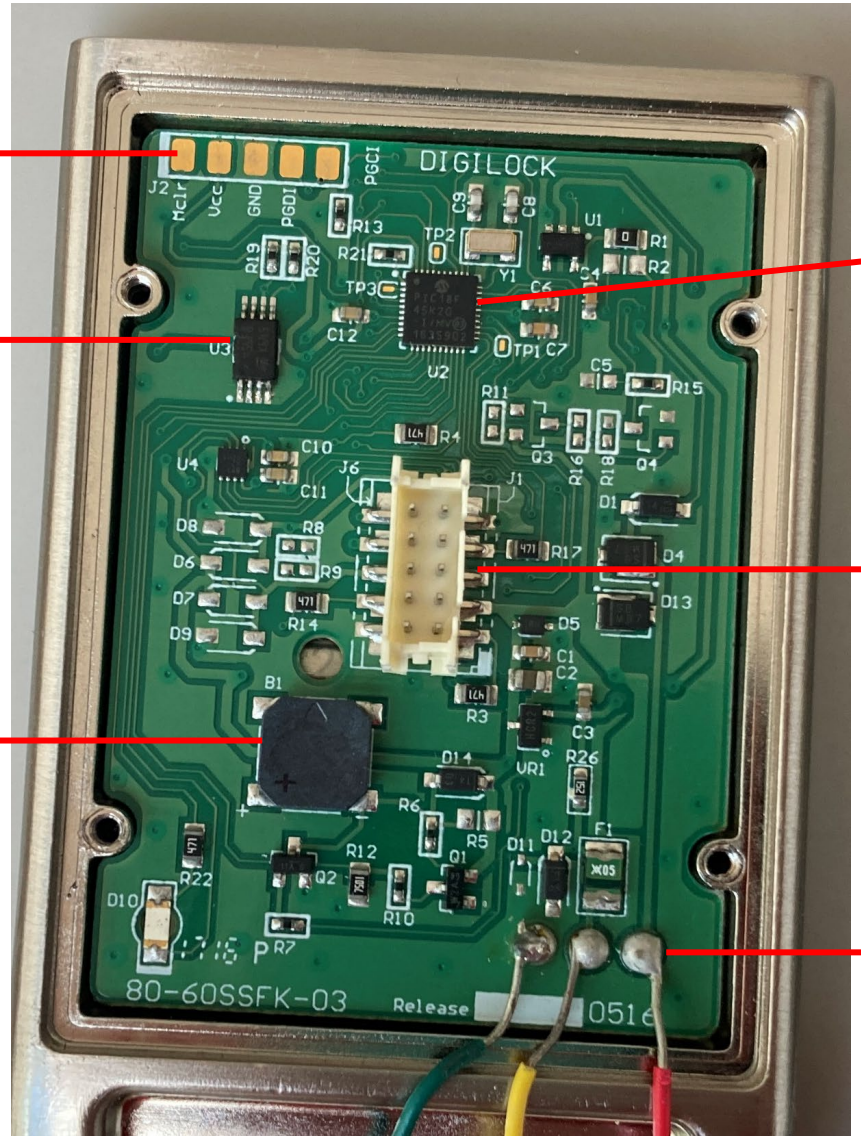
Opened latch unit

Lock Hardware

- All locks have similar hardware
  - Same type of MCU
  - Some support audit
  - No RTC
  - No tamper switches
  - Same interface to latch
- Locking state controlled by latch
- Good protection against physical attacks
- Software features different per brand

Different brands of Digilock locks

Lock Hardware



Digilock NEXT lock backside of PCB

Lock Hardware

- MCU
  - PIC18F45K20
  - PIC18F25K20 (older gen)
  - PIC24FJ256GA705 (last gen)
- EEPROM (for audit)
  - ST M24256-BW 256-Kbit serial I^2C bus EEPROM
- RFID
  - ST ST25R3911B
  - LEGIC SM-6300 (+HSM+BLE)

Keys

- Programming Key (yellow)
  - Only one exist per “locking system”
  - Adds/removes manager keys
  - Allows lock override
  - Power for a dead lock
  - Cloning configuration/audits/etc.
- Manager Key (black)
  - Multiple per “locking system”
  - Allows lock override
  - Power for a dead lock

Digilock keys: Programming, Manager, ADA, Reset

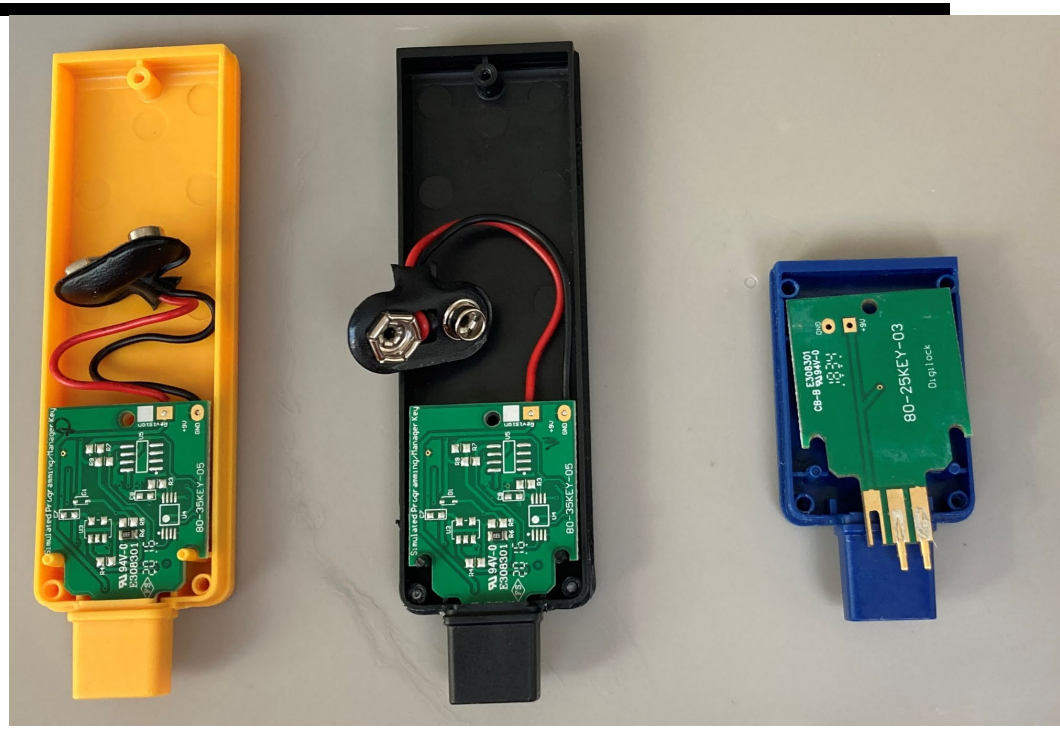
Keys

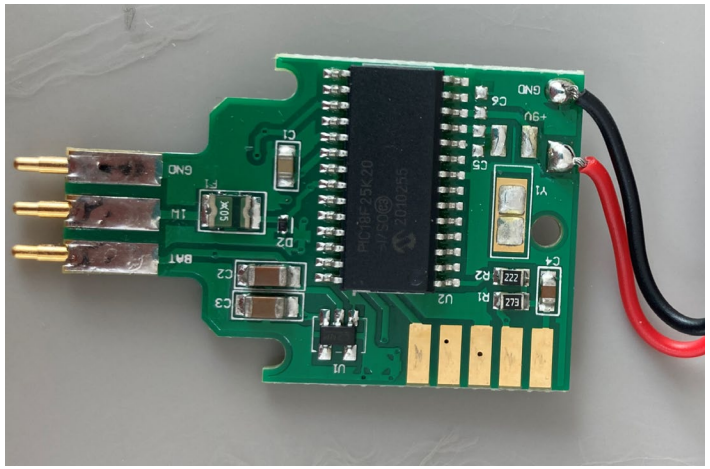
- ADA Key (blue)
  - Alternative to PIN / RFID
- Replacement programming keys (red)
  - Used to reset if programming key is lost

Digilock keys: Programming, Manager, ADA, Reset

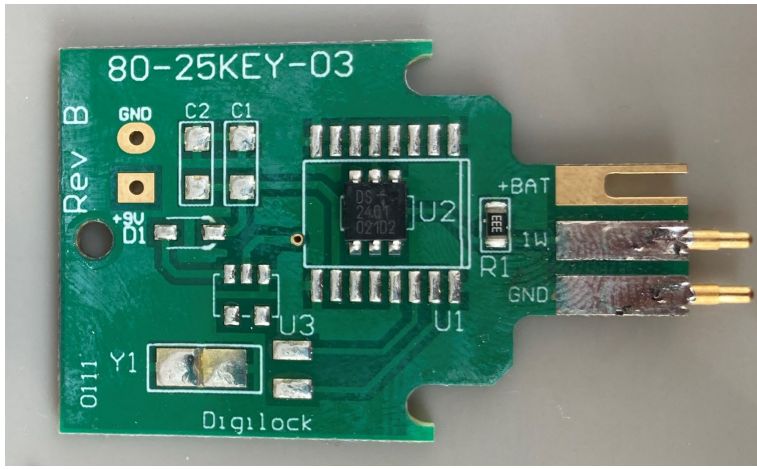
Keys

- Programming/Manager key
  - Same hardware, different case color
  - 9V battery
  - PIC18F25K20 (older gen)
  - PIC programming interface
- ADA key
  - DS2401 aka iButton
  - 48 bit ID





Programming key

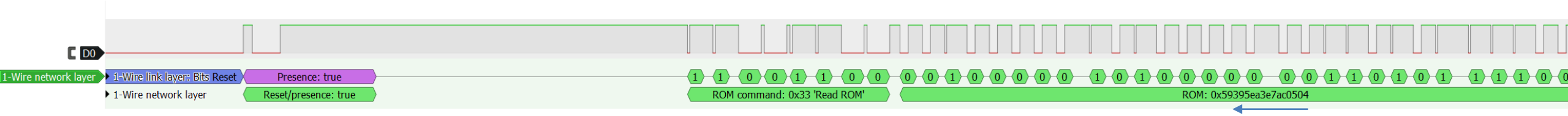


ADA key

One Wire communication

- Easy to intercept with logic analyzer
- Usage of “Read ROM” command
- Keys return 8 byte of ID (7 Data + 1 CRC8)
- Key types identified by first byte of ID
  - 0x01: ADA key
  - 0x04: Programming key
  - 0x14: Management key
- Bus resets after transaction

No crypto, challenge-response or anything else required



Trace of programming key authentication

PIC MCUS

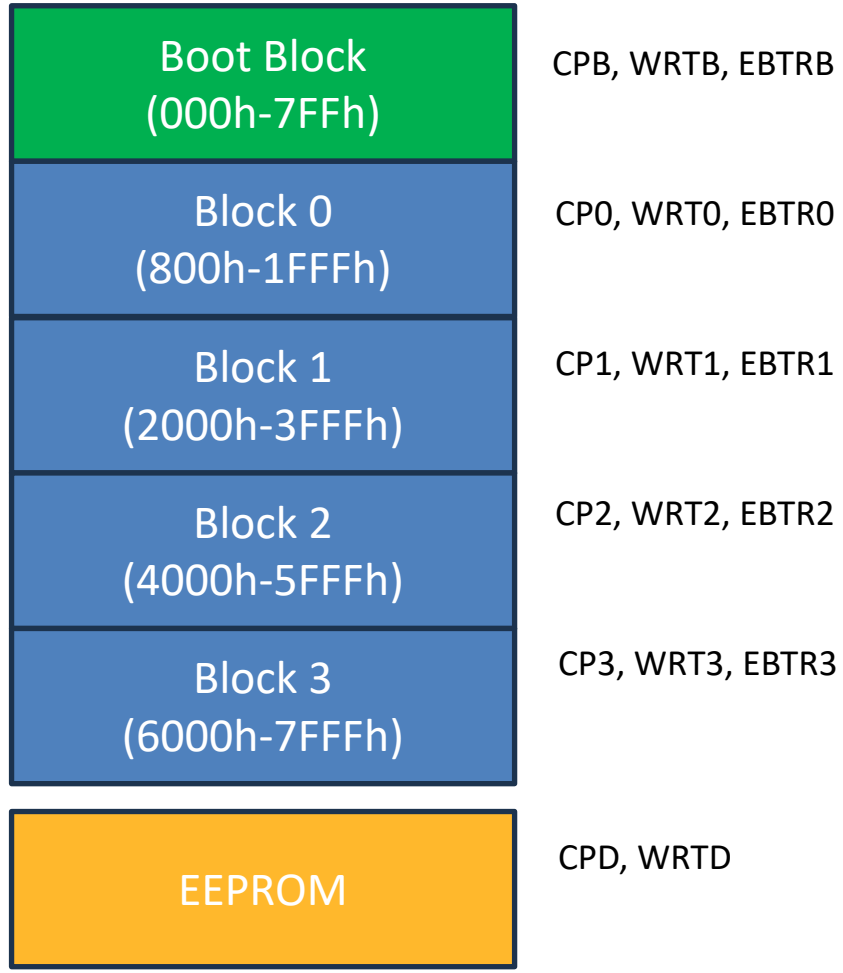
PIC intro

- MCU by Microchip
- Very common for locks
  - Used by Simons&Voss, Schlage, Aqara, etc.
  - Low power, ideal for battery operation
- PIC18
  - 8 Bit MCU, released 2000
  - Config stored in flash after programming
    - Debugging
    - Brownout protection
    - Protections per flash block

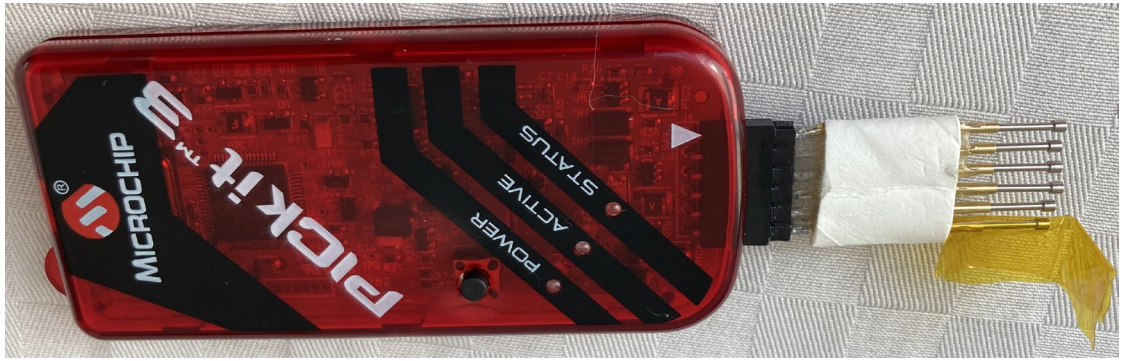
https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors

PIC18F25K20 / PIC18F45K20

- 1536 Bytes SRAM
- 32 KBytes Flash
- 256Bytes EEPROM
- Protections
  - Code Protection (CP)
  - Write Protection (WRT)
  - External Block Table Read (EBTRB)
  - bit 1=default off, bit 0=enabled

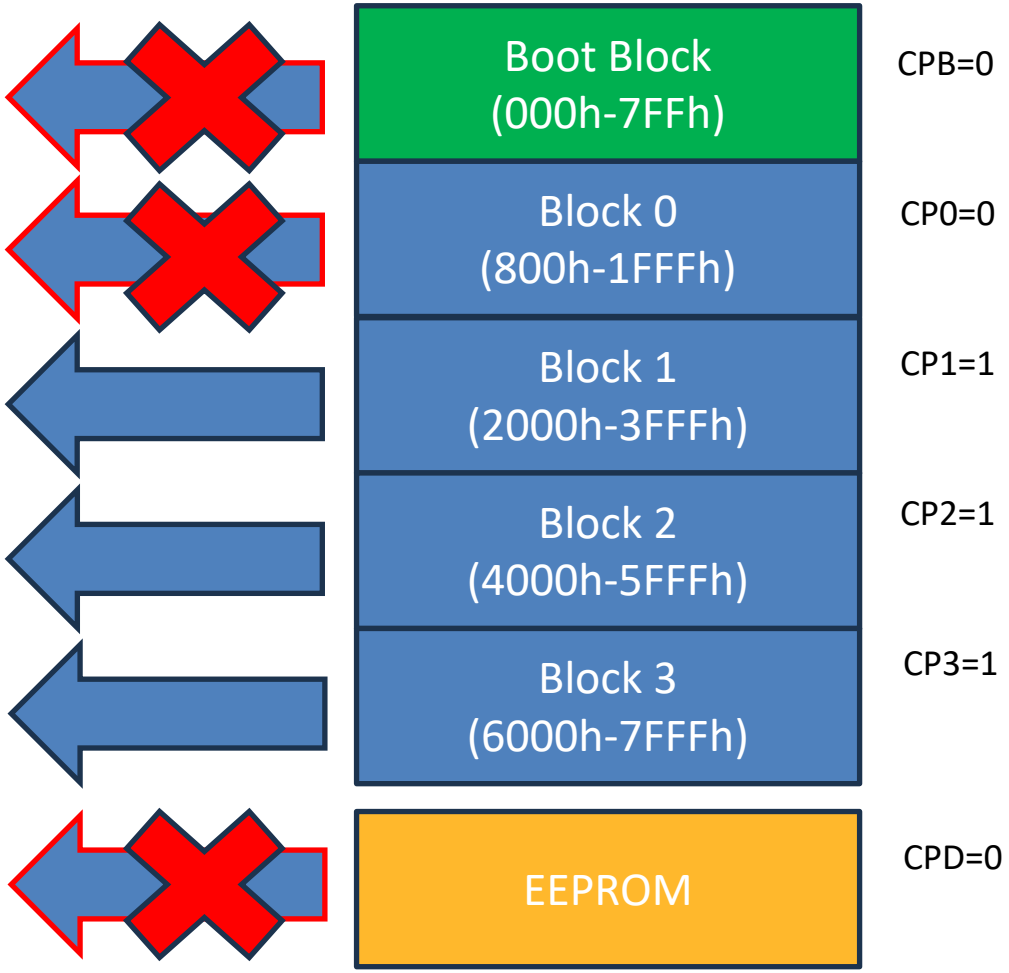


Code Protection example

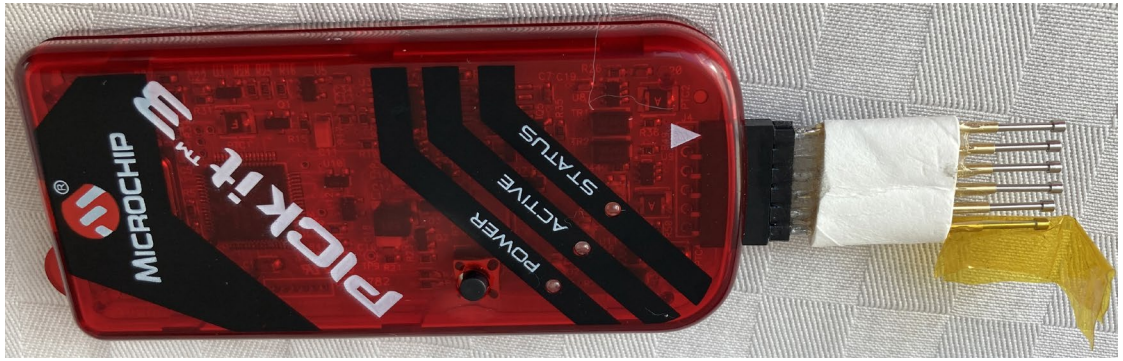


Microchip Pickit debugger

- Blocks Boot, 0 and Data return 0’s
- Blocks 1,2,3 return data

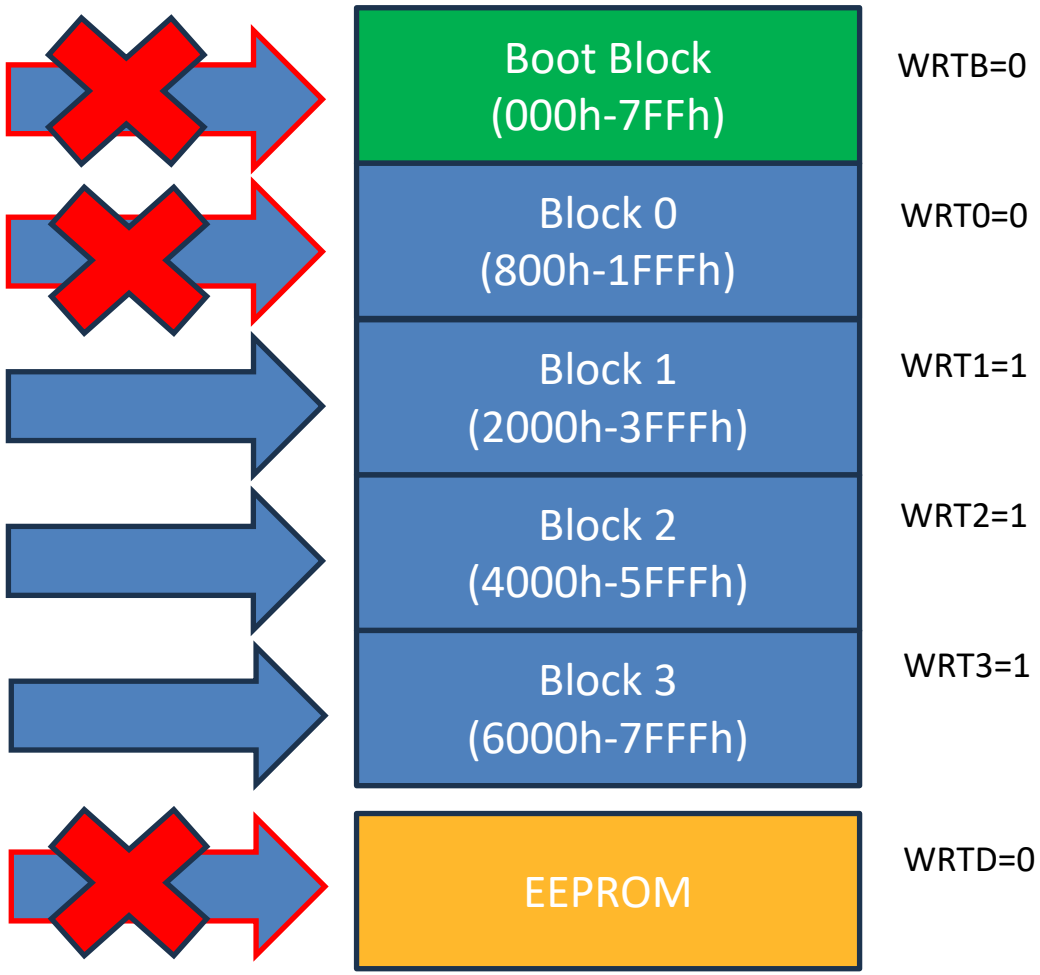


Write Protection example



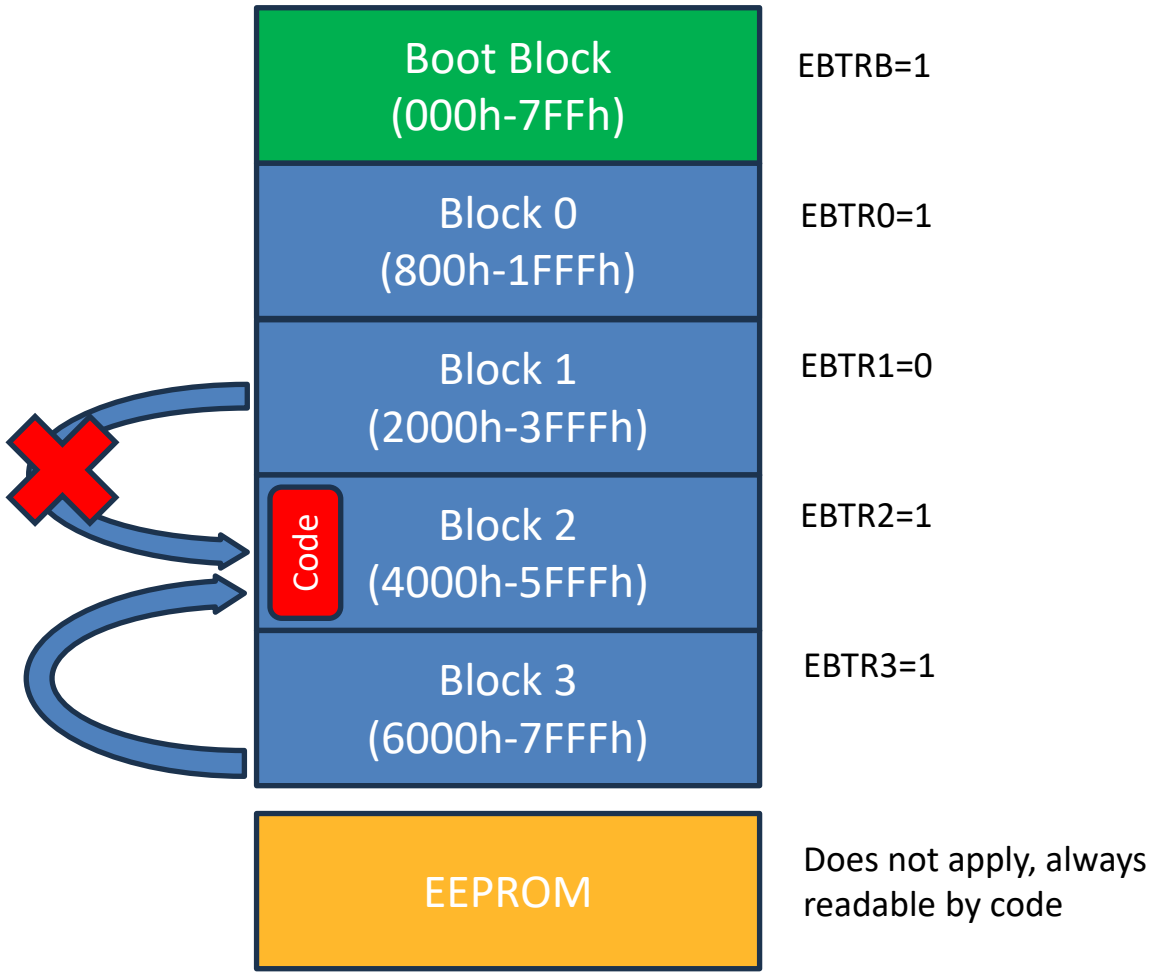
Microchip Pickit debugger

- Blocks Boot, 0 and Data fail writes
- Blocks 1,2,3 are programmed



External Block Table Read example

- Code in Block 2 can read Block 3
- Code in Block any block cannot read Block 1



PIC Security

- MCUs offer only basic protection against attacks
- Many attacks exist (even if protections are enabled)
  - Optical/Laser attacks in 2002
  - UV erasure of config bits
  - Glitching
  - Overwriting individual blocks to dump other blocks

Examples:
Skorobogatov, Sergei & Anderson, Ross. (2002). Optical Fault Induction Attacks. Optical Fault Induction Attacks. 2523. 2-12. 10.1007/3-540-36400-5_2.
https://www.bunniestudios.com/blog/?page_id=40
http://blog.lanka.sk/2013/11/hacking-apc-back-ups-hs-500.html

ATTACKS

What are we looking for?

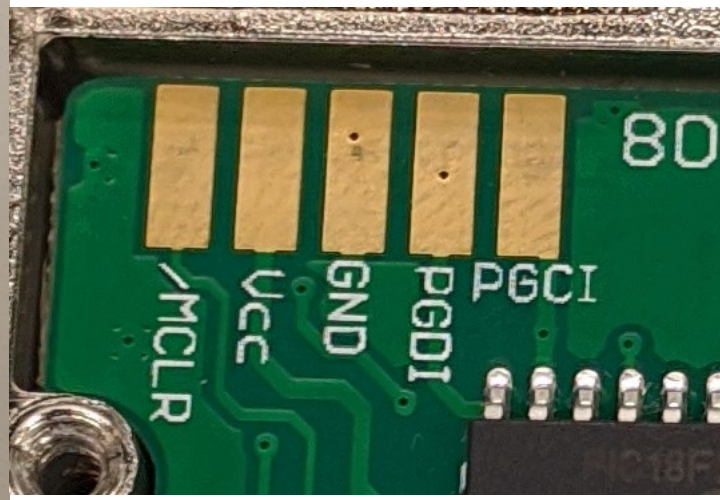
- Firmware
  - Find secret backdoors or bugs
  - Understand functionality
  - Create malicious firmware
- Interesting data
  - Key IDs, user PINs, RFID IDs, logs
- Ways to easily open locks

Attack ideas

- Dumping flash and EEPROM
- Bruteforce PINs and keys
- Sidechannel attacks
- Clone Keys

Dumping flash and EEPROM

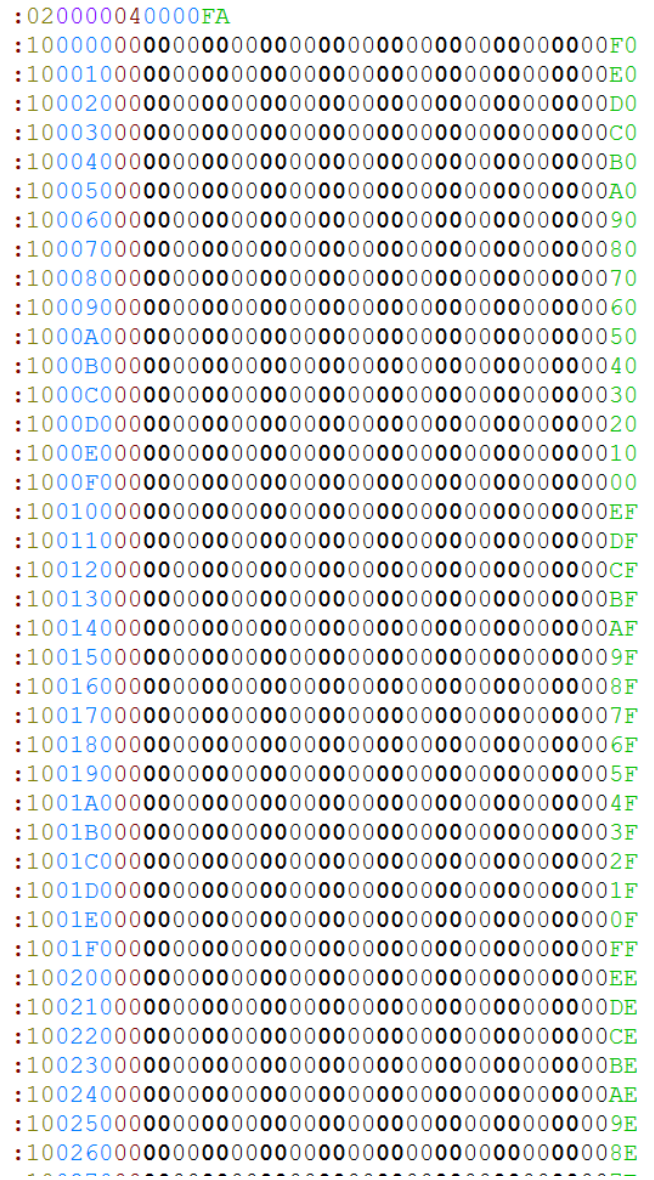
- Naive approach: connect debugger and dump flash
  - Debug pins were exposed on all locks and keys



Dumping flash and EEPROM

- Keys
  - Program memory was protected
  - EEPROM was empty (but not protected)
- Locks
  - Program memory was *mostly* protected
  - EEPROM contained data

Same result for all 12 locks and 6 keys (no matter generation or brand)

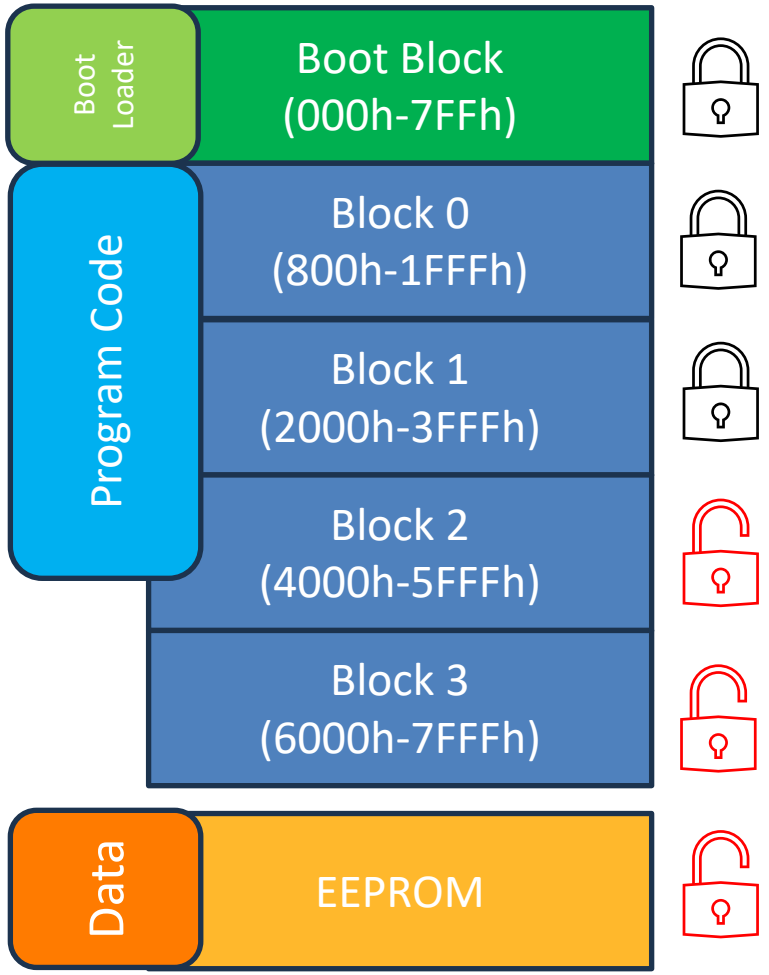


Dumping flash and EEPROM

Address	Name	Value	Field	Option	Category	Setting
300008	CONFIG5L	0C	-	-	-	-
		0	CP0	ON	Code Protection Block 0	Block 0 (000800-001FFFh) code-protected
		0	CP1	ON	Code Protection Block 1	Block 1 (002000-003FFFh) code-protected
		1	CP2	OFF	Code Protection Block 2	Block 2 (004000-005FFFh) not code-protected
		1	CP3	OFF	Code Protection Block 3	Block 3 (006000-007FFFh) not code-protected
300009	CONFIG5H	80	-	-	-	-
		0	CPB	ON	Boot Block Code Protection bit	Boot block (000000-0007FFh) code-protected
		1	CPD	OFF	Data EEPROM Code Protection bit	Data EEPROM not code-protected
30000A	CONFIG6L	0F	-	-	-	-
		1	WRT0	OFF	Write Protection Block 0	Block 0 (000800-001FFFh) not write-protected
		1	WRT1	OFF	Write Protection Block 1	Block 1 (002000-003FFFh) not write-protected
		1	WRT2	OFF	Write Protection Block 2	Block 2 (004000-005FFFh) not write-protected
		1	WRT3	OFF	Write Protection Block 3	Block 3 (006000-007FFFh) not write-protected
30000B	CONFIG6H	E0	-	-	-	-
		1	WRTC	OFF	Configuration Register Write Protection bit	Configuration registers (300000-3000FFh) not write-protected
		1	WRTB	OFF	Boot Block Write Protection bit	Boot Block (000000-0007FFh) not write-protected
		1	WRTD	OFF	Data EEPROM Write Protection bit	Data EEPROM not write-protected
30000C	CONFIG7L	0F	-	-	-	-
		1	EBTR0	OFF	Table Read Protection Block 0	Block 0 (000800-001FFFh) not protected from table reads executed in other blocks
		1	EBTR1	OFF	Table Read Protection Block 1	Block 1 (002000-003FFFh) not protected from table reads executed in other blocks
		1	EBTR2	OFF	Table Read Protection Block 2	Block 2 (004000-005FFFh) not protected from table reads executed in other blocks
		1	EBTR3	OFF	Table Read Protection Block 3	Block 3 (006000-007FFFh) not protected from table reads executed in other blocks
30000D	CONFIG7H	40	EBTRB	OFF	Boot Block Table Read Protection bit	Boot Block (000000-0007FFh) not protected from table reads executed in other blocks

Dumping flash and EEPROM

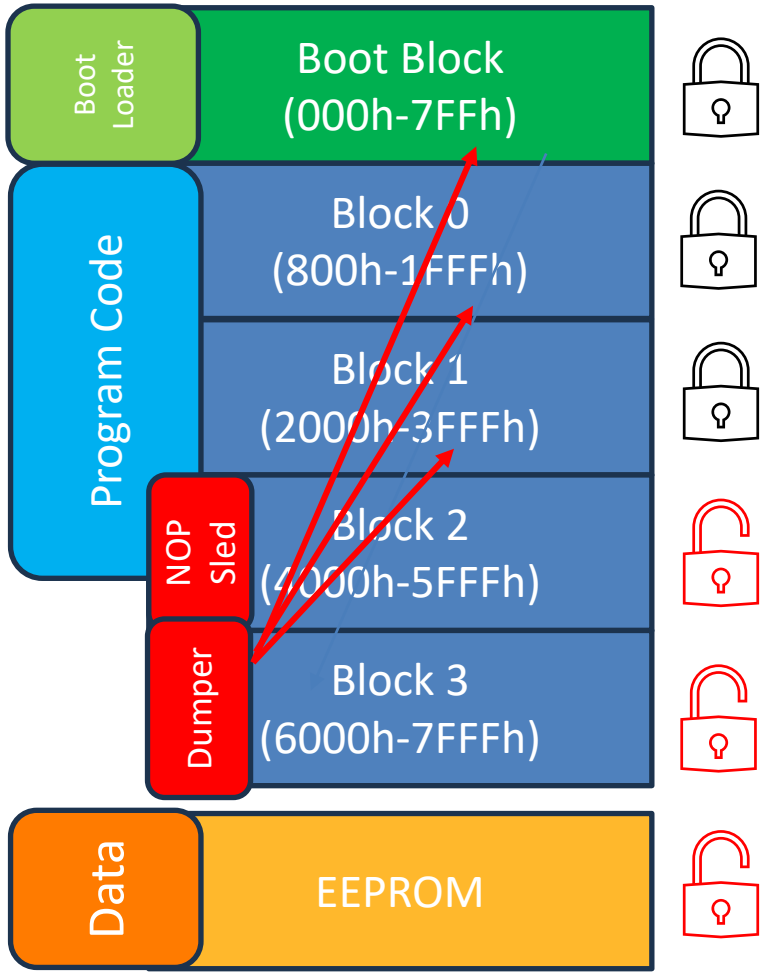
Field	Option	Category	Setting
-	-	-	-
CP0	ON	Code Protection Block 0	Block 0 (000800-001FFFh) code-protected
CP1	ON	Code Protection Block 1	Block 1 (002000-003FFFh) code-protected
CP2	OFF	Code Protection Block 2	Block 2 (004000-005FFFh) not code-protected
CP3	OFF	Code Protection Block 3	Block 3 (006000-007FFFh) not code-protected
-	-	-	-
CPB	ON	Boot Block Code Protection bit	Boot block (000000-0007FFh) code-protected
CPD	OFF	Data EEPROM Code Protection bit	Data EEPROM not code-protected
-	-	-	-
WRT0	OFF	Write Protection Block 0	Block 0 (000800-001FFFh) not write-protected
WRT1	OFF	Write Protection Block 1	Block 1 (002000-003FFFh) not write-protected
WRT2	OFF	Write Protection Block 2	Block 2 (004000-005FFFh) not write-protected
WRT3	OFF	Write Protection Block 3	Block 3 (006000-007FFFh) not write-protected
-	-	-	-
WRTC	OFF	Configuration Register Write Protection bit	Configuration registers (300000-3000FFh) not write-prot
WRTB	OFF	Boot Block Write Protection bit	Boot Block (000000-0007FFh) not write-protected
WRTD	OFF	Data EEPROM Write Protection bit	Data EEPROM not write-protected
-	-	-	-
EBTR0	OFF	Table Read Protection Block 0	Block 0 (000800-001FFFh) not protected from table reads
EBTR1	OFF	Table Read Protection Block 1	Block 1 (002000-003FFFh) not protected from table reads
EBTR2	OFF	Table Read Protection Block 2	Block 2 (004000-005FFFh) not protected from table reads
EBTR3	OFF	Table Read Protection Block 3	Block 3 (006000-007FFFh) not protected from table reads
EBTRB	OFF	Boot Block Table Read Protection bit	Boot Block (000000-0007FFh) not protected from table re



Code/Data on NEXT CUE lock

Dumping flash and EEPROM

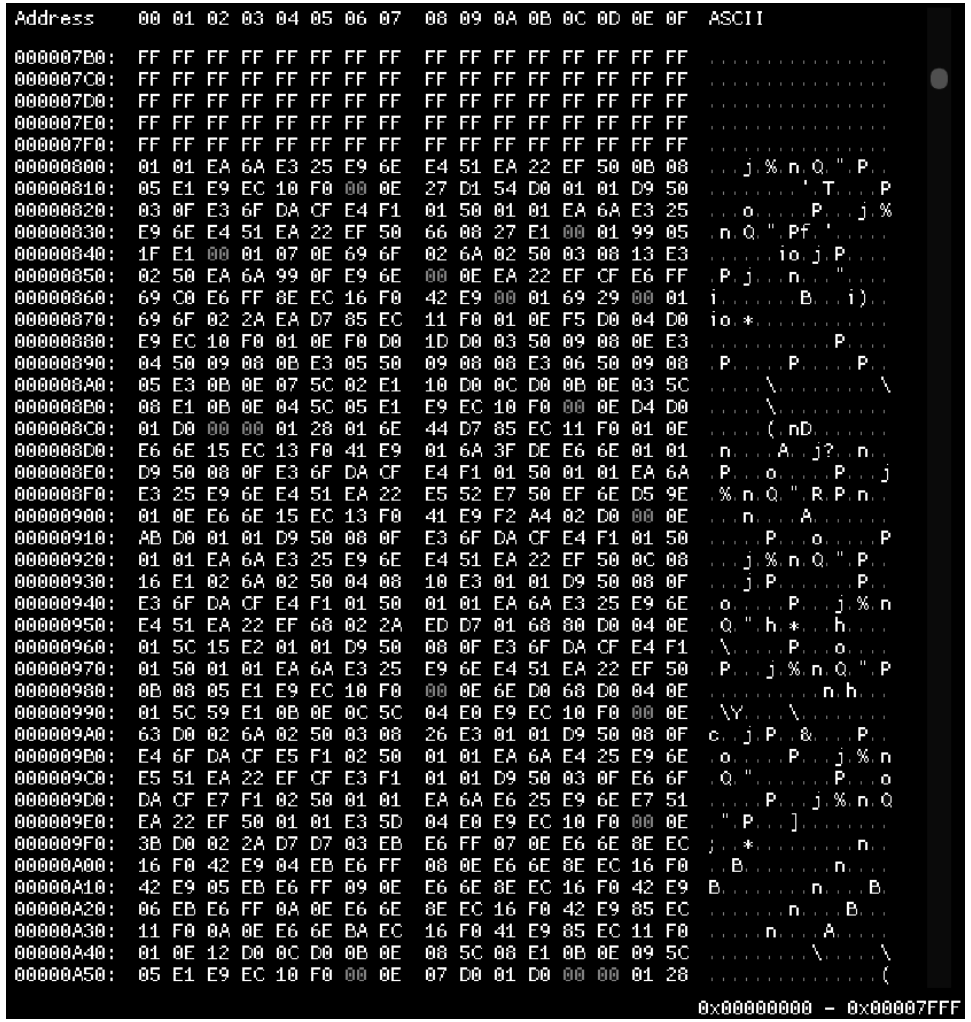
- Idea: tamper with Block 2
  - Backup Block 2
  - Overwrite Block 2 with instructions
  - Hope that instructions are called
  - Add code that dumps other blocks
  - Exfiltrate contents via UART
- Alternative for shorter code: Block 1
  - Overwrite code in Block 1 (destructive)
  - Extract Bootloader and Block 0



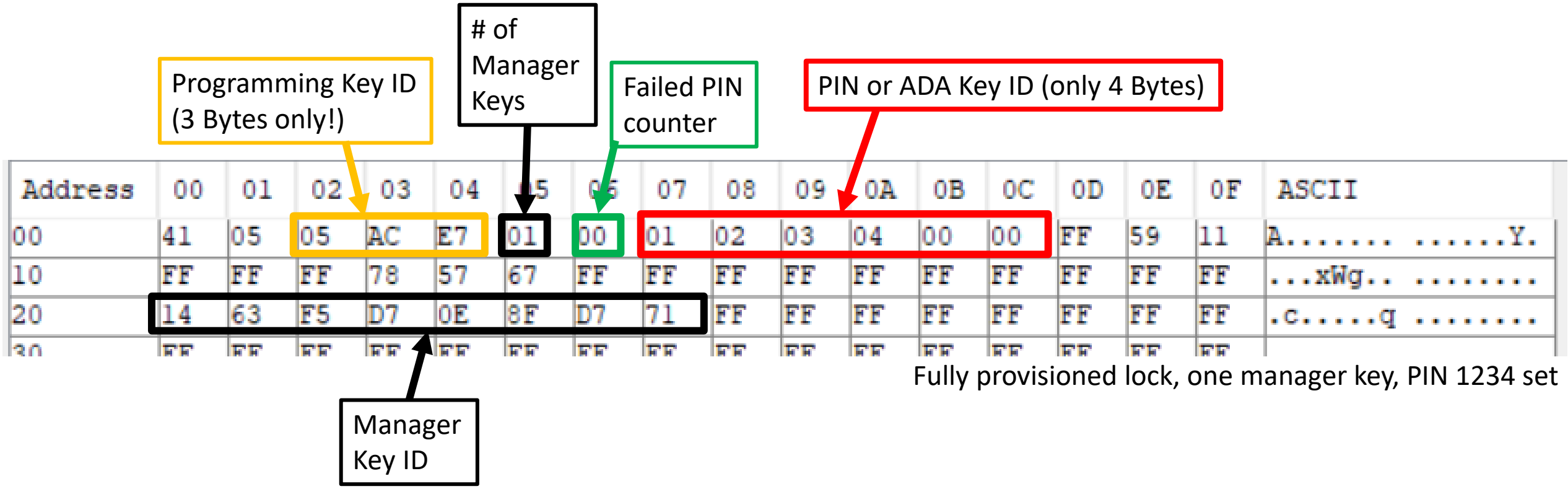
Code/Data on NEXT CUE lock

Dumping flash and EEPROM

- Extraction was successful
  - Using multiple keys to dump firmware
  - Non-destructive for big firmware
- Binary can be analyzed and modified
- Method well known and established



EEPROM contents



Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
00	41	05	05	AC	E7	01	00	01	02	03	04	00	00	FF	59	11	A..... .....Y.
10	FF	FF	FF	78	57	67	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	...xWg.. .....
20	14	63	F5	D7	0E	8F	D7	71	FF	FF	FF	FF	FF	FF	FF	FF	.c.....q .....
30	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	

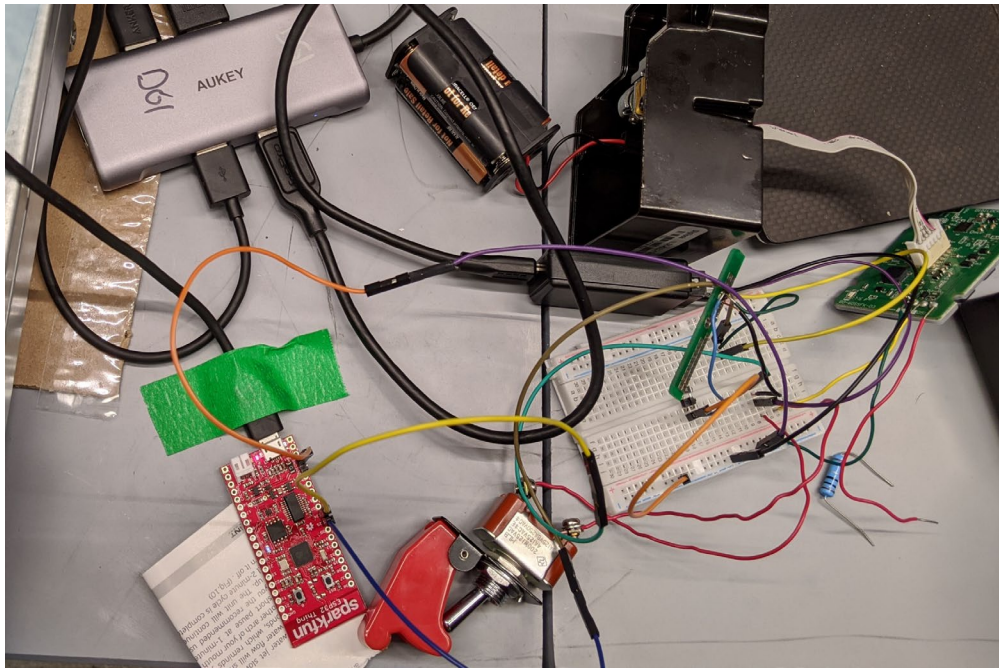
Fully provisioned lock, one manager key, PIN 1234 set

Bruteforce

- Problem: rate limiting of PIN entries
  - Lock blocks itself after 3 invalid entries for 1 minute
  - Can be partially bypassed by glitching power
    - (does not reset counter and bad things will happen if > 255)
- Rate limiting does not apply to OneWire interface
  - Limitation:
    - only specific first bytes are accepted
    - Needs to start with 0x01, 0x04, 0x14, etc.
  - Idea: Brute forcing Programming key (3Bytes)

Bruteforce

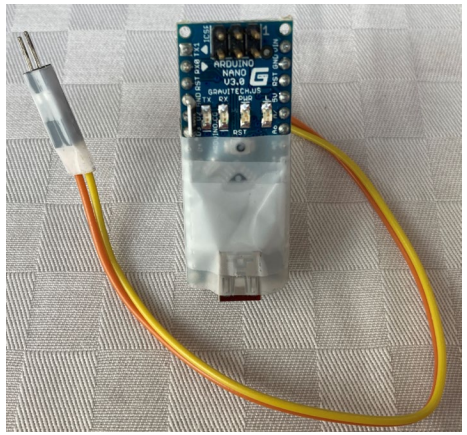
- Timing can be observed via OneWire
  - between command and bus reset
  - 260ms for correct ID/PIN
  - 235ms for incorrect ID/PIN
  - Takes too long
- Alternative: using ID comparison as side-channel
  - Timing differs depending on which byte is wrong
  - Significantly faster to bruteforce
  - Problem: Jitter



Example setup for brute forcing

Emulation/Cloning Keys

- Only ID is required for cloning
  - trivial to dump
- Can be emulated easily
  - Arduino
  - Flipper Zero




```
#include "OneWireHub.h"
#include "DS2401.h"

constexpr uint8_t pin_led      { 13 };
constexpr uint8_t pin_onewire   { 3 };

auto hub    = OneWireHub(pin_onewire);
// 0x01 (ADA) ,0x04 (PROG), 0x14 (MGMT)
auto ds2401 = DS2401( 0x14, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE,
0xFF );
void setup()
{
   hub.attach(ds2401);
}

void loop()
{
   // following function must be called periodically
   hub.poll();
}
```

Example code for emulating keys

Other attacks

- Cloning lock configuration via OneWire
  - Dumps manager key IDs from lock
- Modifying contents of audit EEPROM
- More power glitching
- Usage of malicious firmware

RESULTS AND CONCLUSION

Example Scenario

- Assumption: Attacker has access to *any* open locker/cabinet
- Required tools: Debugger, Philips Screwdriver, Arduino/Flipper
- Approach to get key:
  - Remove 2 screws to detach the lock
  - Remove 2-4 screws to open the lock and access the PCB
  - Attach debugger and dump flash
  - Create fake Manager or Programming key
- Required time: 1-2 min total

Example Scenario cont.

- Approach to break into lockers
  - Use programming / manager key to open target locker
  - Repeat dumping procedure to extract PIN
  - Tamper with stuff inside the locker
  - Use programming / manager key to close target locker
  - (the PIN remains the same if the special key is used)
- Result:
  - You have a special key for the whole system
  - You know the pin that was used to lock the target locker

Take away lessons

- Do not re-use an important PIN for lockers/cabinets/safes
- Never borrow someone electronic keys
- Be aware about the security limitations of these devices
- Do not trust audit logs of devices
- Even experienced and big companies make mistakes
- Producing a high-security but cheap system is difficult
- There might be interesting cyber-physical systems around you

Do not forget the human factor! Just ask nicely for the key?

Summary

- We can extract firmware and keys from Digilock locks
  - We see the undocumented methods
  - Attacks might not apply to last gen devices
- Access to one lock can give you access to all (in one location)
- Clone and emulate keys is easy
- Attacks do not require complicated tools and are cheap

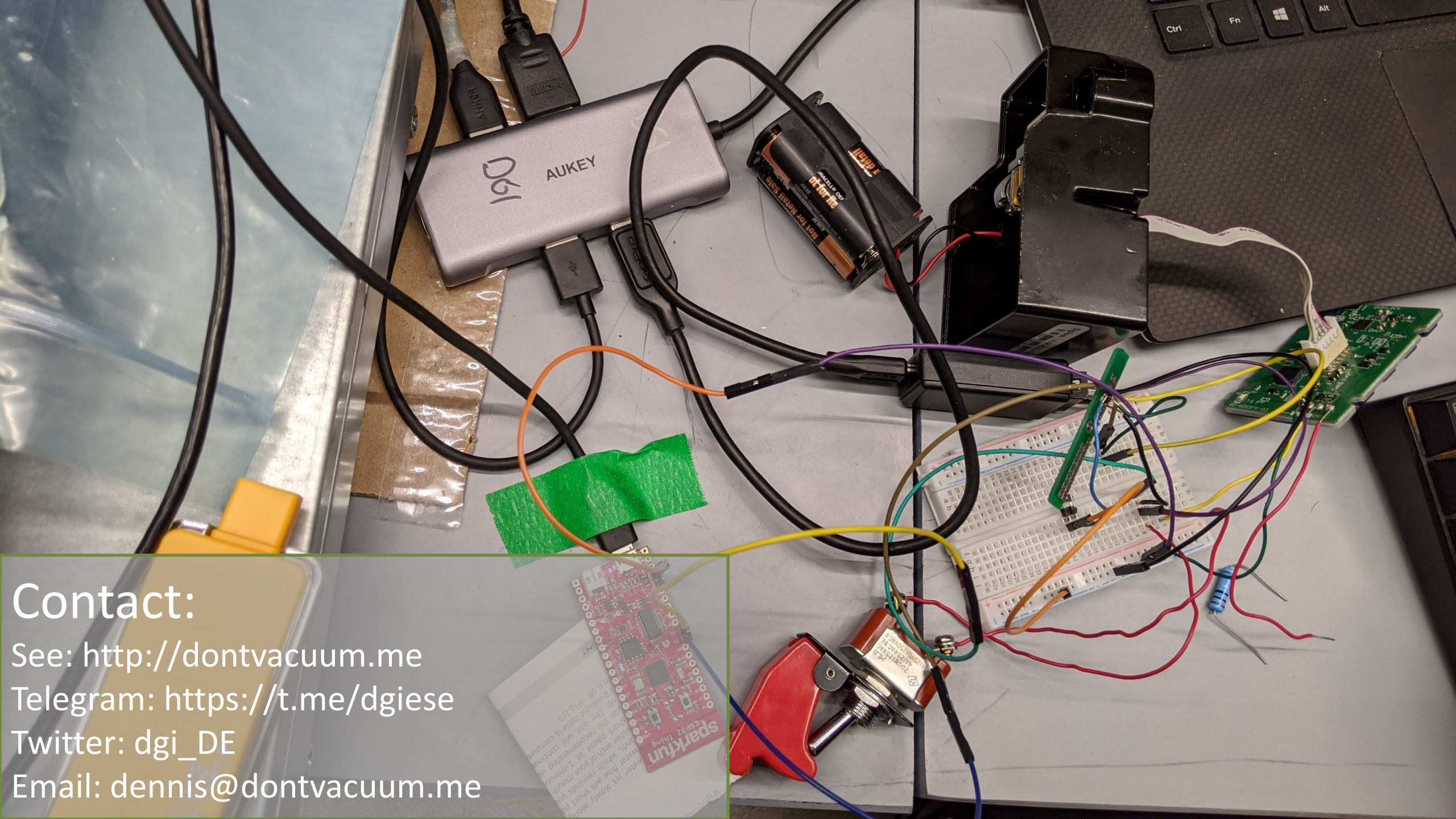
Reminder: The locks are not sold as high-security locks!

Final notes

- Please do not break into lockers you do not own!
- Messing with locks can permanently brick them
- There are more attacks that have not been covered here
- Other companies and products are vulnerable, too
  - Just because someone did not get hacked, does not mean that they are good

Acknowledgements

- Tihmstar
- Shannon Assouline
- Ben Helfrich
- Braelynn
- Sören Beye
- Xenia
- Guevara Noubir



Contact:

See: http://dontvacuum.me
Telegram: https://t.me/dgiese
Twitter: dgi_DE
Email: dennis@dontvacuum.me