Uncovering Vulnerabilities in Hoermann BiSecur

An AES Encrypted Radio System

Markus Müllner, Markus Kammerstetter, Christian Kudera and Daniel Burian

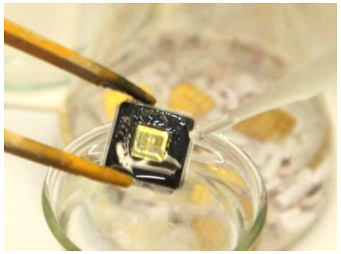
Trustworks KG

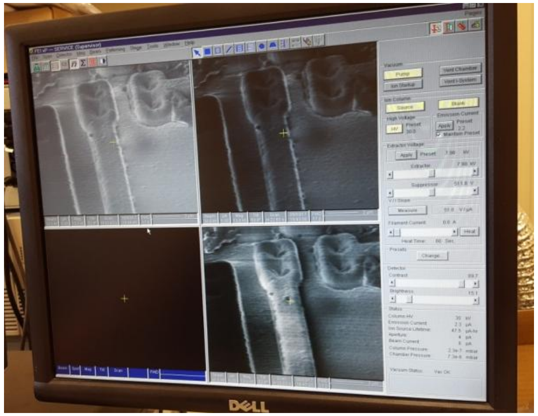
Web: https://www.trustworks.at
E-Mail: office <at> trustworks.at

Who we are

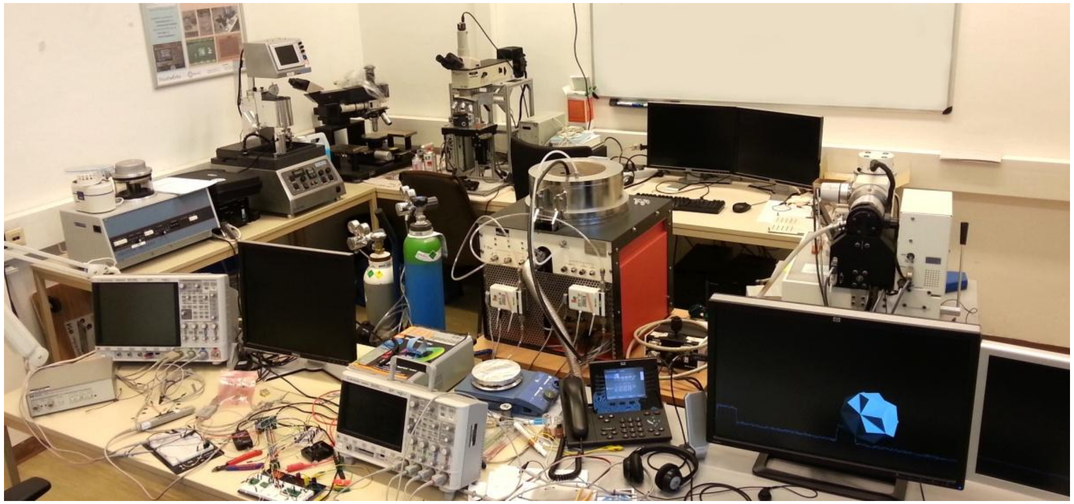
- Security consulting, engineering and research company, founded 2012
- Security consulting services since 2005
- Core business area:
  - Security audits of Web-applications and enterprise software products down to embedded devices and microchips
  - Embedded & Security Engineering
  - High-speed cryptography
- Own Hardware Security Lab to support embedded security audits

Our Hardware Security Lab





- Dedicated lab equipment such as FIB and SEM to conduct embedded security audits down to the microchip level
- Non-invasive (SCA, FI) to fully-invasive (FIB/SEM) evaluation methods
- General idea: More advanced lab tools reduce required analysis time during security audits



Why Garage Doors ?

- Many wireless garage door systems are known to be insecure:
  - Mostly static or simple rolling code schemes
  - No encryption
  - Replay attacks and cloning possible
- Hoermann BiSecur:
  - Use of AES algorithm, established high-security system
  - Big security improvement over "classical" systems
  - Two of our security analysts already had a BiSecur system at home

Open Questions …

We have those systems at home, are they really secure ?

- Is the system implementation secure in practice ?
  - How does the system use AES encryption ?
  - How is the key material generated and used ?
  - Is the key material individual ?
  - What messages are exchanged on the RF interface ?
  - Kerckhoff's principle: Is the system secure if all system internals except for the key material are known ?

→ We decided to conduct a security audit

Hand Transmitters

- We already had a few
- We also obtained new ones

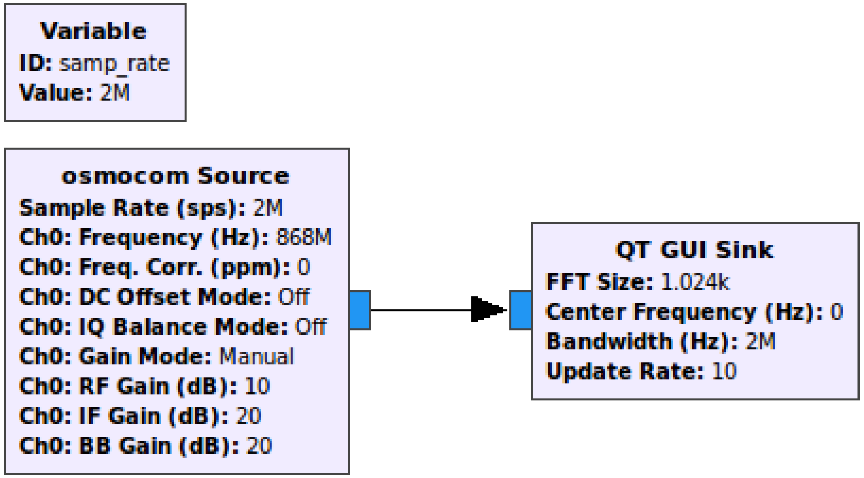
- Result:
  - Different models
  - Different manufacturing dates
  - Our analysis will cover a broader range

RF Signal Analysis

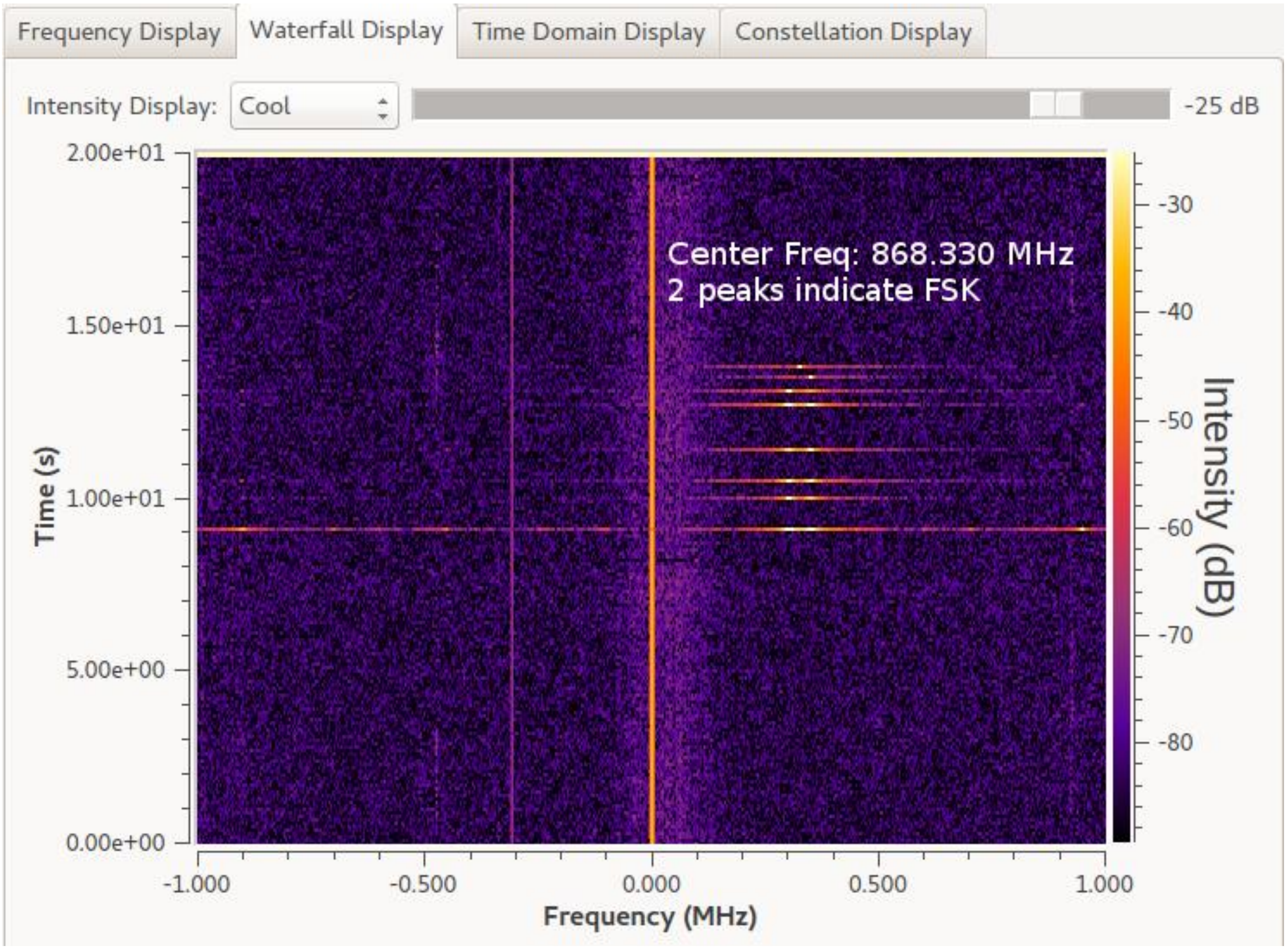
- Manual: Signal at 868 MHz
- Use of BladeRF SDR
- Goals:
  - Find exact frequency
  - Identify modulation scheme
  - Identify channel coding
  - Decode RF frames

Finding the Exact Frequency

- GNU Radio SDR Suite
- Spectral analysis (Waterfall Plot)



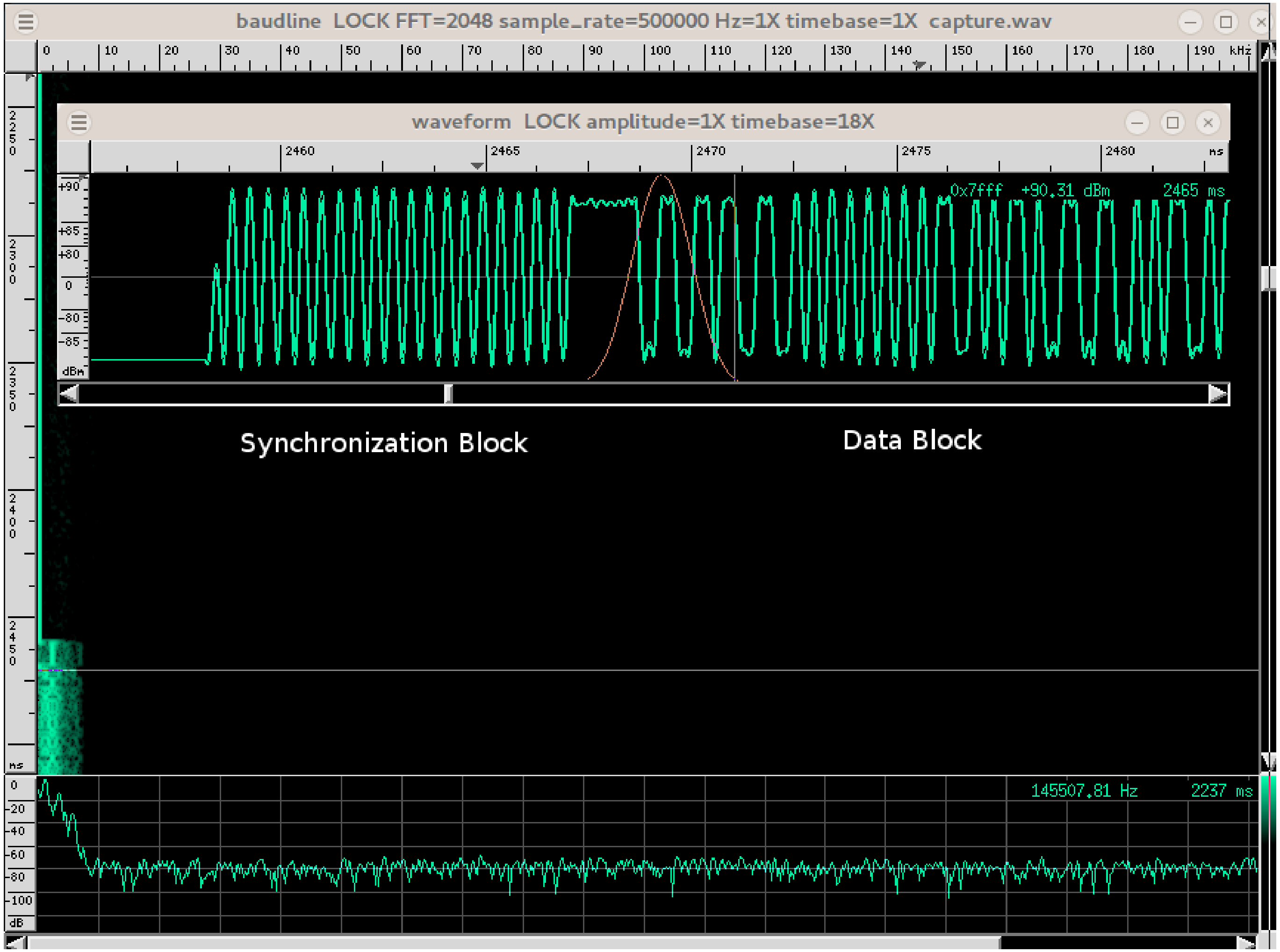
Finding the Exact Frequency



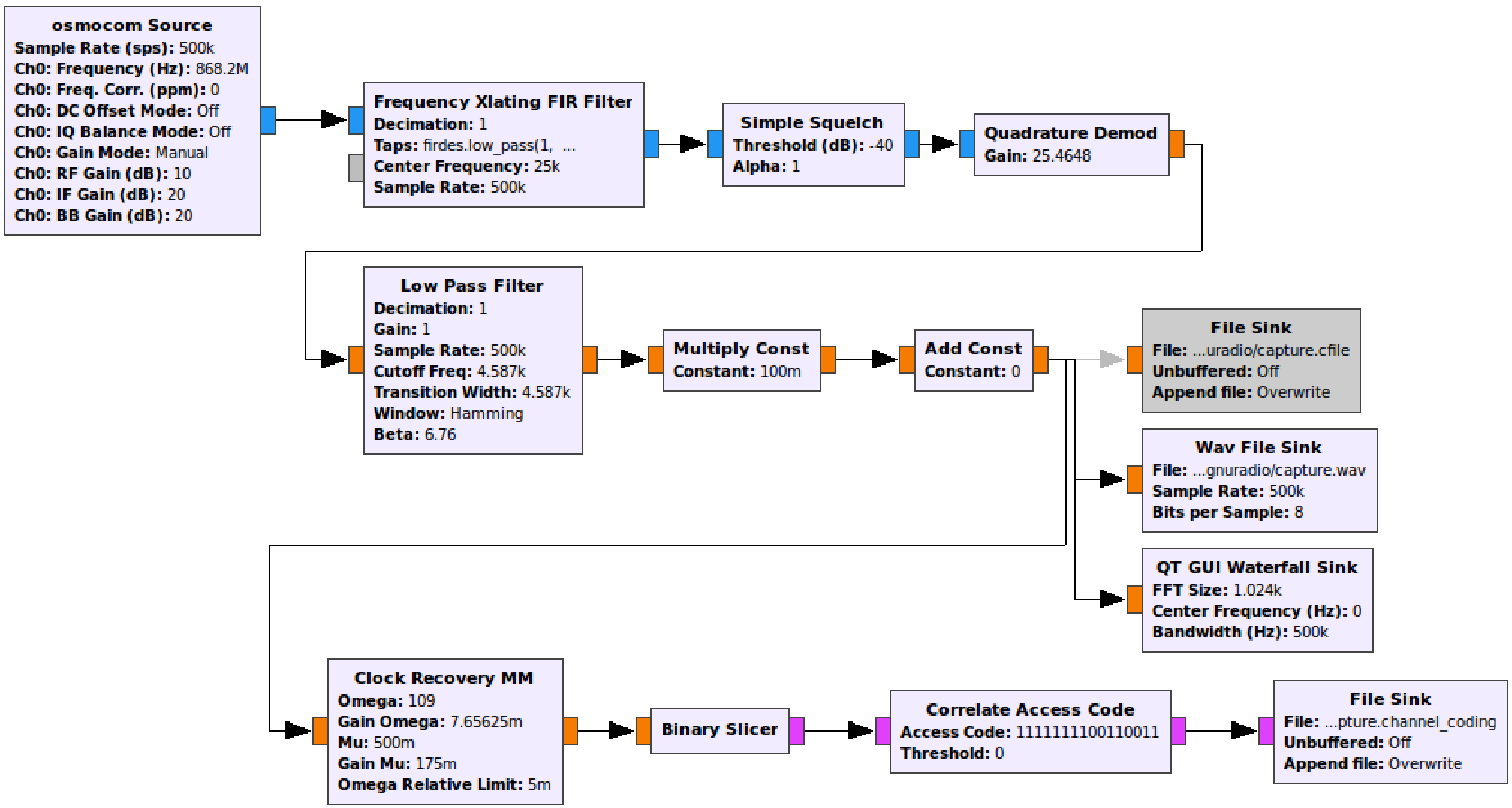
Identify Channel Coding

- Typical transmissions contain a synchronizing block (i.e. alternating high/low waveform)
- Allows receiver to synchronize its symbol rate to the symbol rate of the transmitter
- Allows us to determine the symbol rate as well

Identify Channel Coding

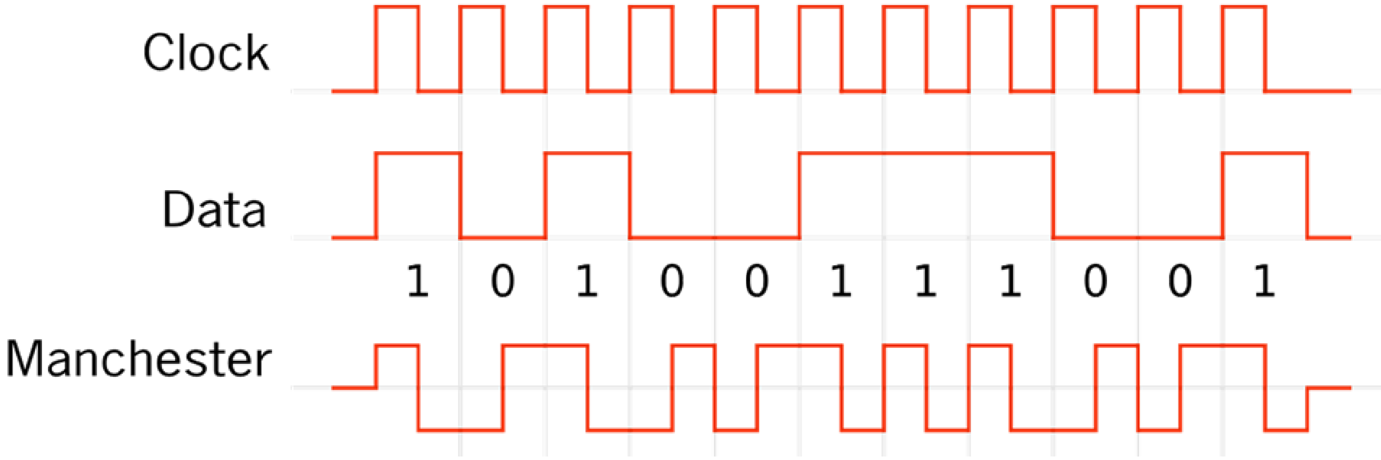


Identify Channel Coding



Channel Coding Result

- We analyzed the captured dump file
- Demodulated signal contains either ‘01’ or ‘10’ sequences --> Manchester encoding
- 2 symbols represent 1 bit of data



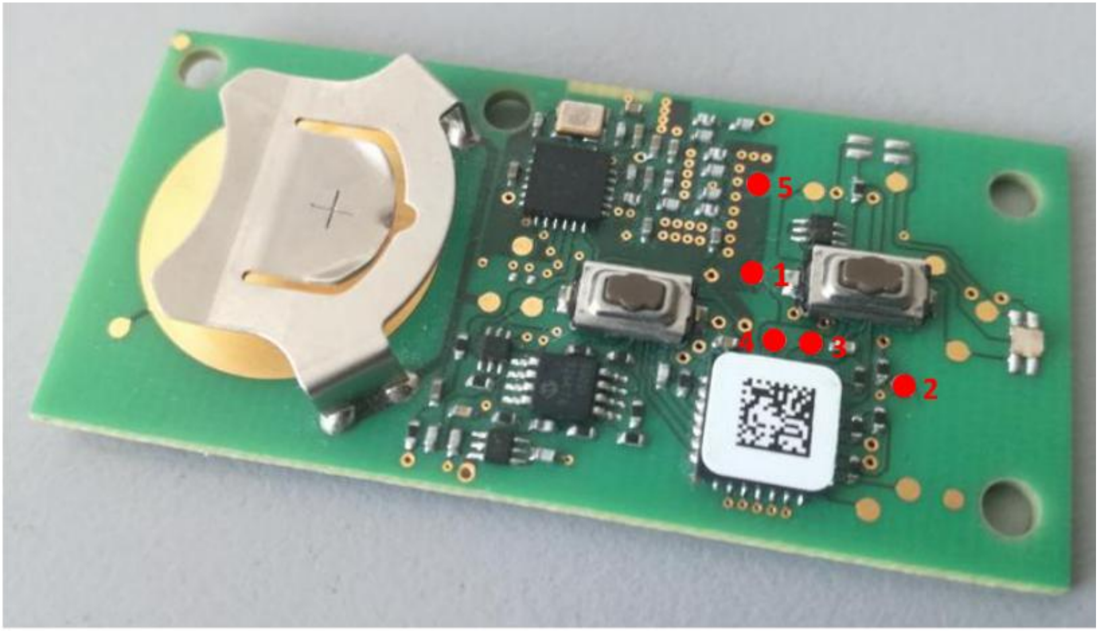
[Wikipedia, Manchester Code]

Decoding RF Frames

- Decoding of data bits from recorded frame (Python script with GNU Radio framework)
- Assumption: CRC value at end (unknown CRC)
- Not sure yet if we got everything right

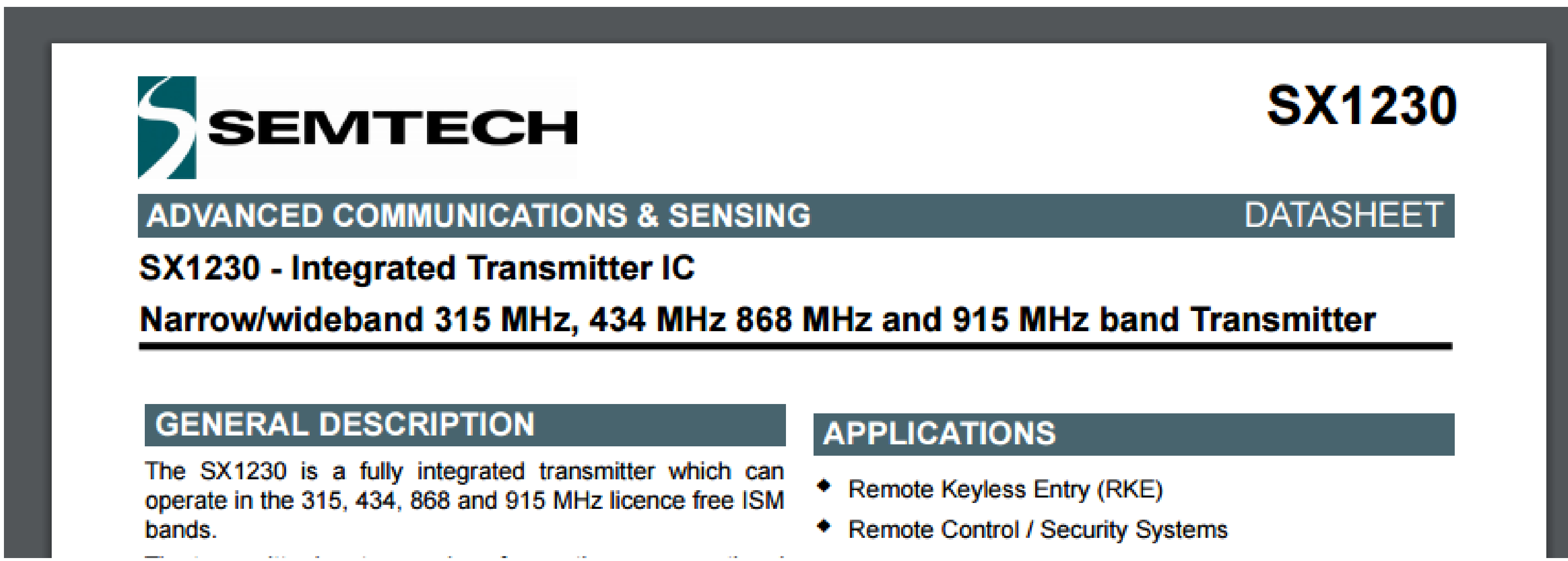
Field	Length (Byte)	Comment
Constant	1	0x70, 0x50
Serial Number	4	unique to device
Encrypted Data	16	
Checksum	1	CRC (unknown)

Hardware Analysis



Hardware Components

- SX1230 --> Radio Chip with public datasheet



SEMTECH

SX1230

ADVANCED COMMUNICATIONS & SENSING DATASHEET

SX1230 - Integrated Transmitter IC

Narrow/wideband 315 MHz, 434 MHz 868 MHz and 915 MHz band Transmitter

GENERAL DESCRIPTION

The SX1230 is a fully integrated transmitter which can operate in the 315, 434, 868 and 915 MHz licence free ISM bands.

APPLICATIONS

- Remote Keyless Entry (RKE)
- Remote Control / Security Systems

Hardware Components

- Unknown microcontroller (PIC ?)
- Microchip Logo and „EE001 20000 1507AHW“
- What do we know ?
  - 28 Pins
  - QFN package
  - Chip older than 3 years, so no Atmel
  - GND on Pins 5 and 16
  - PCB likely has test points for programming

Identifying the Controller

- Let‘s search a component distributor:
  - “manufacturer: Microchip”, 28-pins, QFN
  - Results: PIC16F, PIC18F, PIC24F, PIC32F
  - Let’s look into their datasheets !

- PIC24 and PIC32 would need GND on Pin24, we don’t have that on the PCB
  --> It must be either PIC16F or PIC18F

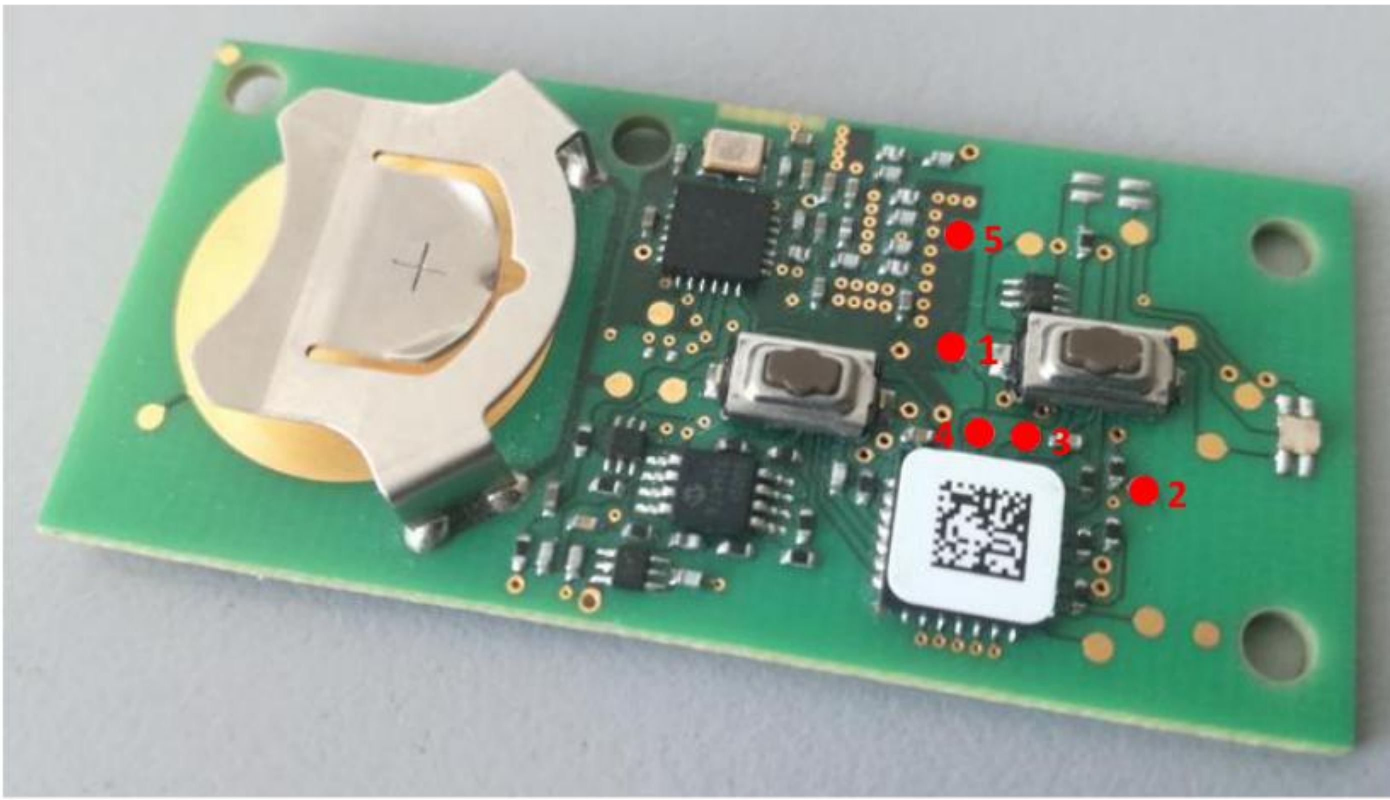
Programming Pins

- Pins required for PIC programming are: /MCLR, PGD, PGC as well as Vdd and GND

- 28-pin PIC16F and PIC18F have their programming pins at the same location

- Let’s PICkit3 it to read the device number !

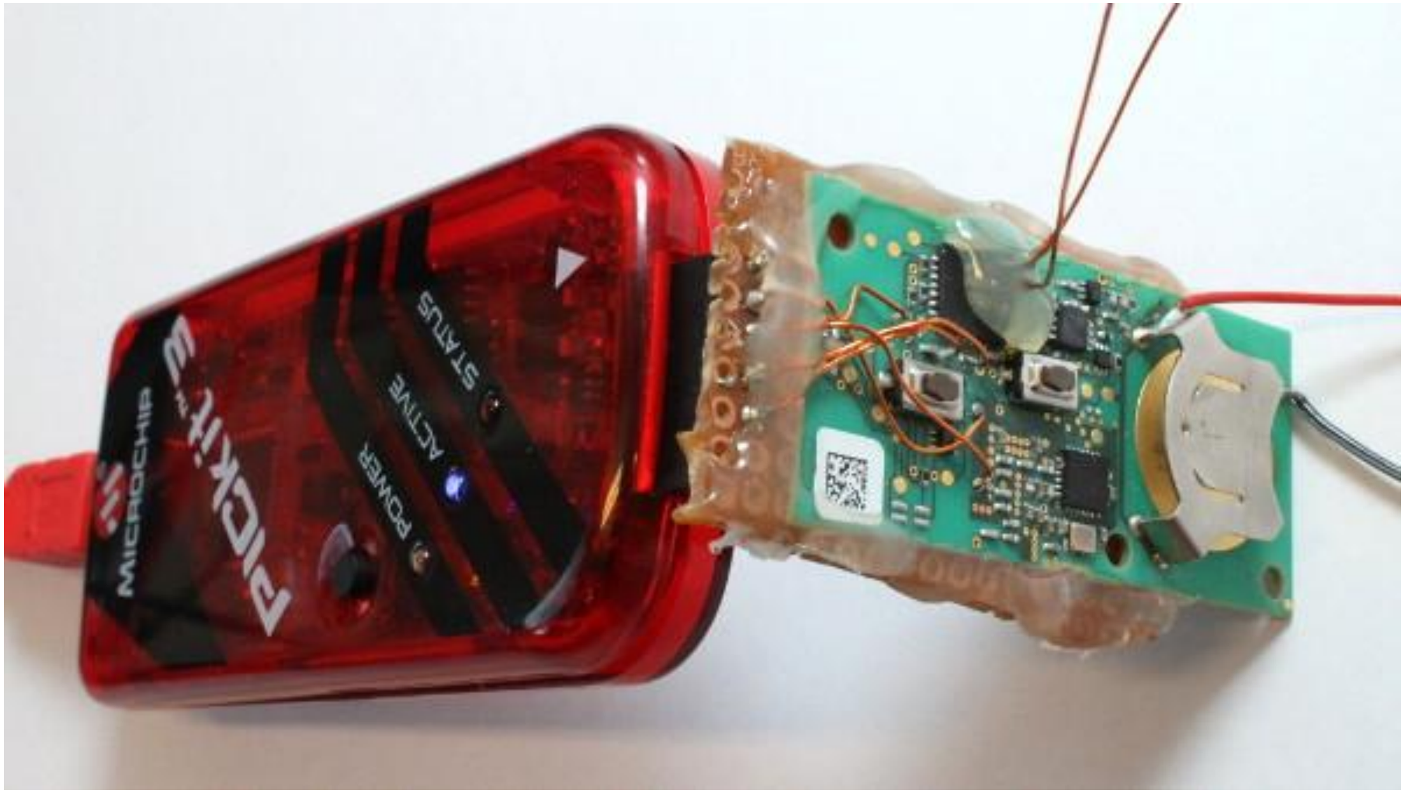
Programming Pads



1	$\overline{\text{MCLR}}$
2	V_{DD}
3	V_{SS}
4	PGD
5	PGC

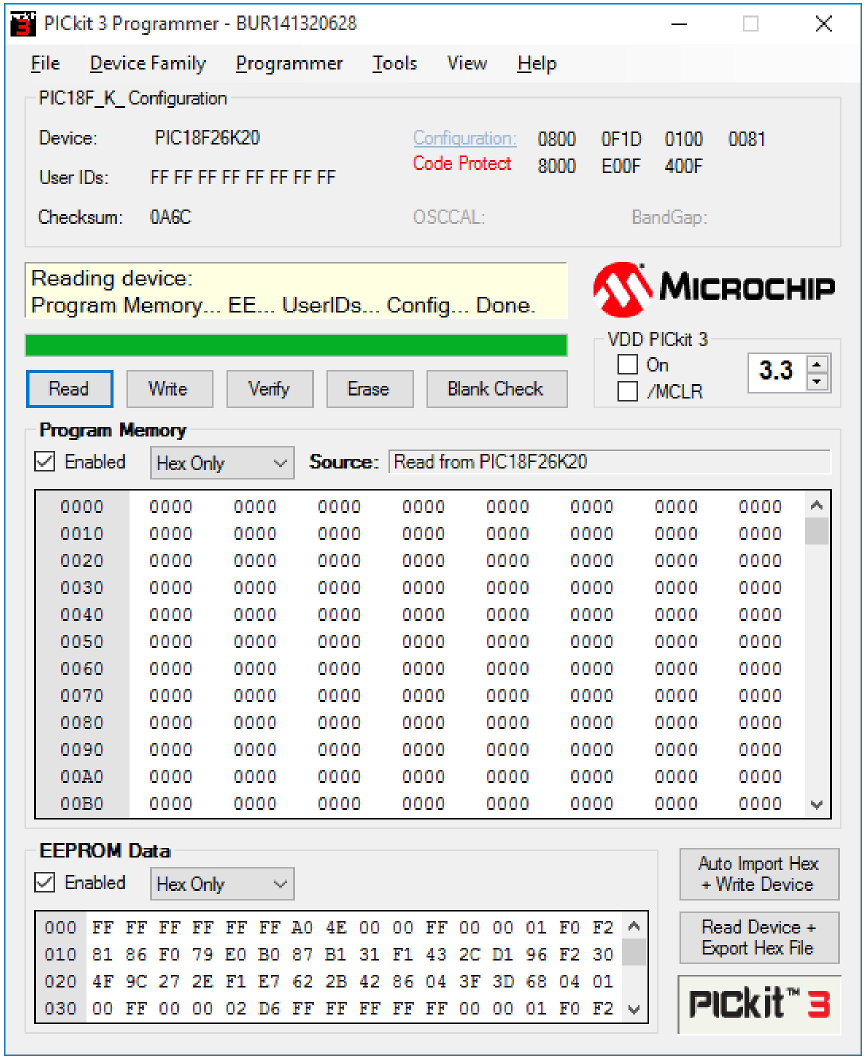
PICkit3

- We created a little break-out board



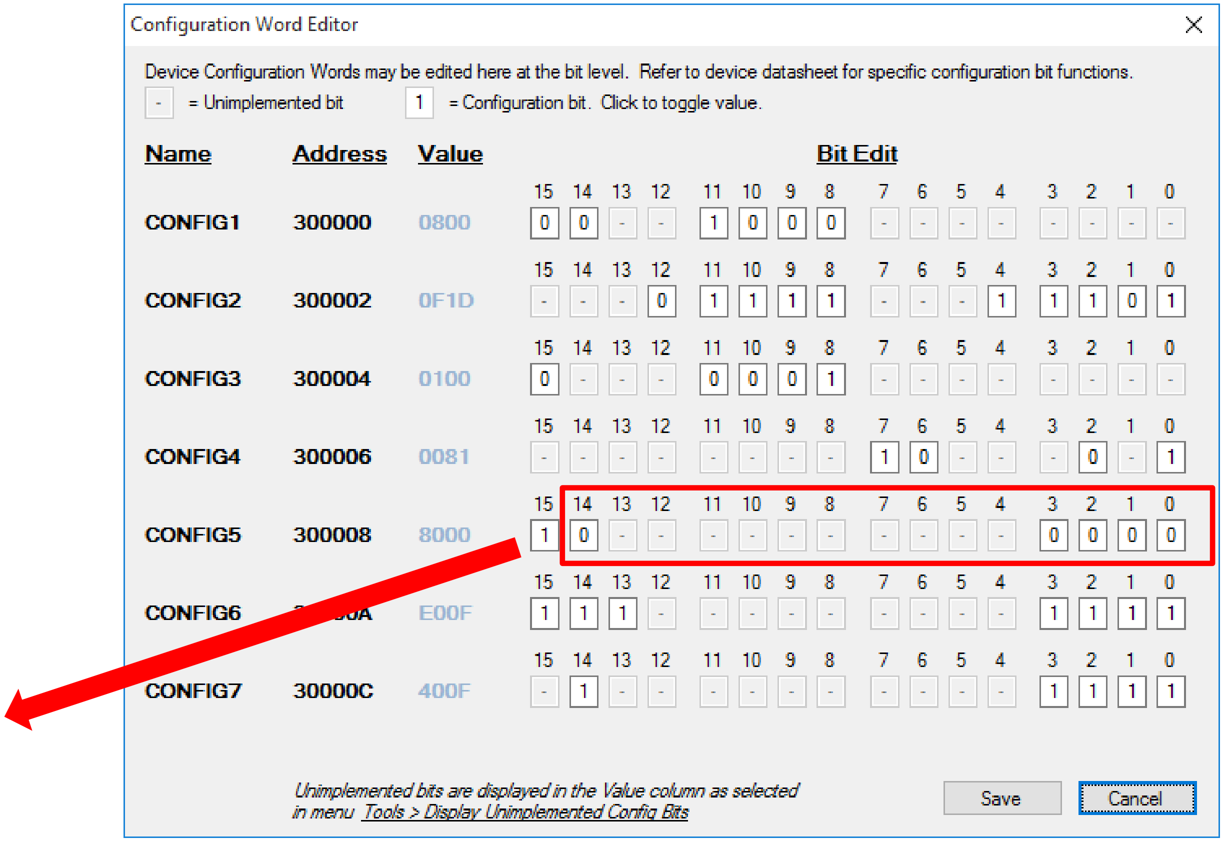
PICkit3: Results

- MCU is PIC18F26K20



PICkit3: Results

- All Flash blocks are locked (code protection)



EEPROM not locked

PIC Firmware Extraction Techniques

- To analyze the firmware and thus the system for security vulnerabilities, we need to analyze the firmware

- The firmware is currently locked

- PIC locking mechanism:
  - "locking bits" --> security fuse

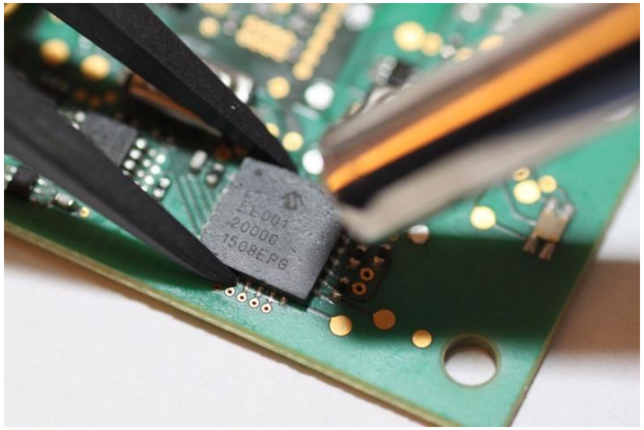
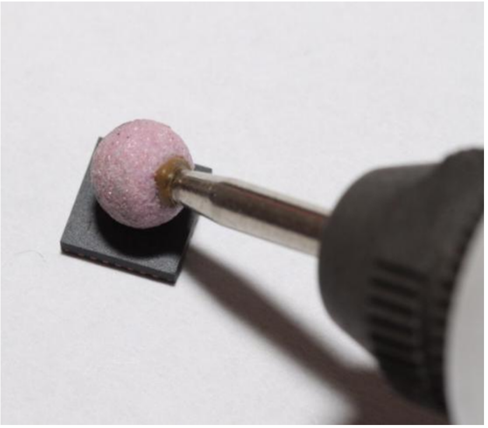
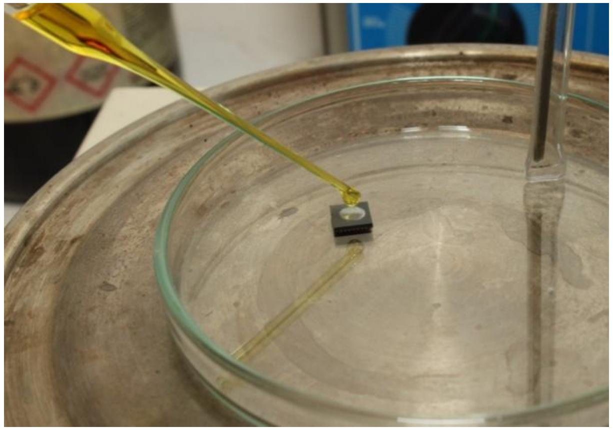
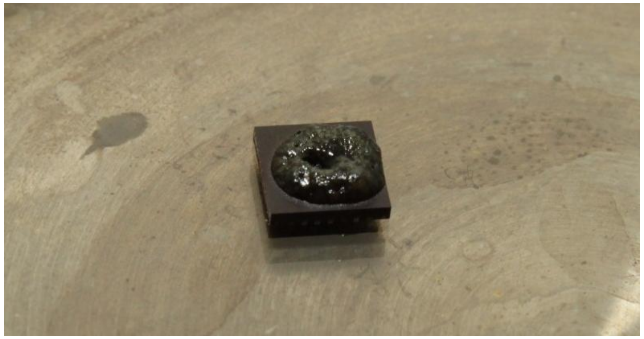
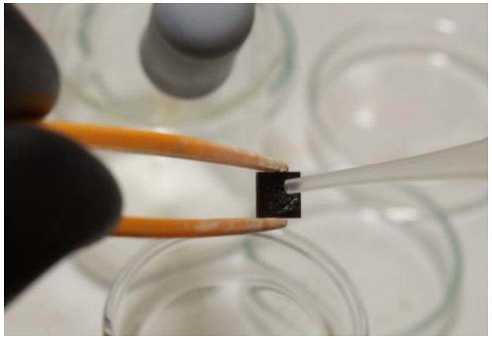
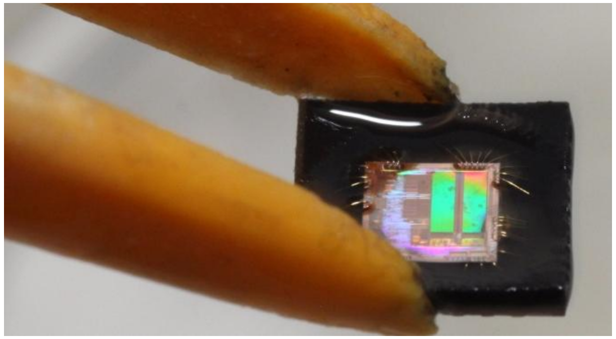
- Back to security-by-obscurity ?

IC Analysis

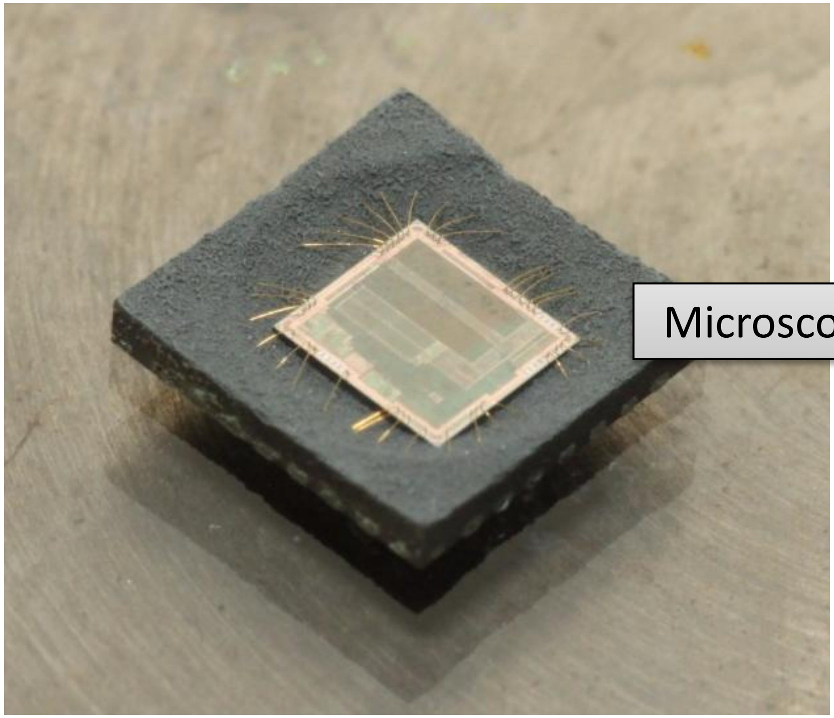
- How is the security fuse logic implemented on the PIC18F microcontroller ?

- Approach:
  - IC Decapsulation
  - (Rough) microscopic analysis

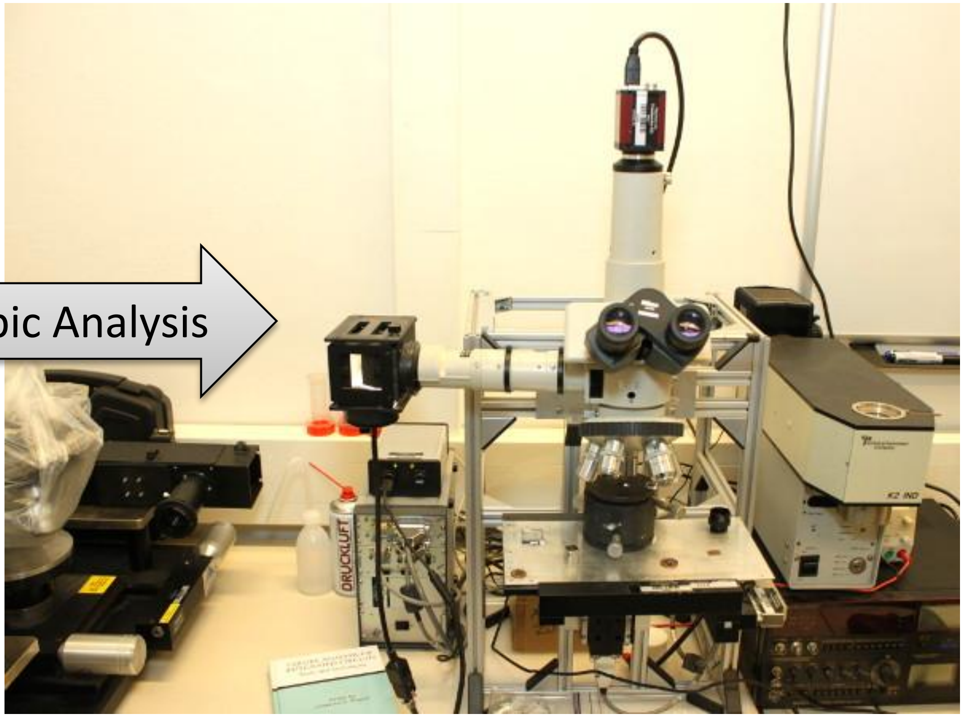
IC Analysis

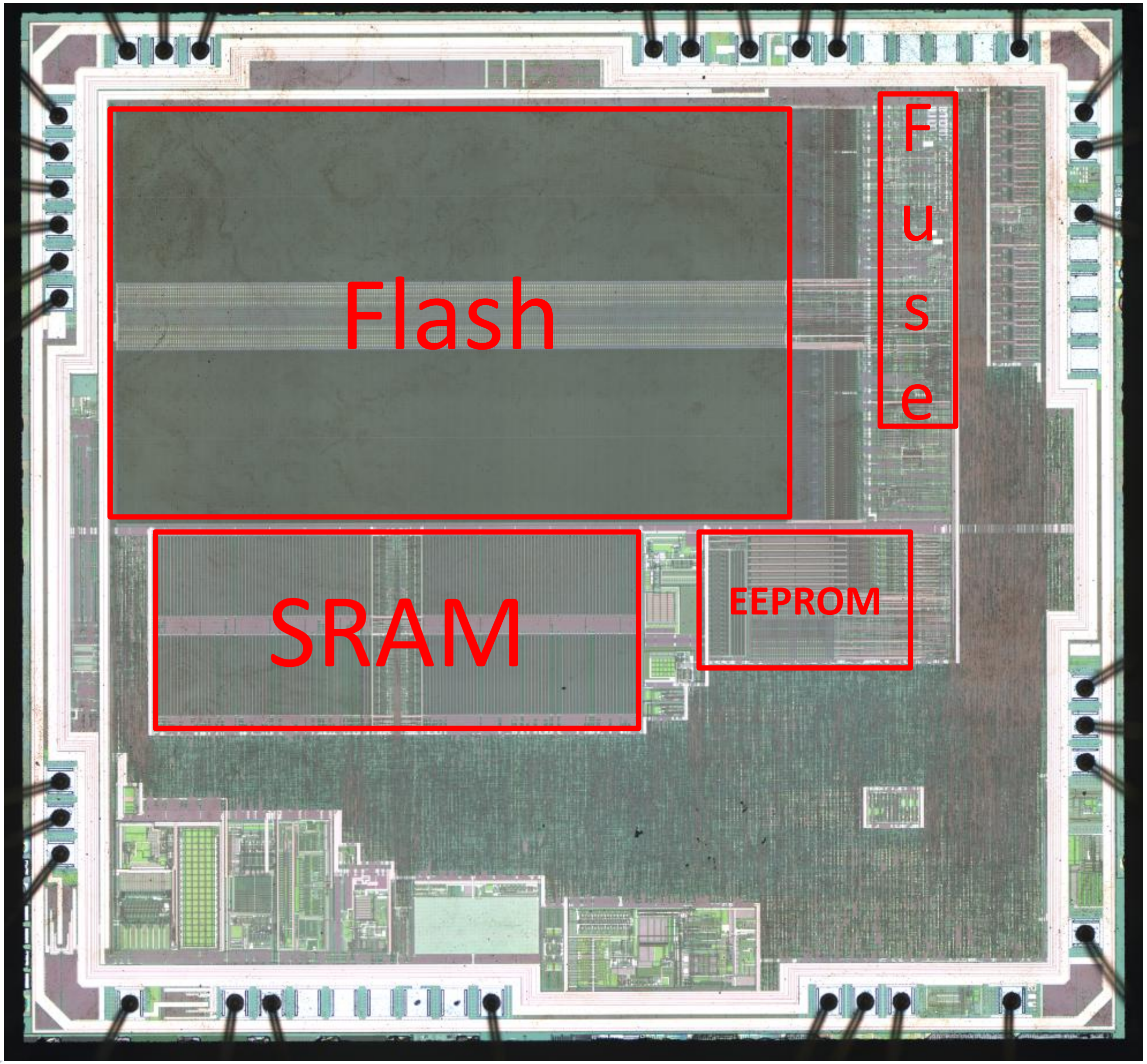
IC Analysis



Microscopic Analysis



IC Analysis



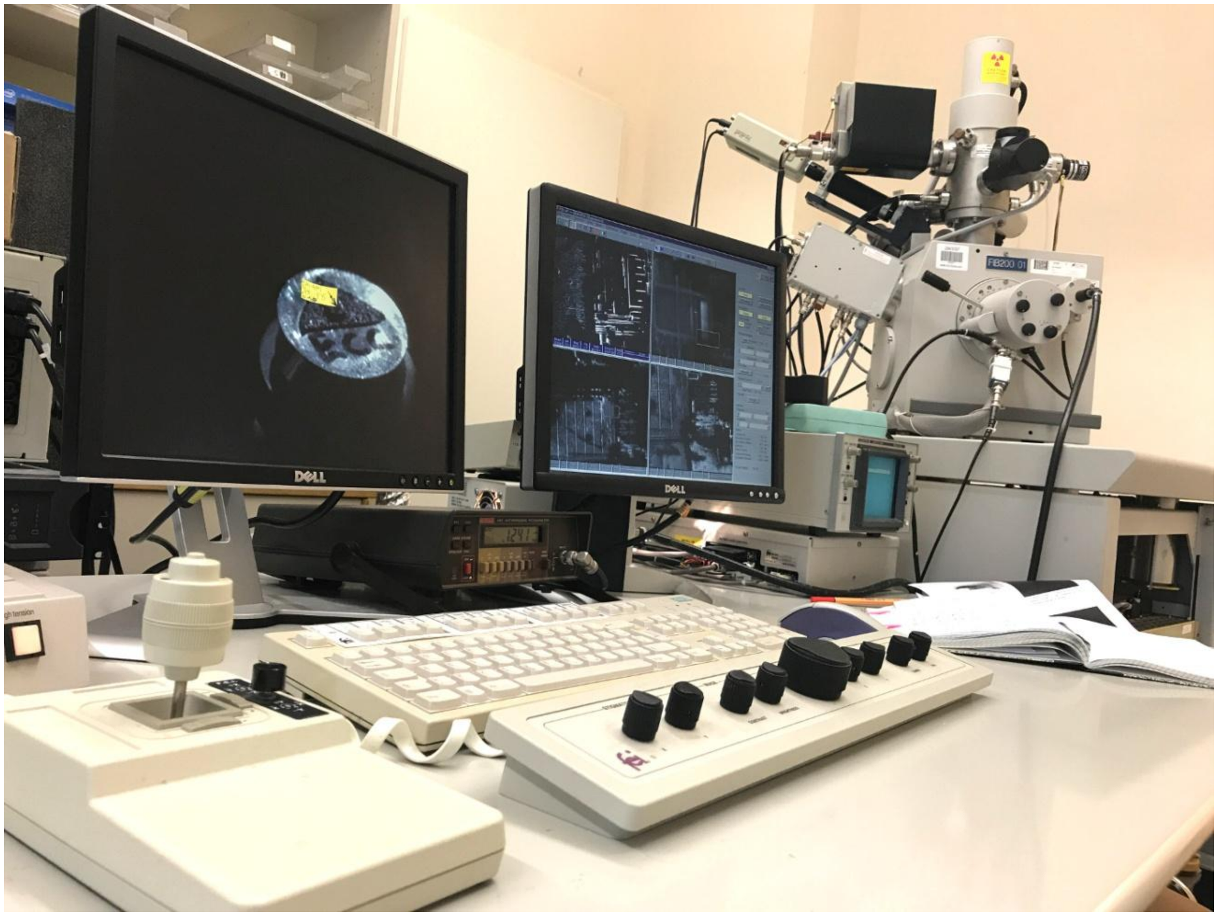
Fuse Close-up

Method 1: Invasive FIB Circuit Edit

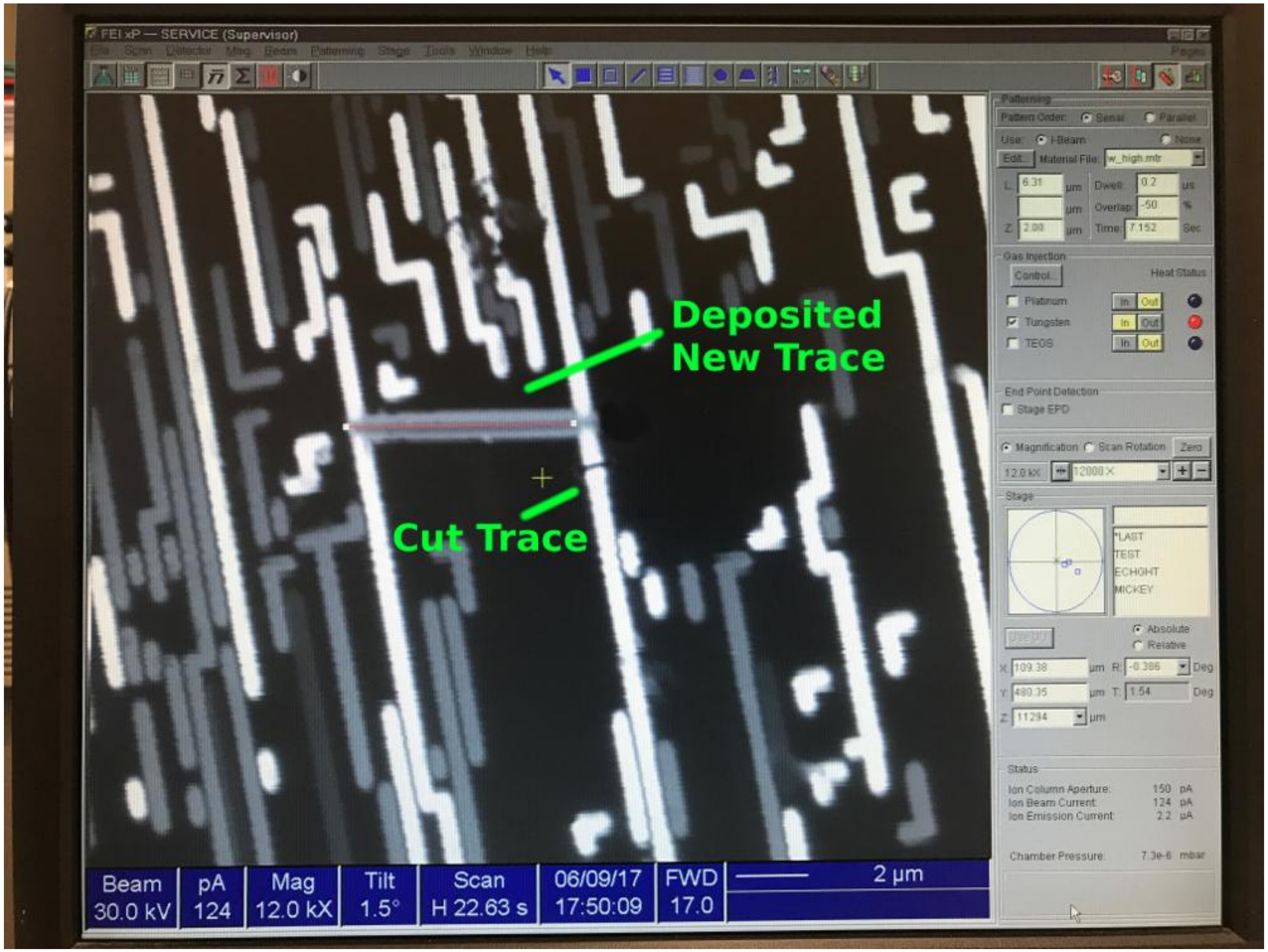
- Approach:
  1. Identify security fuse logic (might involve IC deprocessing and analysis other PICs)
  2. Bypass security fuse with FIB circuit edit

- Advantage: High success rate
- Disadvantage: Large time effort if no existing recipe

Method 1: Invasive FIB Circuit Edit



Method 1: Invasive FIB Circuit Edit



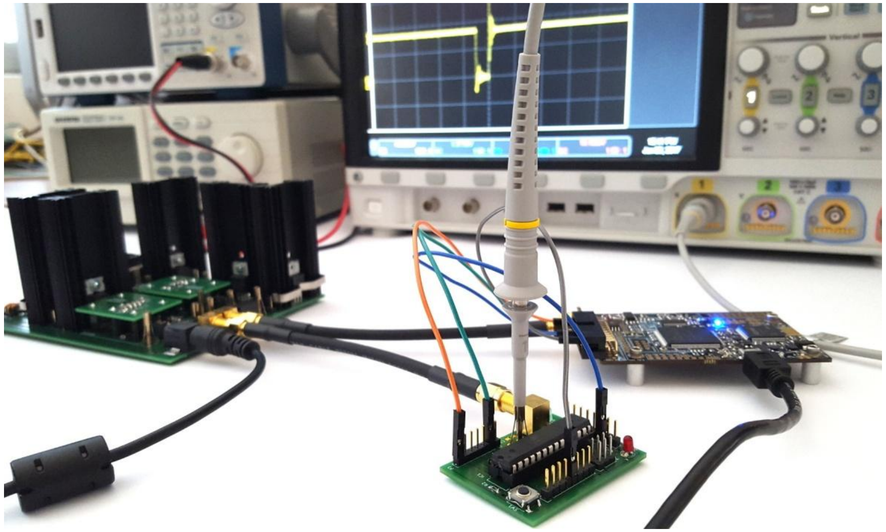
Method 2: „Bunnie“ Attack

- Fuses can be cleared with UV, but Flash needs to be protected (e.g. tape or permanent marker)
- Shields avoid direct UV exposure (see our die image)
- UV light might still get in at a steep angle
  --> “Bunnie” attack (original idea by Bunnie Huang)

- Advantage: Easy to conduct and our chip is already open
- Disadvantage: Unclear whether we have anti-fuses, potential to damage bond-wires during masking

Method 3: Voltage Glitching

- Our glitch amplifier can deliver <12ns pulses at high drive current



- Advantage: easy to conduct
- Disadvantage: Unclear if successful on PIC18F, glitch parameters need to be found first

Method 4: The Easy Way :-)

- There is a trivial design issue in the PIC18F security fuse logic

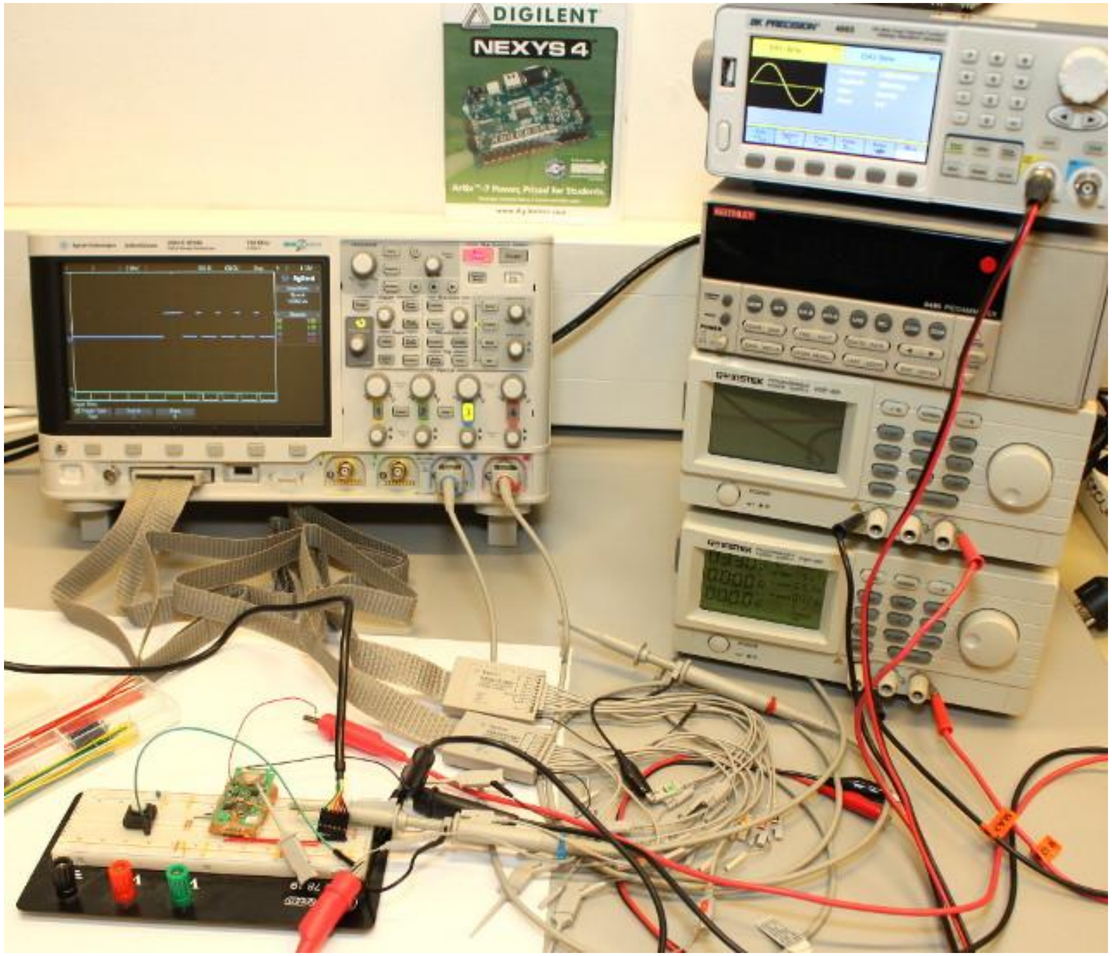
- Memory blocks have individual fuse bits

- A block can be reprogrammed while the others keep their original content

- Presented at 27C3 (2010):
  - [Milosch Meriac, 27C3: Heart of Darkness – exploring the uncharted backwaters of HID iCLASS security, https://www.openpcd.org/images/HID-iCLASS-security.pdf]

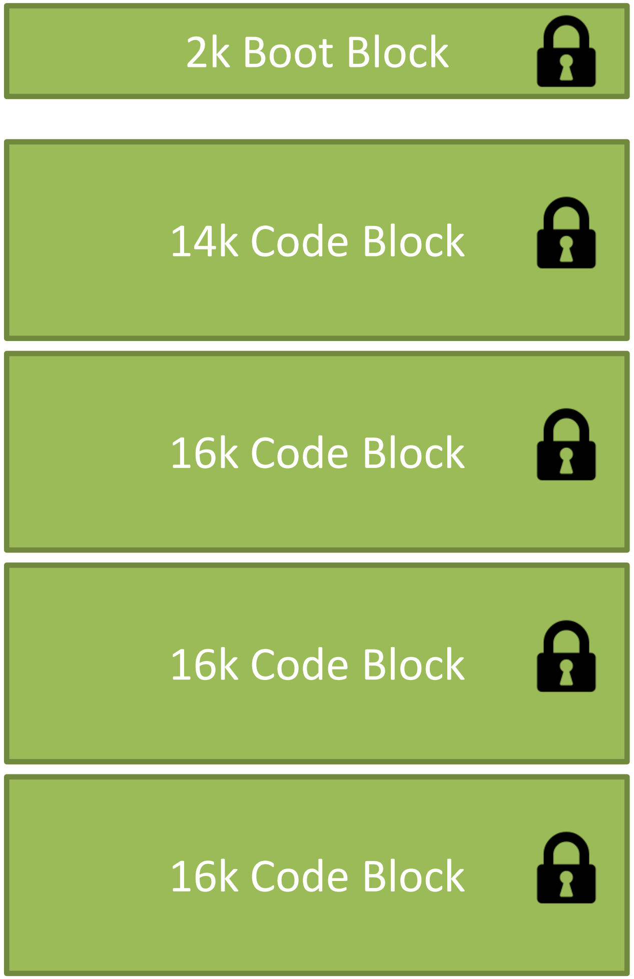
We Implemented a PIC Firmware Extraction Tool

- Breadboard HW
- FTDI (USB-TTL)
- Short SW Script



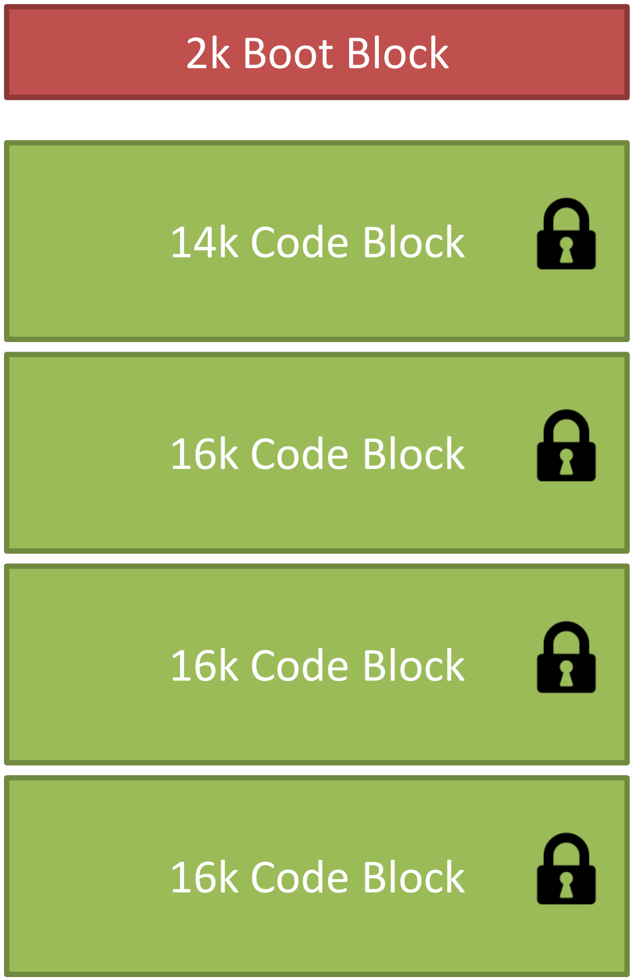
PIC Firmware Extraction

- 5 memory blocks (Flash)
- Individual lock bits

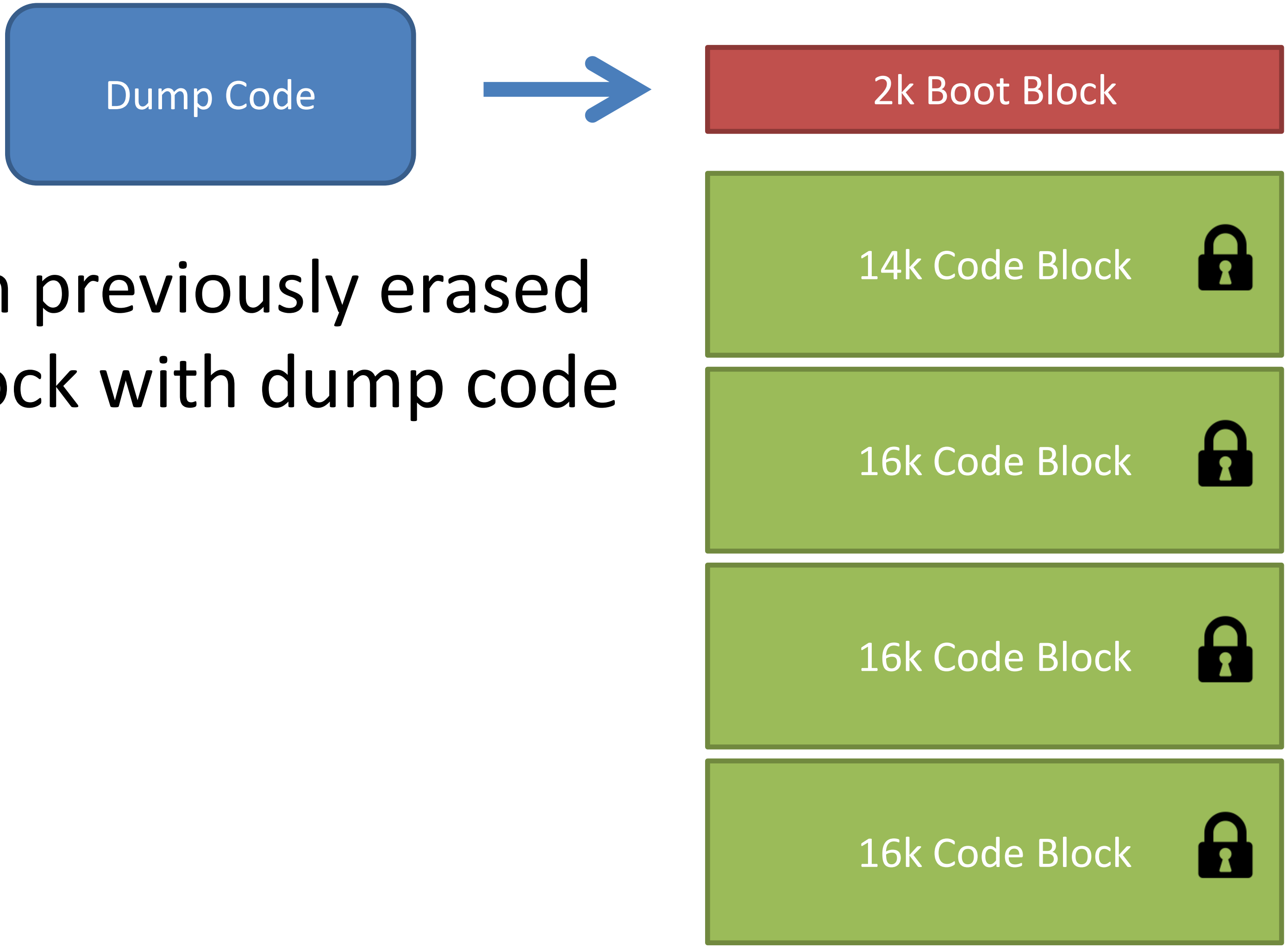


PIC Firmware Extraction

- Remove lock bit of boot block --> boot block is erased
- The other blocks remain intact



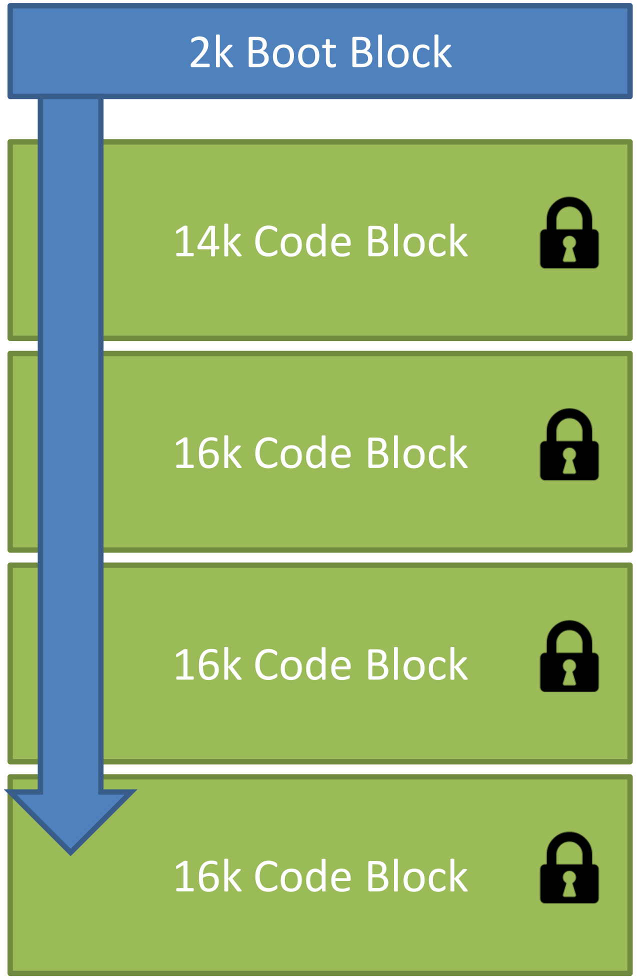
PIC Firmware Extraction



- Program previously erased boot block with dump code

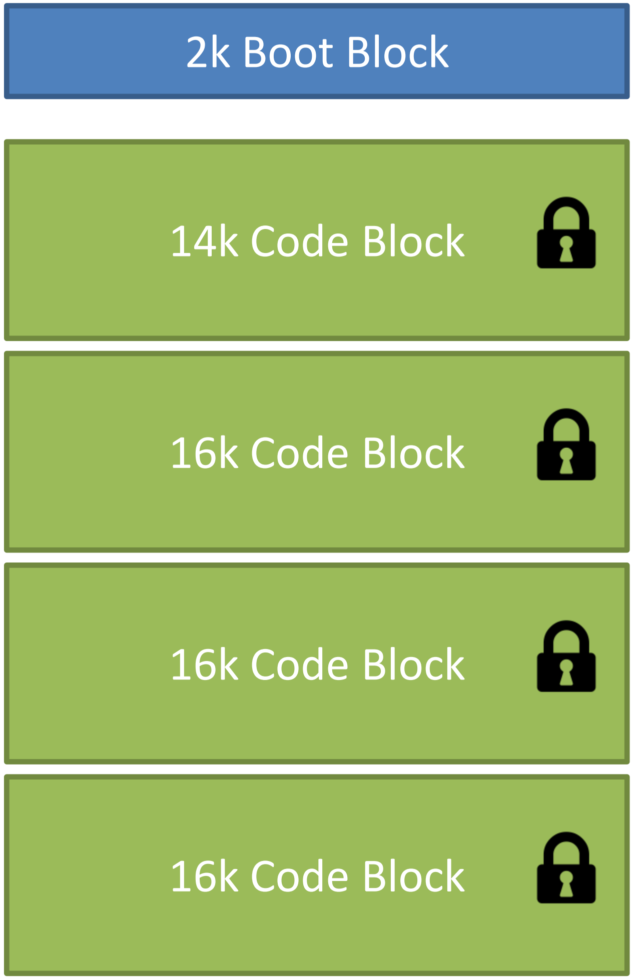
PIC Firmware Extraction

- Dump code reads remaining 4 code blocks
- Output on UART



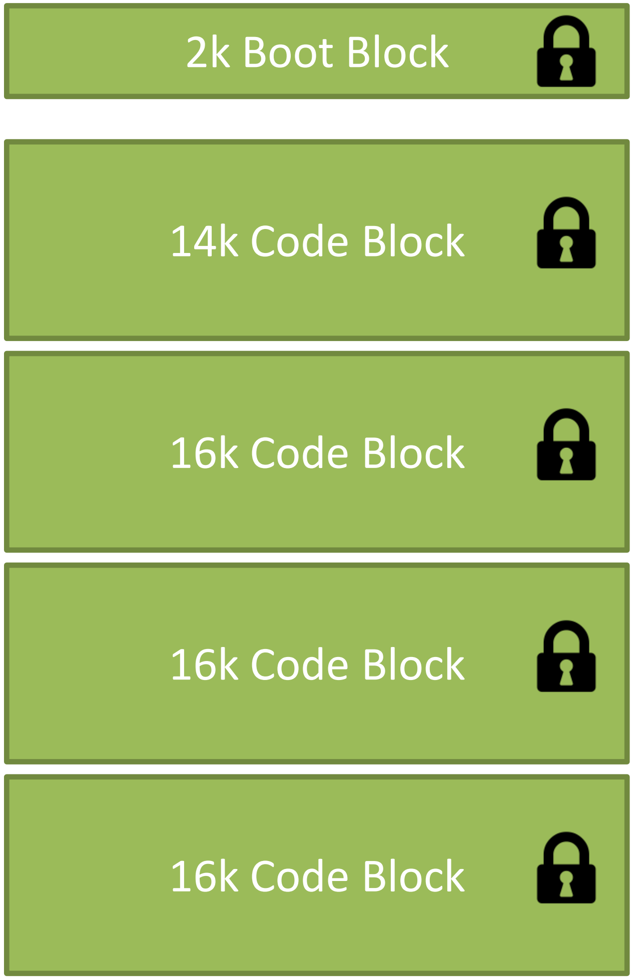
PIC Firmware Extraction

- We have a successful dump of the 4 code blocks now
- We overwrote the 2k boot block and the content is lost :-(



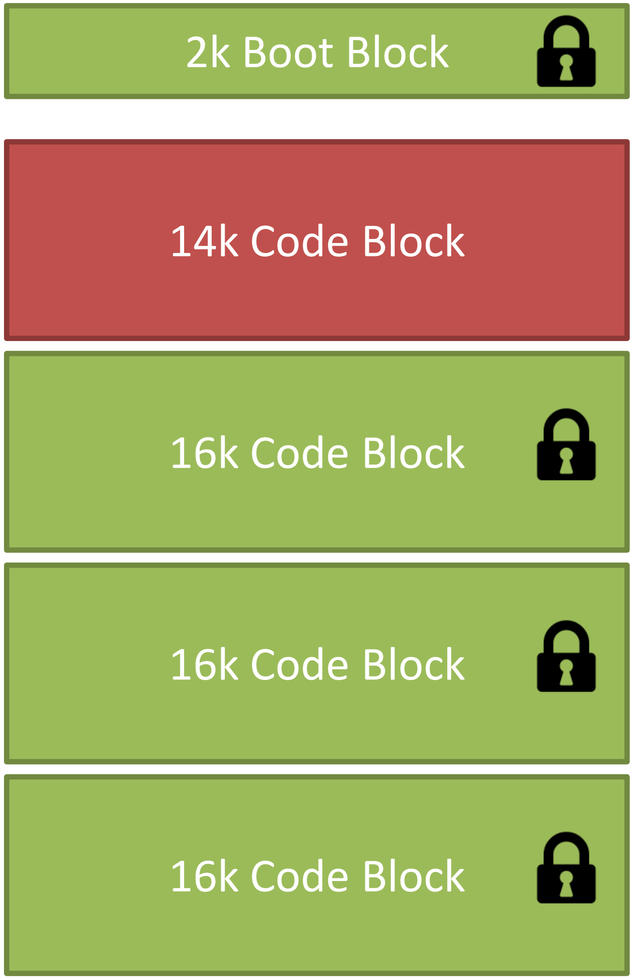
PIC Firmware Extraction

- Solution: Take another PIC with identical programming
- Restart the process

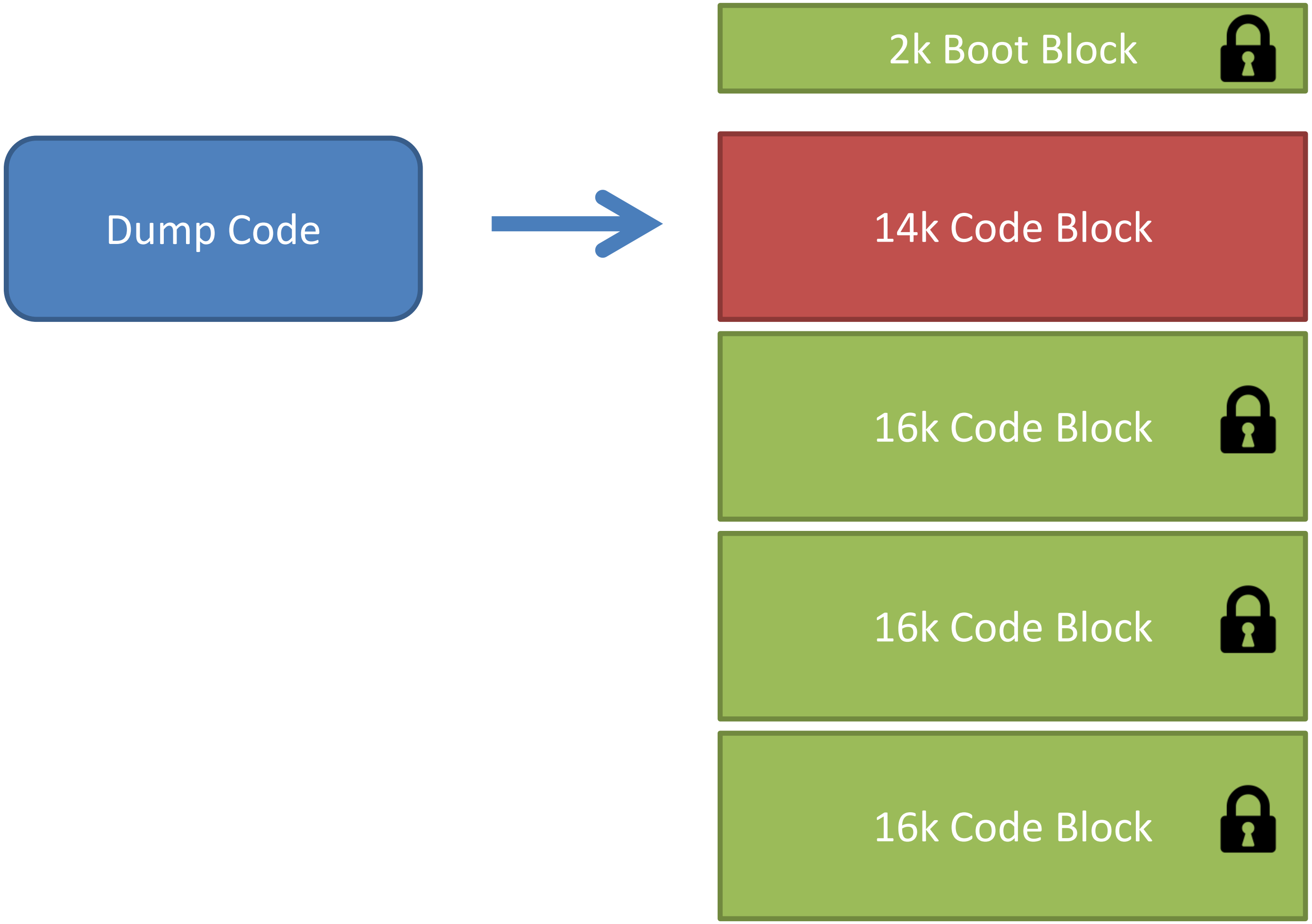


PIC Firmware Extraction

- This time we unlock a code block
- The code block is thus erased

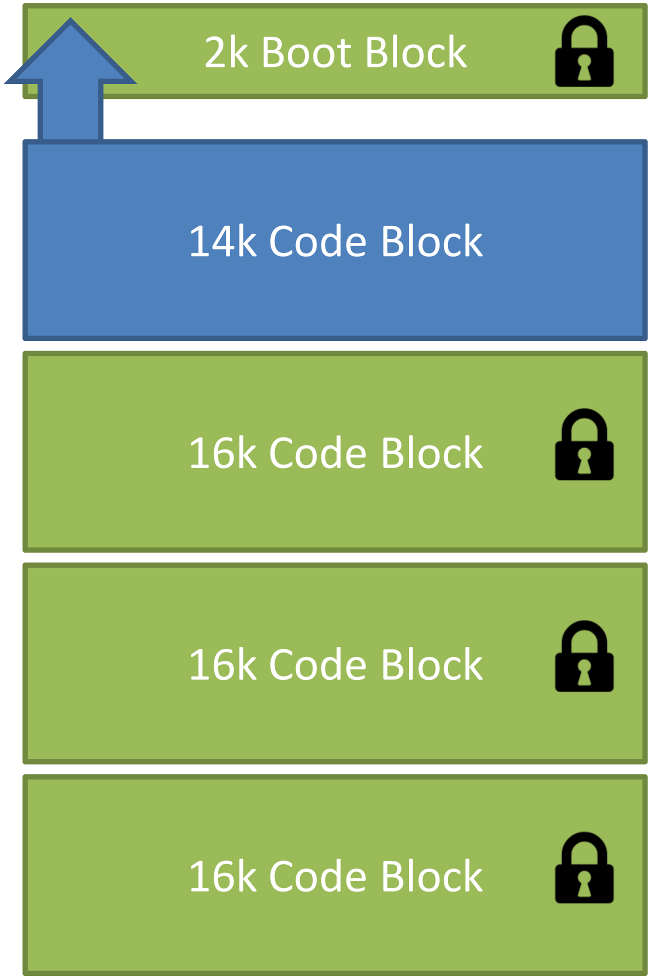


PIC Firmware Extraction



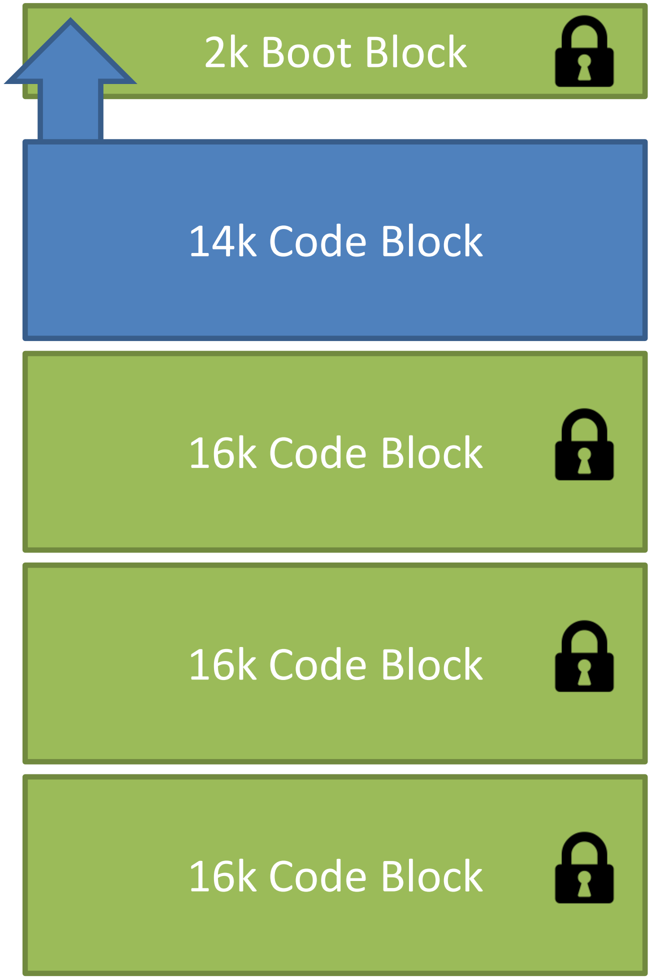
PIC Firmware Extraction

- Wait until execution jumps from another block to our dump block
- NOP slide
- Read the boot block



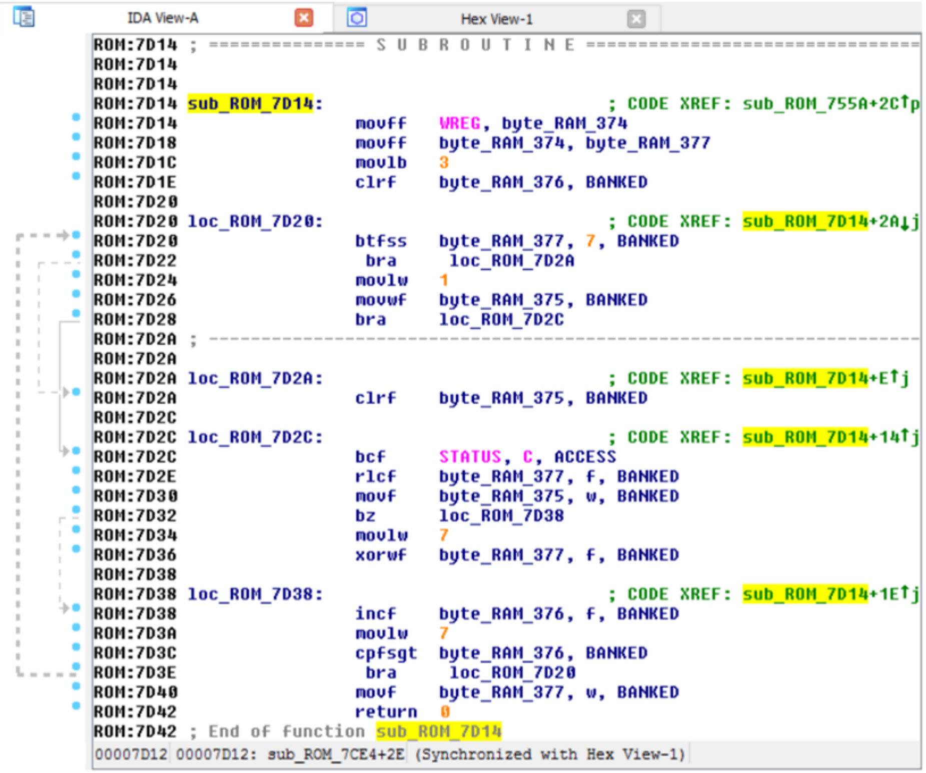
PIC Firmware Extraction

- Both PICs are (partially) dumped now
- We combine the two partials dumps to obtain a full dump
- We can re-flash the two PICs so that they contain their original programming



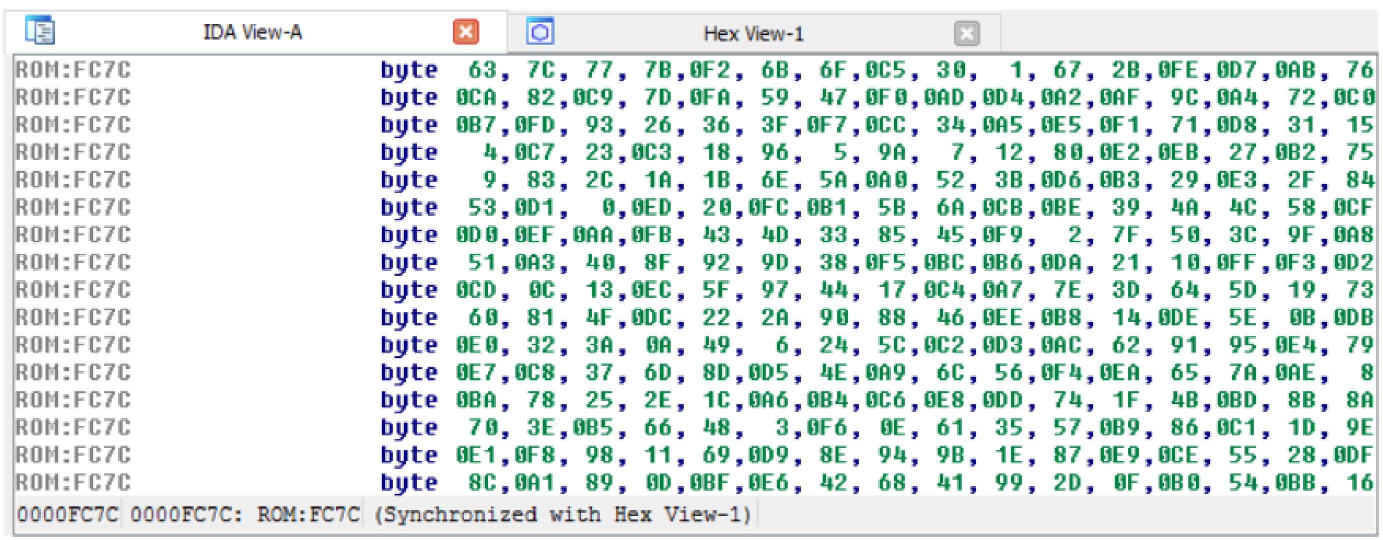
Firmware Analysis

- Analysis with Ida Pro tool (supports PIC)



Firmware Analysis

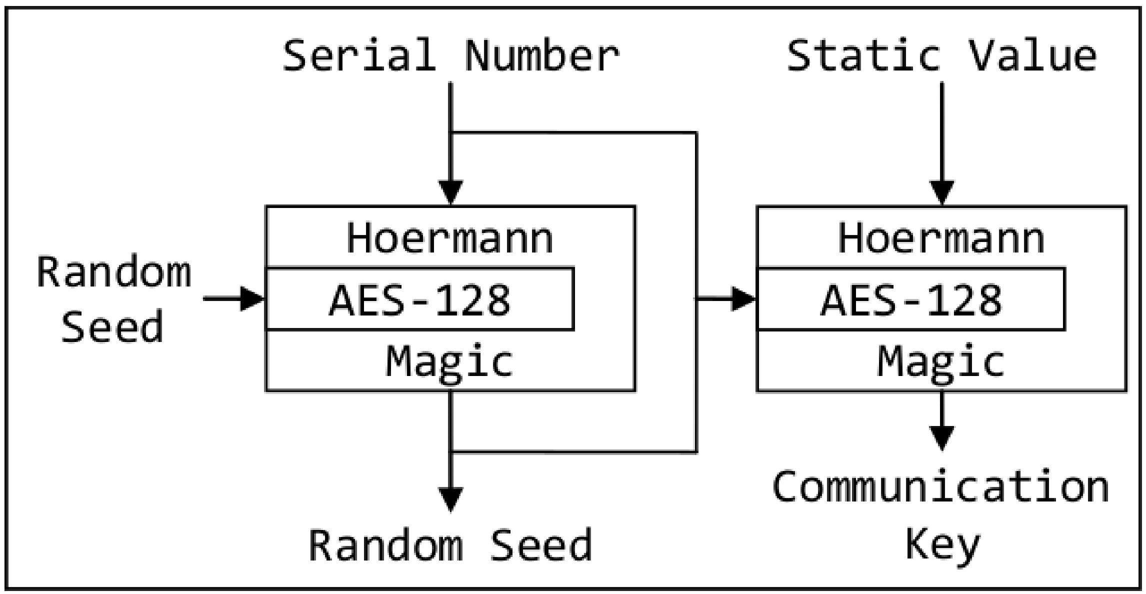
- Firmware analysis provides deep implementation insight, including: RF protocol, checksum computation, encryption methods



RIJNDAEL S-BOX FOR AES

Firmware Analysis Result

- Cryptographic scheme and key generation

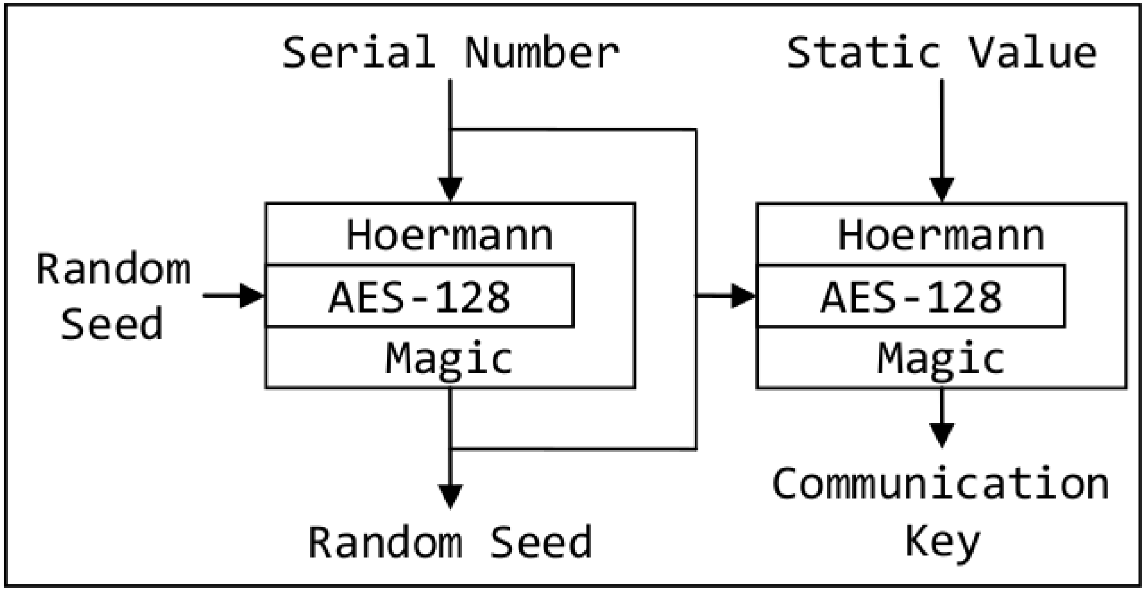


GENERATION OF COMMUNICATION KEYS

System Analysis

Field	Length (Byte)	Comment
Constant	1	0x70, 0x50
Serial Number	4	unique to device
Encrypted Data	16	adapted AES-128
Checksum	1	CRC

RF Frame (i.e., „open/close door“)



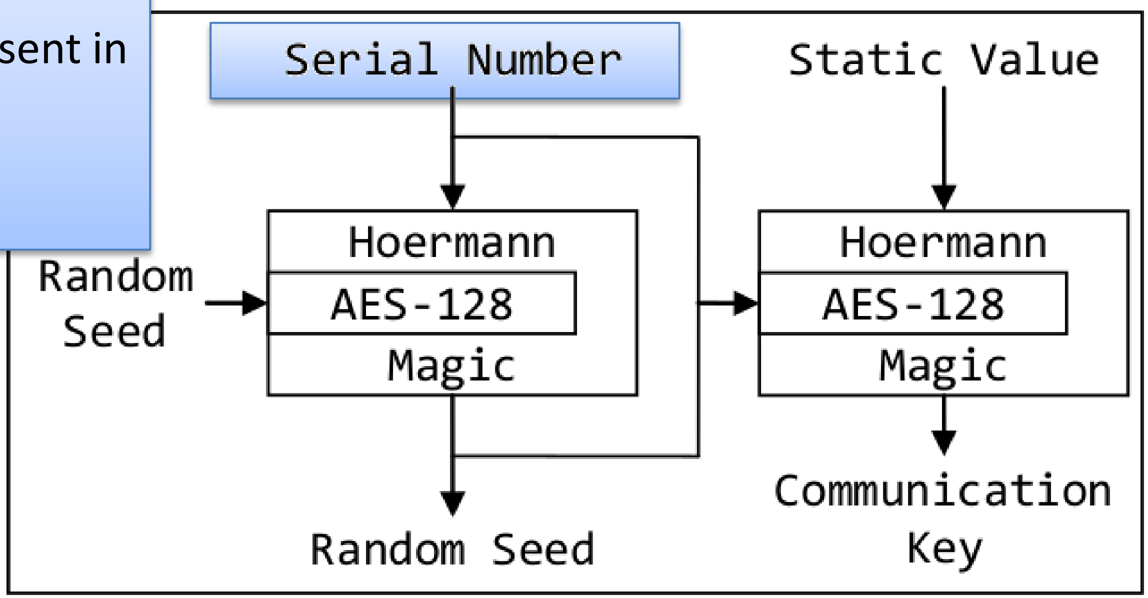
GENERATION OF COMMUNICATION KEYS

System Analysis

Field	Length (Byte)	Comment
Constant	1	0x70, 0x50
Serial Number	4	unique to device
Encrypted Data	16	adapted AES-128
Checksum	1	CRC

RF Frame (i.e., „open/close door“)

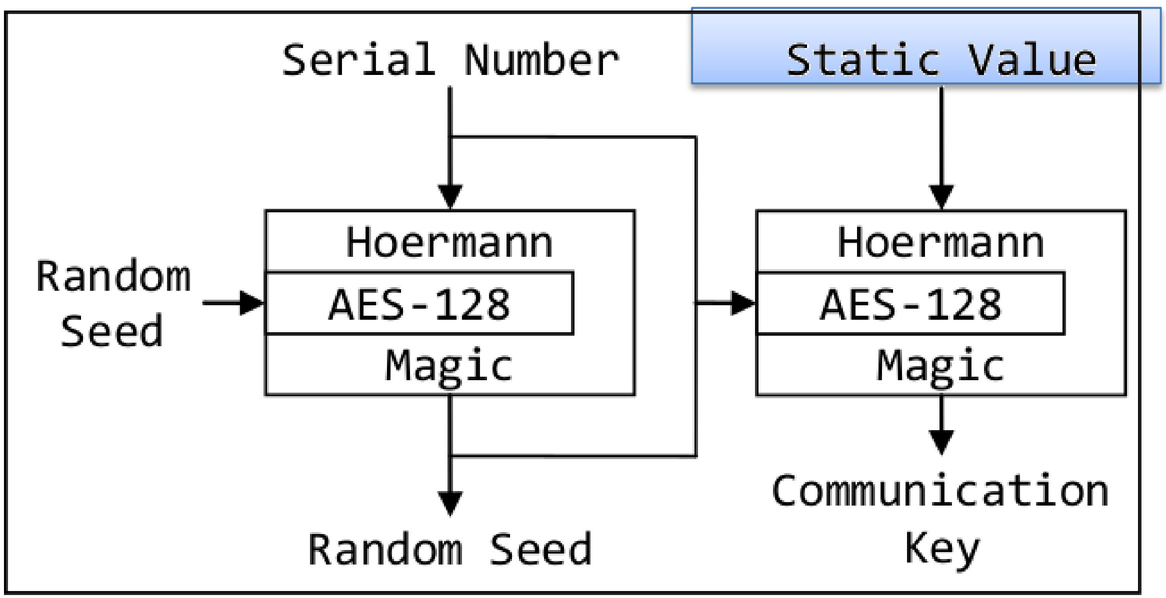
Serial number is sent in the clear



GENERATION OF COMMUNICATION KEYS

System Analysis

The static value is hardcoded in the firmware

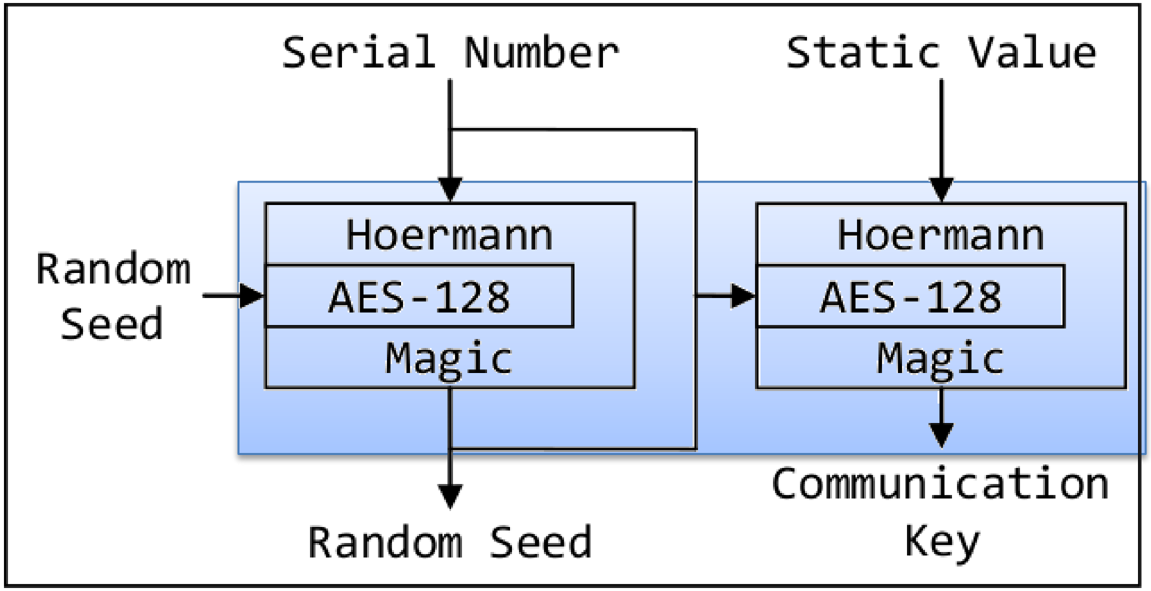


GENERATION OF COMMUNICATION KEYS

System Analysis

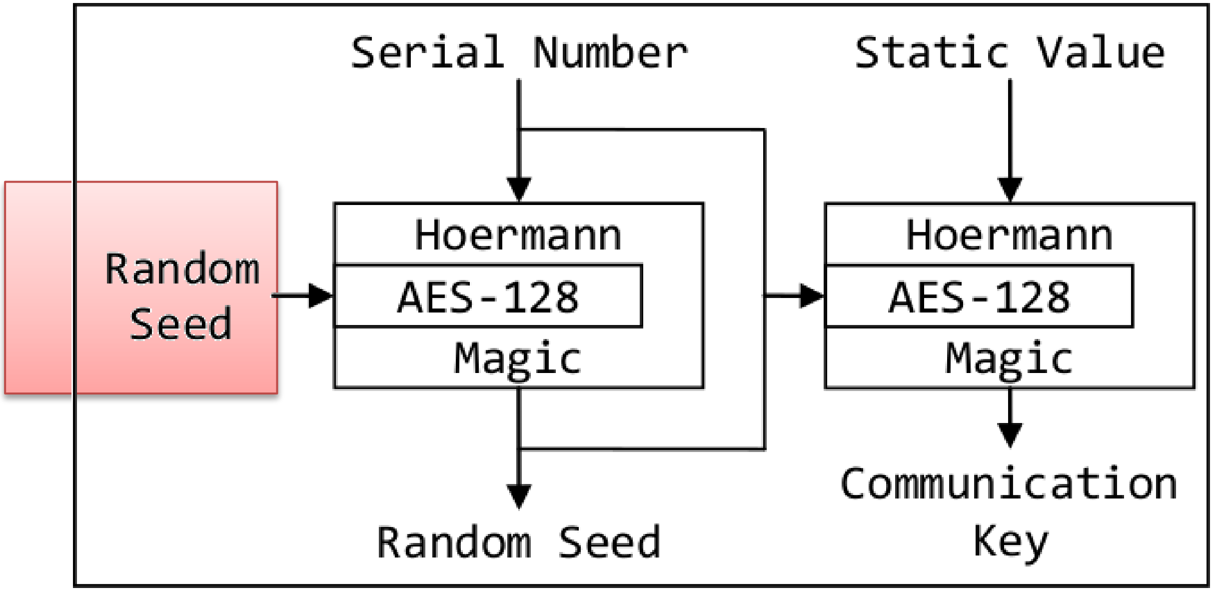
The encryption scheme is implemented in the firmware.

Kerckhoff‘s principle



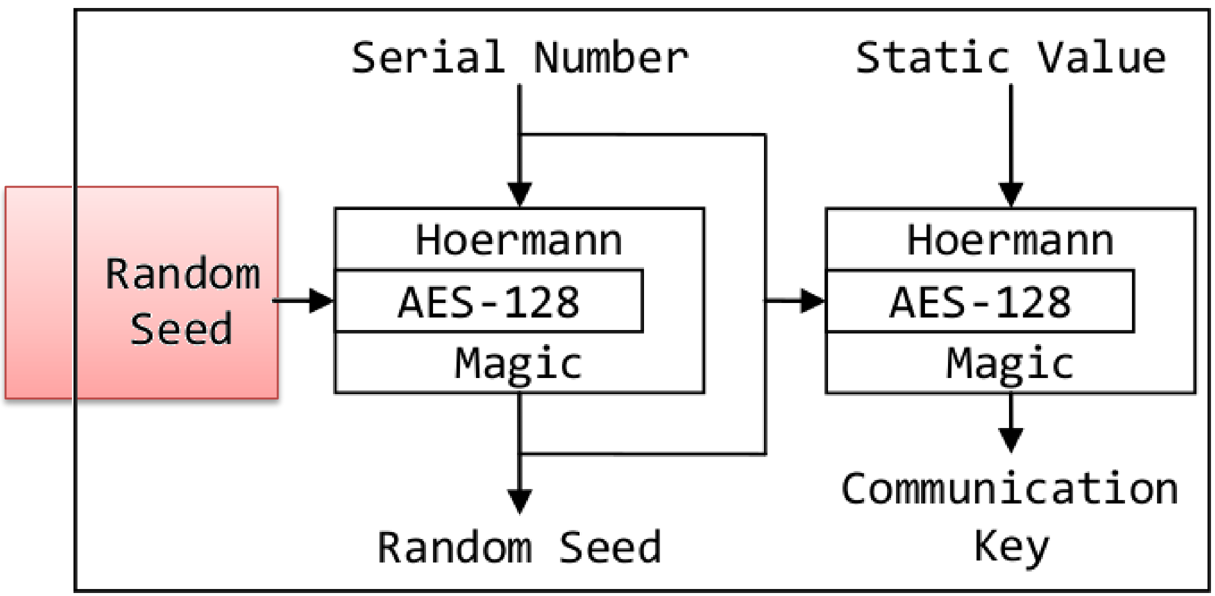
System Analysis

Initial random seed was identical on all our devices



System Analysis

Initial random seed was identical on all our devices → **Vulnerability**



GENERATION OF COMMUNICATION KEYS

How does the Attack Work ?

- Low-cost SDR such as the CCC rad1o or HackRF

- The CCC rad1o was available for free at the CCC Camp 2015 (“conference badge”)

How does the Attack Work ?

- Step 1: Record the RF transmission from a BiSecur hand transmitter

How does the Attack Work ?

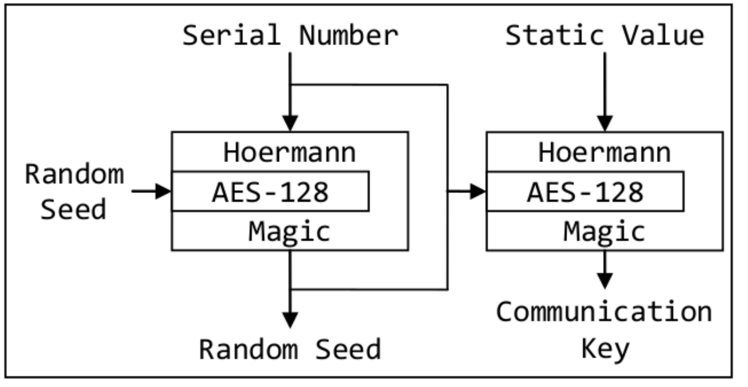
- Step 1: Record the RF transmission from a BiSecur hand transmitter

Field	Length (Byte)	Comment
Constant	1	0x70, 0x50
Serial Number	4	unique to device
Encrypted Data	16	adapted AES-128
Checksum	1	CRC

How does the Attack Work ?

- Step 2: Use the information obtained from an arbitrary hand transmitter* and the RF frame

- We know:
  - Encryption scheme (AES + ”magic”)
  - Static value
  - Initial random seed
  - Structure of a decrypted message
  - Serial number (from recorded RF frame)
  - Encrypted payload (from recorded RF frame)



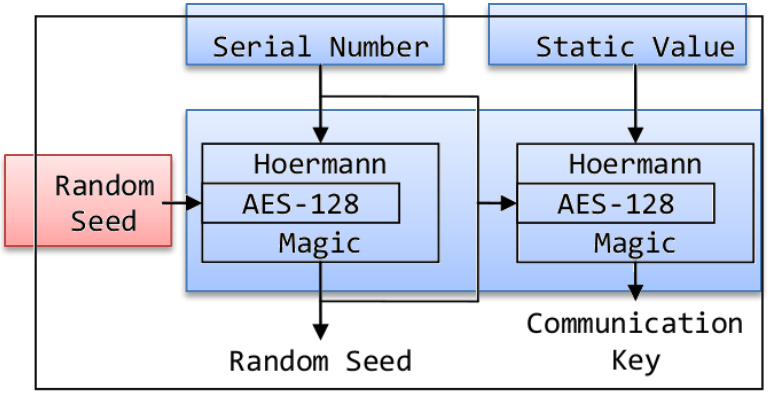
GENERATION OF COMMUNICATION KEYS

* to protect BiSecur customers, we do not disclose this information

How does the Attack Work ?

- Step 2: Use the information obtained from an arbitrary hand transmitter and the RF frame

- We know:
  - Encryption scheme (AES + ”magic”)
  - Static value
  - Initial random seed
  - Structure of a decrypted message
  - Serial number (from recorded RF frame)
  - Encrypted payload (from recorded RF frame)

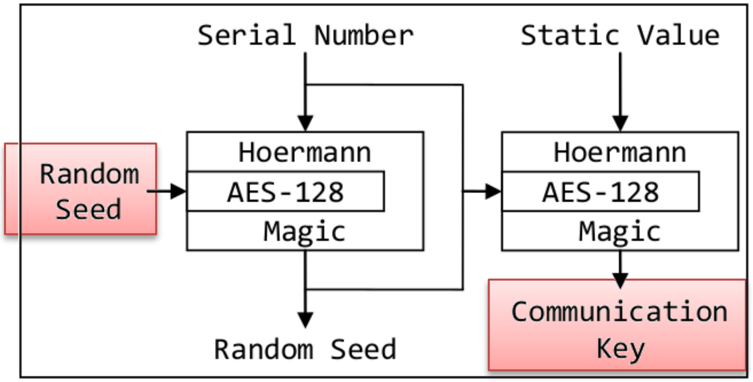


GENERATION OF COMMUNICATION KEYS

How does the Attack Work ?

- Step 3: Compute Communication Key candidate

- Candidate is correct if it has the expected plaintext structure

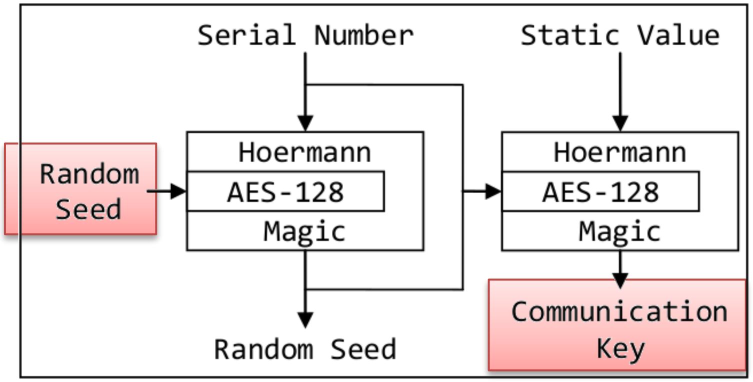


GENERATION OF COMMUNICATION KEYS

- Otherwise repeat (this is the case if the user has manually generated a new key)

How does the Attack Work ?

- Step 4: Obtain the current counter value from the decrypted message



GENERATION OF COMMUNICATION KEYS

How does the Attack Work ?

- Step 5: Increase the counter value by one, encrypt the message with the obtained Communication Key

How does the Attack Work ?

- Step 6: Transmit RF frame, door should open

- Proof-of-Concept Demo

Impact Assessment

- Observation: serial numbers of same model hand transmitters bought at the same time were close to each other
- Assumption: Sequential serial numbers --> probably millions of devices in the field
- Not sure if our guess is correct !

0x043F3D68	71.253.352	HSE1 glossy (old working)
0x043F3D78	71.253.368	HSE1 glossy (old defect)
0x046972A0	74.019.488	HSE1 matte (new)
0x0404F1AB	67.432.875	HSE2 black (new)
0x046A2489	74.065.033	HSE2 white (new)
0x0462DD51	73,588,049	HSE2 in set (new)
0x0462DD5A	73.588.058	HSE2 in set (new)

How can the Vulnerability be Fixed ?

- Each hand transmitter needs to have its individual random seed value (80-bit)

- Since the random seed is no longer shared between all hand transmitters, an attacker can no longer compute the communication key without brute-forcing the 80-bit random seed

Responsible Disclosure

[https://www.siteground.com/blog/responsible-disclosure]

Responsible Disclosure

- 4.10.2017 - Involving the Austrian national CERT team as coordinator, we reported the security vulnerability including a detailed advisory and a suggested security fix so that the manufacturer can fix the issue

- 31.10.2017 – Confirmation from CERT that the manufacturer received and understood the security problem

Responsible Disclosure

- [...]: various e-mails and phone calls
- End of Nov. 2017: Meeting with manufacturer: we presented the vulnerability and the suggested security fix
- Dec. 2017: Security fix implemented and in testing phase

Conclusion

- We presented a viable methodology to analyze wireless RF systems with microcontrollers
- We believe that independent security audits are an essential tool to achieve a high level of security in a product
- When it comes to hardware security, it is good to have a Hardware Security lab at hand :-)
- We followed a responsible disclosure process and supported the manufacturer in understanding and fixing the vulnerability
- We will publish the security advisory after this talk
  (CVE ID: CVE-2017-17910, CVSSv2: 9.7)

Thank you for your Attention !

Trustworks KG

Rienoesslgasse 14/17
1040 Wien

Visit our homepage or contact us at:

Web: https://www.trustworks.at
E-Mail: office <at> trustworks.at