Ten years of studies on the security of connected objects: a wrap-up

Conférence C&ESAR, Cyber Week,
Rennes, 21/11/2023

Romain CAYRE, Vincent NICOMETTE
romain.cayre@eurecom.fr, vincent.nicomette@laas.fr

RELATED PHDs

- 2012 – 2015 : Yann Bachy, LAAS-CNRS, Thalès
    - *« Sécurité des équipements grand public connectés à Internet : évaluation des liens de communication »*
- 2016 – 2020 : Jonathan Roux, LAAS-CNRS
    - *« Détection d'intrusion dans des environnements connectés sans fil par l'analyse de l'activité radio »*
- 2020 – 2023 : Florent Galtier, LAAS-CNRS
    - *« Sécurité des réseaux sans fil à courte et longue portée basée sur des mécanismes de monitoring de la couche physique »*
- 2020 – 2023 : Romain Cayre, LAAS-CNRS, Airbus Protect
    - *« Offensive and defensive approaches for wireless communication protocols security in IoT »*

Supervisors : Eric Alata, Guillaume Auriol, Mohamed Kaâniche, Vincent Nicomette

CONTEXT OF THE RESEARCH WORK

Lack of knowledge of connected object manufacturers

Embedded systems with limited resources

CONNECTED OBJECTS

Rapid expansion

Interaction with the physical world

CONTEXT OF THE RESEARCH WORK

CONNECTED OBJECTS

- Embedded systems with limited resources
- Lack of knowledge of connected object manufacturers
- Rapid expansion
- Interaction with the physical world

WIRELESS COMMUNICATION PROTOCOLS

- Heterogeneous technologies
- Peer to peer communications
- Mobile equipments
- Co-existence in same environments

RESEARCH OBJECTIVES AND ISSUES

- How can we identify and assess new threats linked to the deployment of wireless communication protocols in the context of the Internet of Things ?
- How can we automate the audit of such communication protocols ?
- How to design appropriate and innovative intrusion prevention and detection systems ?

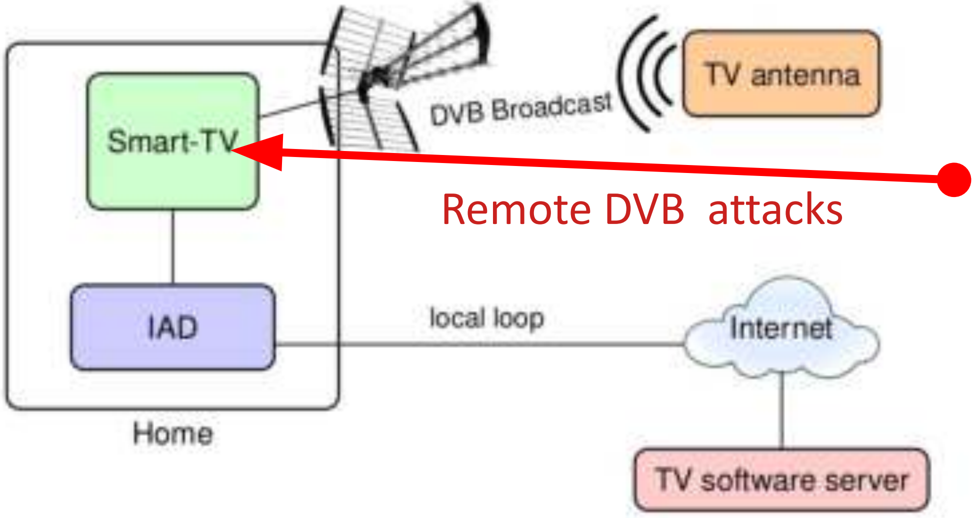
OUTLINE

- **DVB vulnerability : Smart TVs**
- **Mirage : a framework for auditing IoT communication protocols security**
- **Cross-protocols vulnerabilities : WazaBee**
- **BLE vulnerability : InjectaBLE**
- **Bad design : Padlock, Keyboards and mices vulnerabilities**
- **Conclusion**

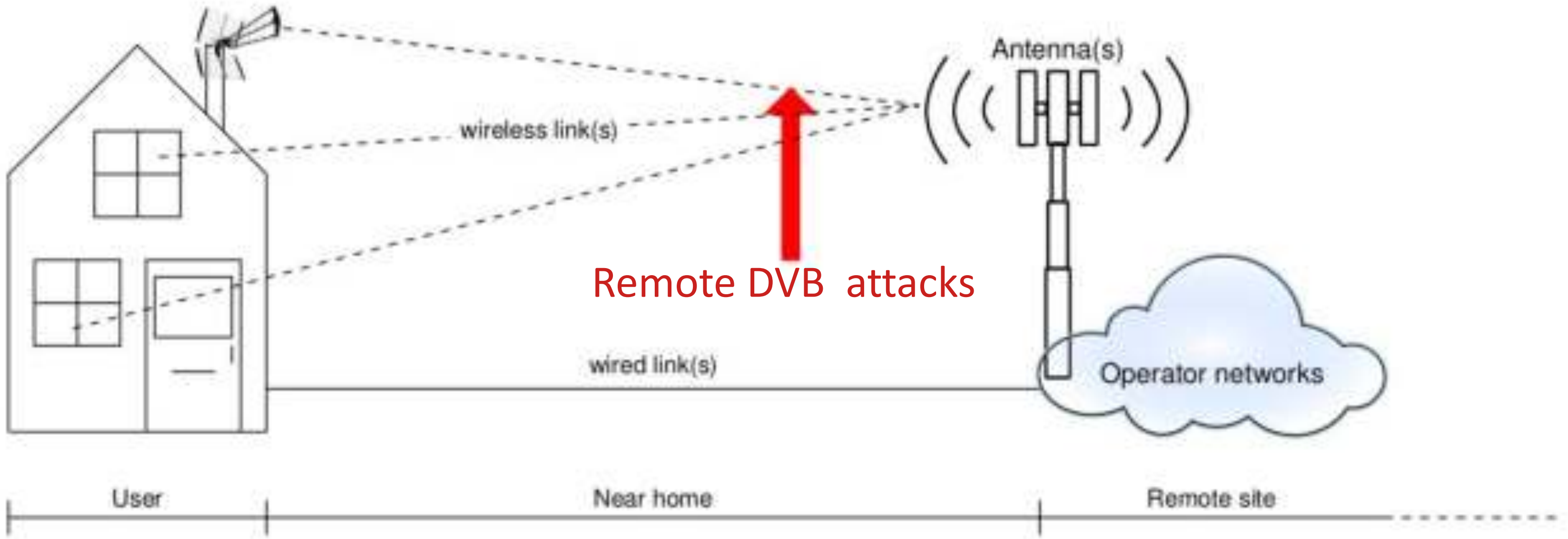
DVB Vulnerability : Smarts TVs

RESEARCH OBJECTIVES AND ISSUES

- Smarts TV are connected to the home network and to DVB broadcaster
- Is it possible to build a fake DVB broadcaster and inject attacks that could propagate in the home network ?



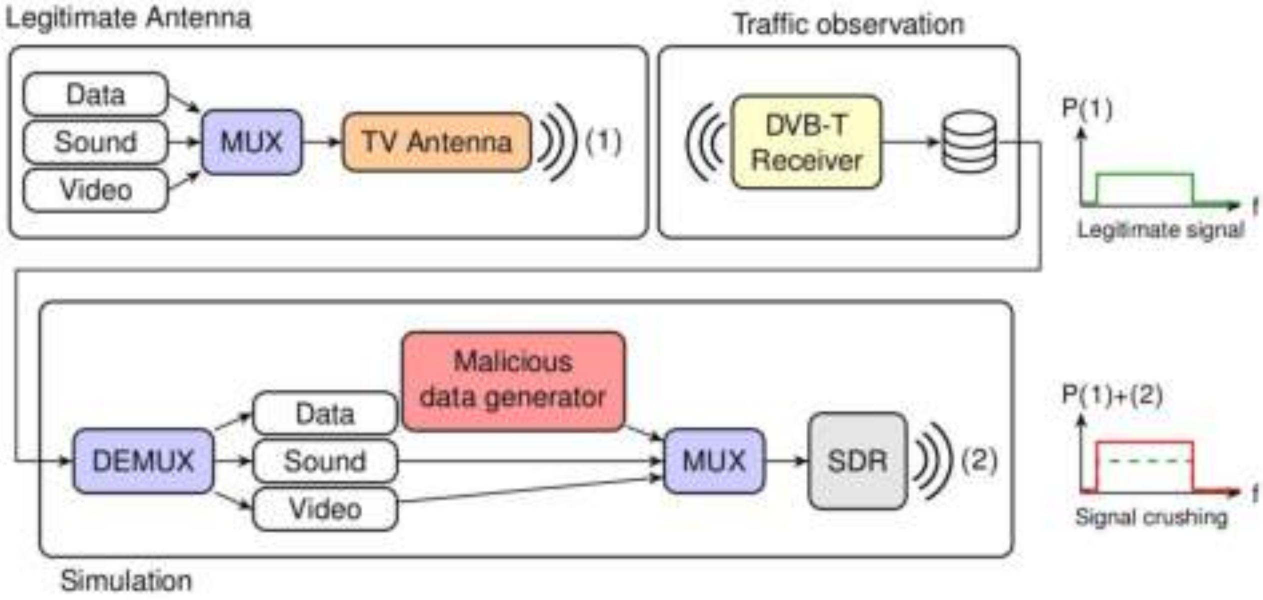
SMART TV – DVB



Antenna(s)
wireless link(s)
Remote DVB attacks
wired link(s)
Operator networks
User
Near home
Remote site

SMART TV – Building a fake broadcaster

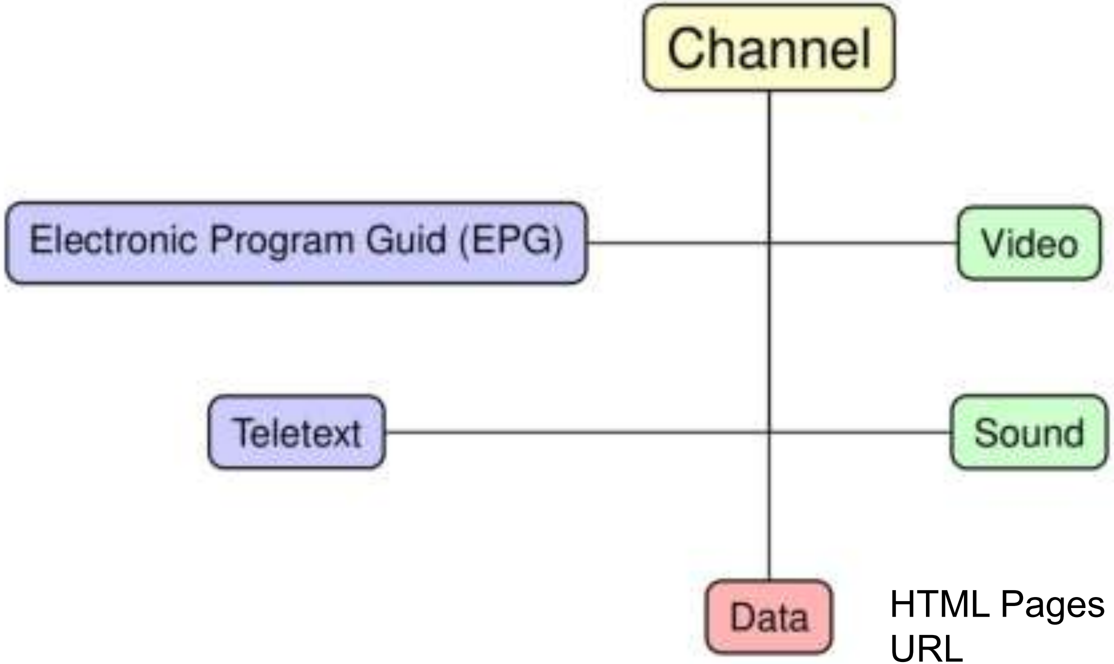
SMART TV – Building a fake broadcaster



(1) = Legitimate signal

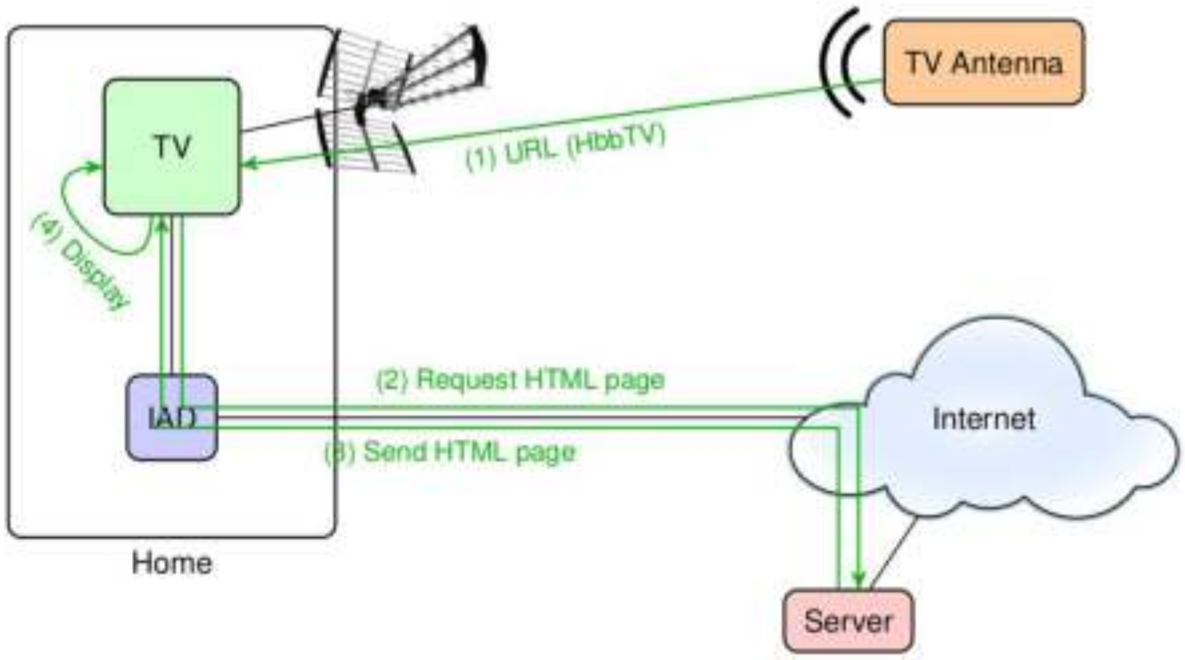
(2) = Malicious signal

SMART TV – Content of a DVB flow

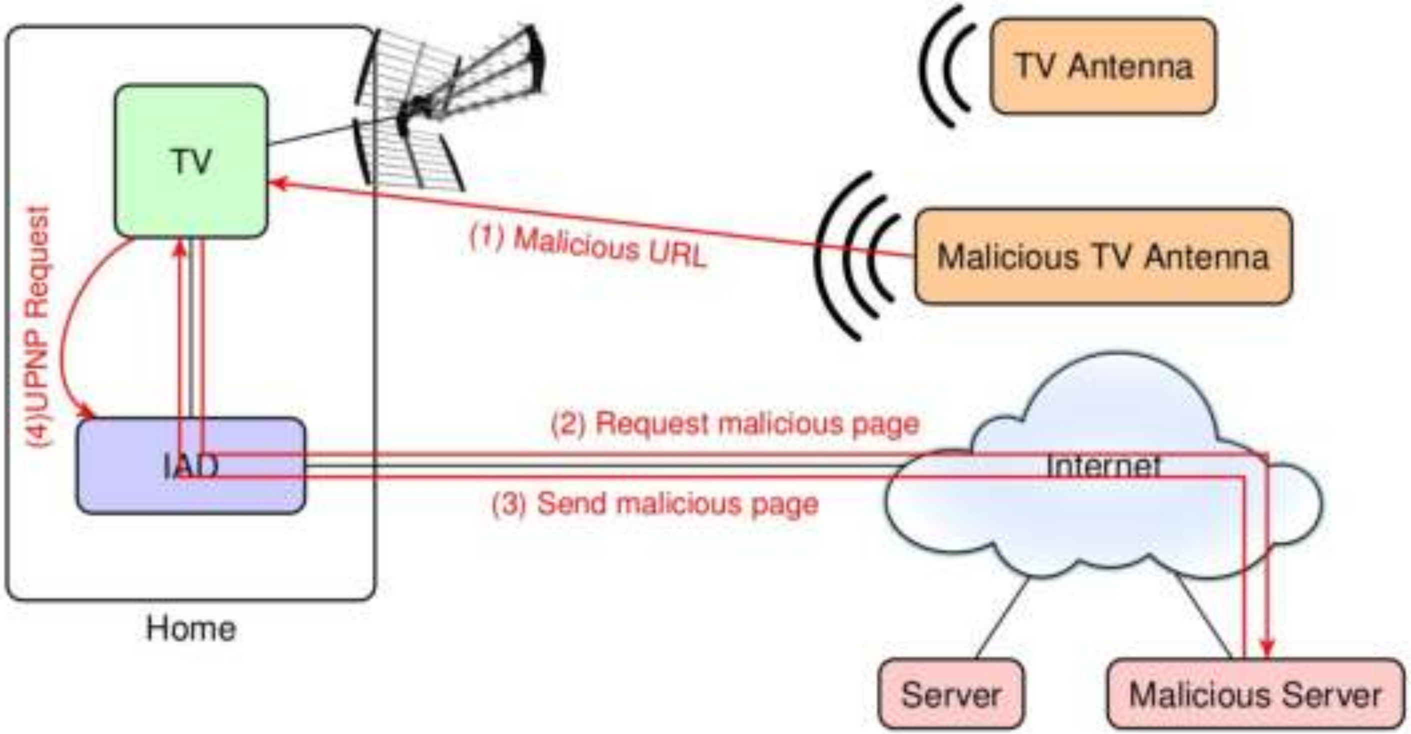


HTML Pages
URL

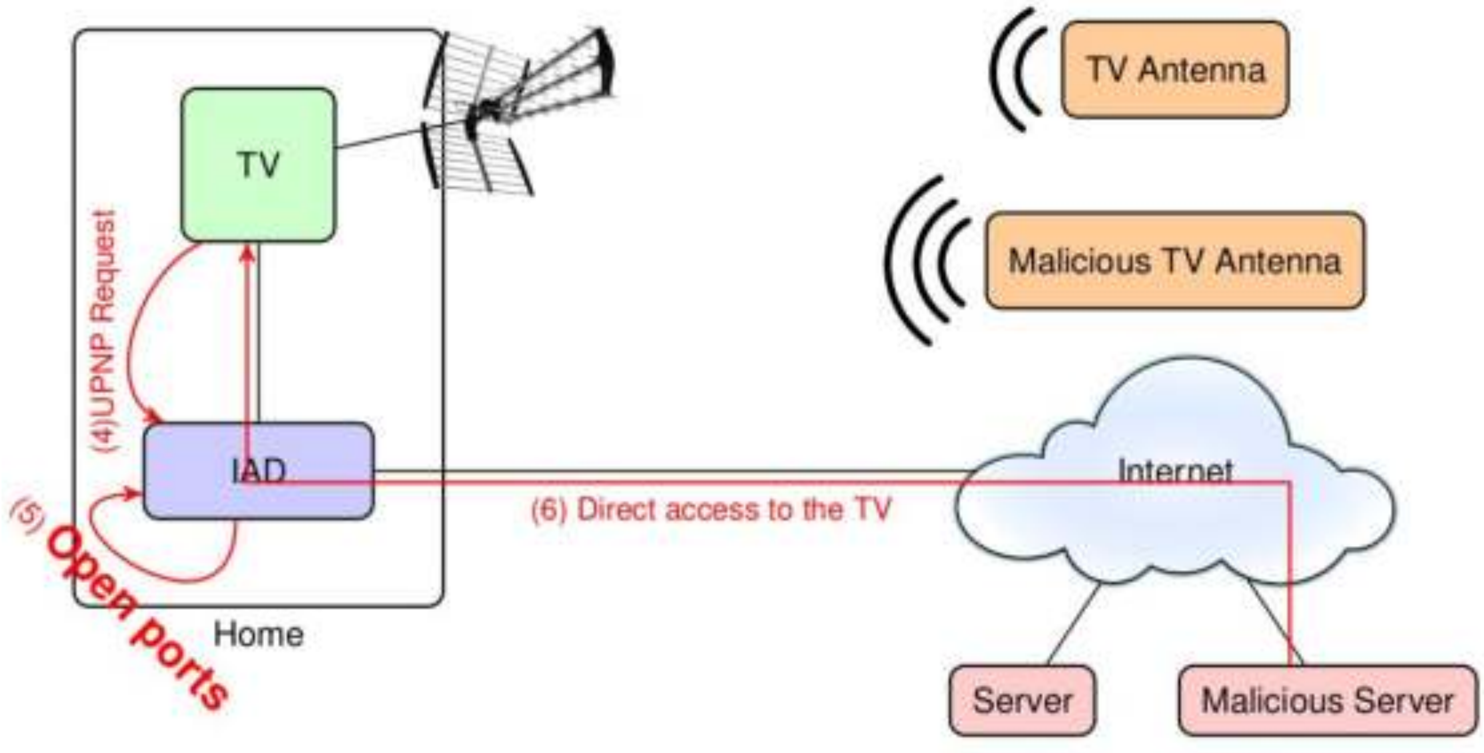
SMART TV – Legitimate scenario



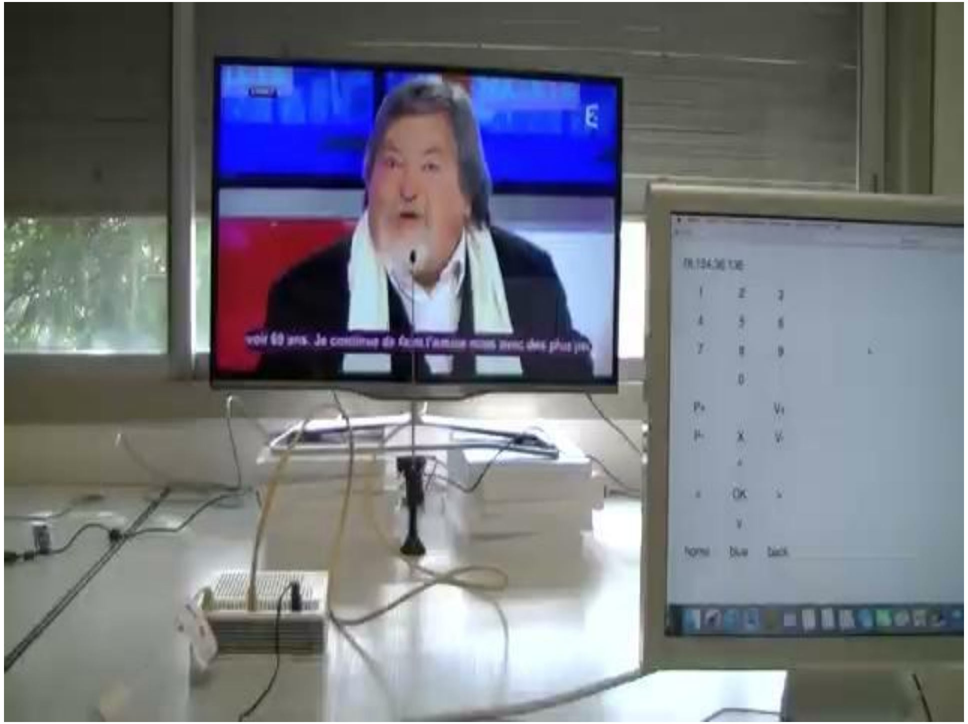
SMART TV – Malicious scenario



SMART TV – Malicious scenario



SMART TV – Demo



SMART TV – Conclusion

- Some quite obvious vulnerabilities that let us feel that security is not seriously considered
- A lack of tool to automate the audit

- Publications
  - [SSTIC 2014, 2015], [DSN 2015], [JICV 2015], [EDCC 2015]

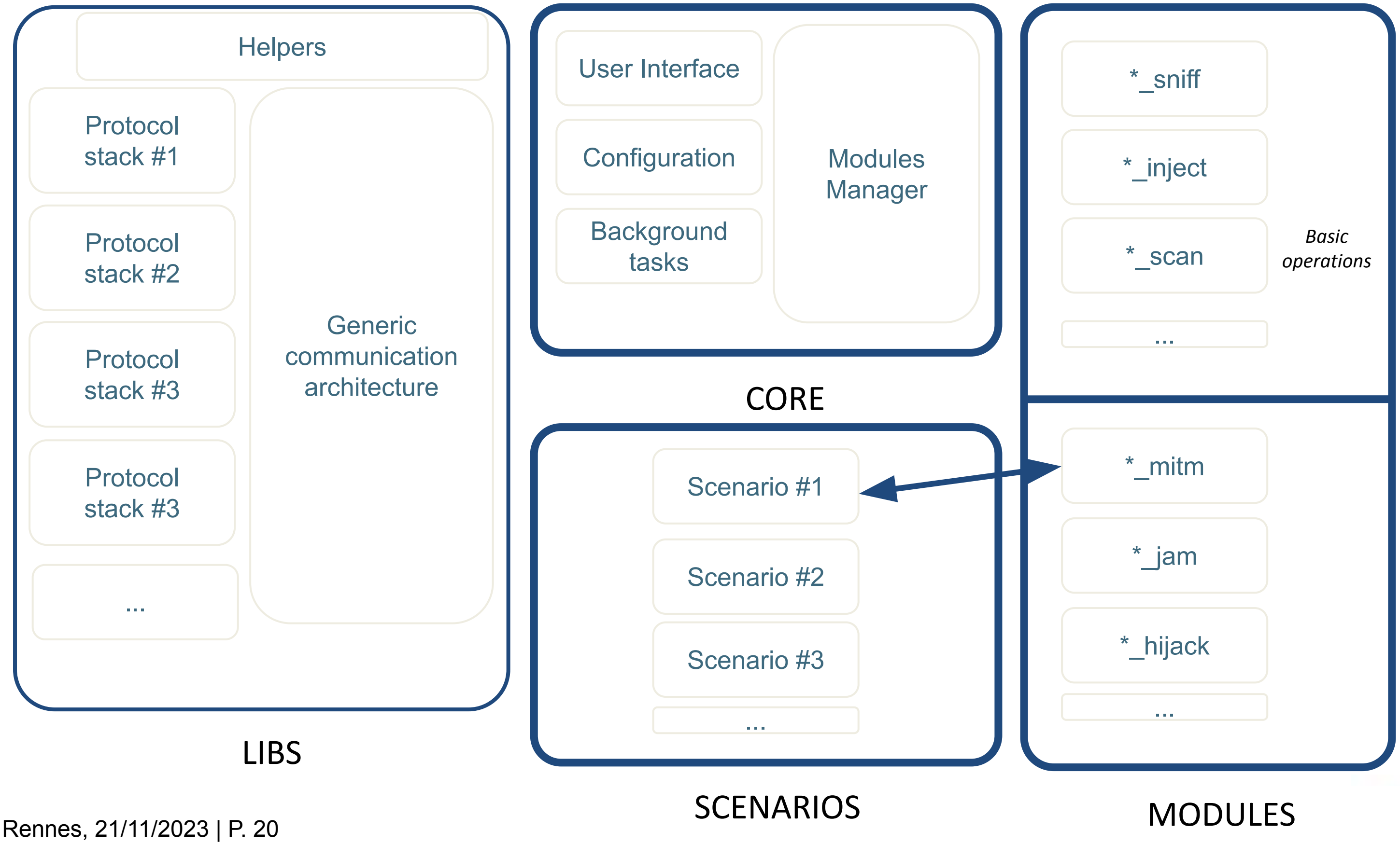
Mirage : a framework for auditing IoT communication protocols security

RESEARCH OBJECTIVES AND ISSUES

- How to generate some realistic attack scenarios ?
- How to simulate an attacker’s behavior ?
- How to conduct reproducible and efficient security audits targeting IoT devices in order to discover vulnerabilities ?
- How to evaluate the efficiency of an intrusion detection system ?

→ **Mirage framework**

MIRAGE – Global architecture



MIRAGE – Complex tasks

- A security analyst has to :
  - Execute complex attack workflows, involving multiple operations
  - Detect some specific events and be able to execute a custom code when they occur

- Mirage provides
  - A chaining operator

    ```
    $ mirage ’ble_scan | ble_adv’
    ```

  - A mechanism of callbacks (« scenarios »)

MIRAGE – Conclusion

- Successful approach: Mirage was decisive to identify a lot of vulnerabilities (some of them are described in the next parts of this presentation)
- Really useful capability: perform low-level attacks

- Publications
    - [ISSRE 2019], [SSTIC2019]

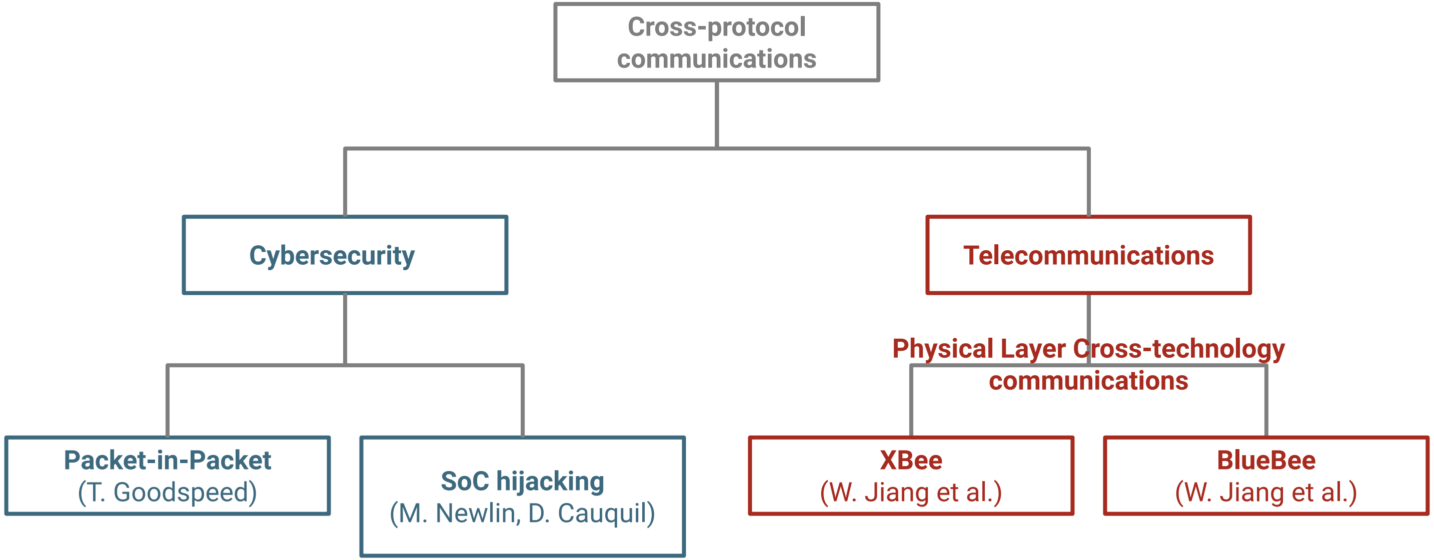
Cross-protocols vulnerabilities : WazaBee

Unexplored attack strategy: successive compromise of equipment by "pivoting", exploiting different wireless protocols.

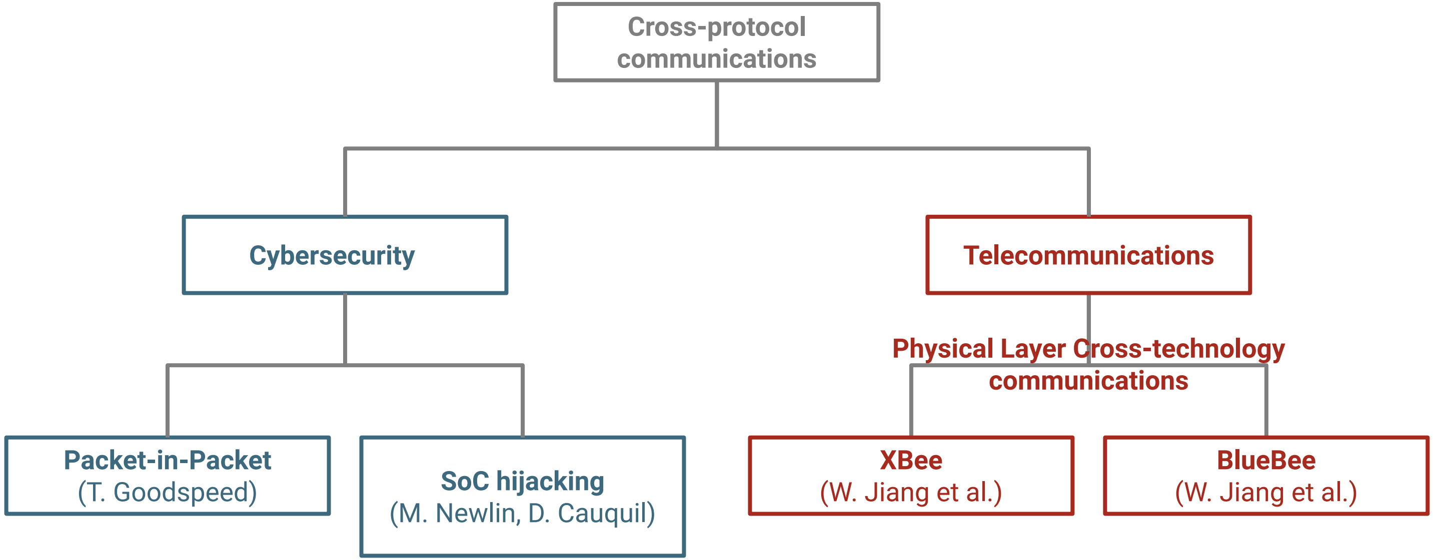
- Most wireless protocols are incompatible (modulations)
- Coexistence in the same physical environments, shared resources (2.4GHz band)

Example: compromise of an employee's BLE connected watch on public transport, then use of the watch as an intermediary to attack an 802.15.4 sensor network.

WAZABEE – State of the art



WAZABEE – State of the art



Limitations:

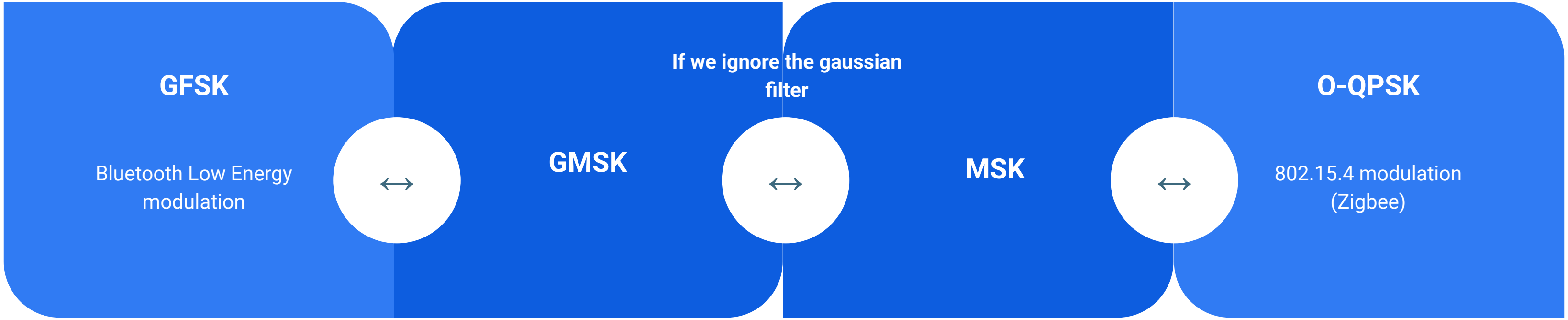
- **Compatible modulations (GFSK),**
- **Specific chips, difficult to generalise**

Limitations:

- **Requires the cooperation of end devices**

WAZABEE - Overview

- **Diverting BLE chips to build cross-protocol reception and emission primitives**
  - Bluetooth Low Energy 5.0 ⟶ 802.15.4 (ZigBee) - incompatible modulations
  - Independent of the cooperation of other devices
- Exploitation of the similarity of the modulations (GFSK ↔ O-QPSK) and modification of low-level control mechanisms (whitening, CRC, …).



- Implementation of the approach on BLE 5.0-compatible devices from different manufacturers and with different APIs (nRF52832 from Nordic SemiConductors and TI CC1352-R1 from Texas Instruments).

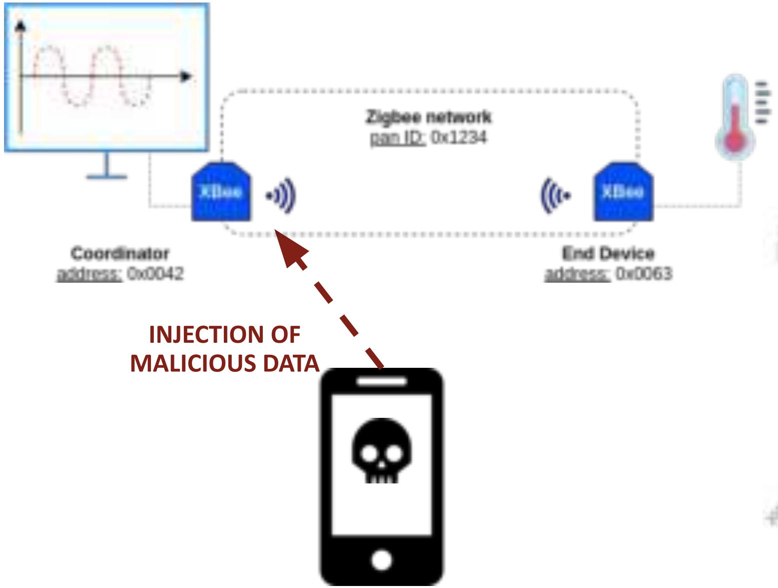
WAZABEE – Experimental Environment

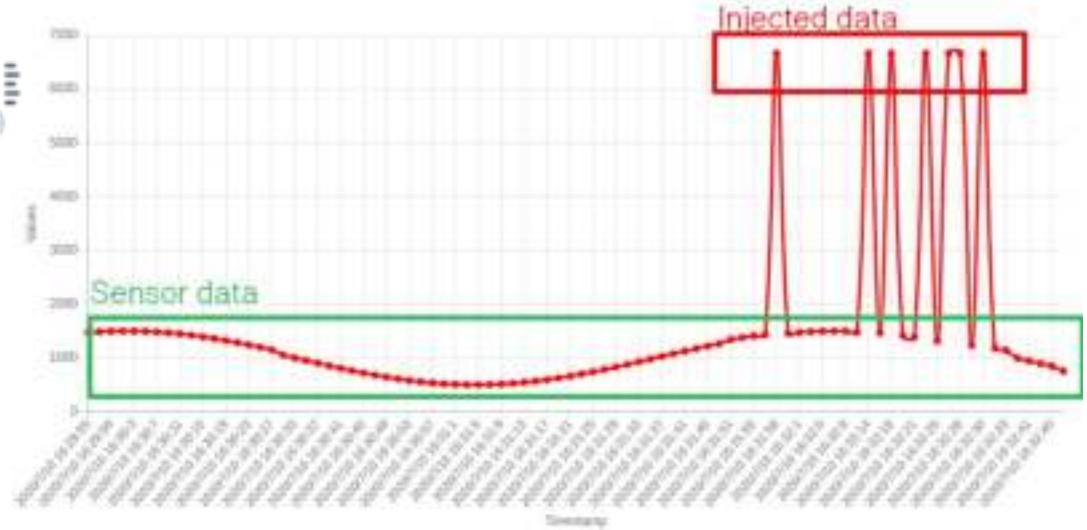
LAAS CNRS

EURECOM

WAZABEE - Scenario 1

- Injection of Zigbee frames from a non-rooted Android (OnePlus 6T),
- No access to low-level functionalities, use of the Android API (attacker with few privileges),
- Partial implementation (emission primitive)

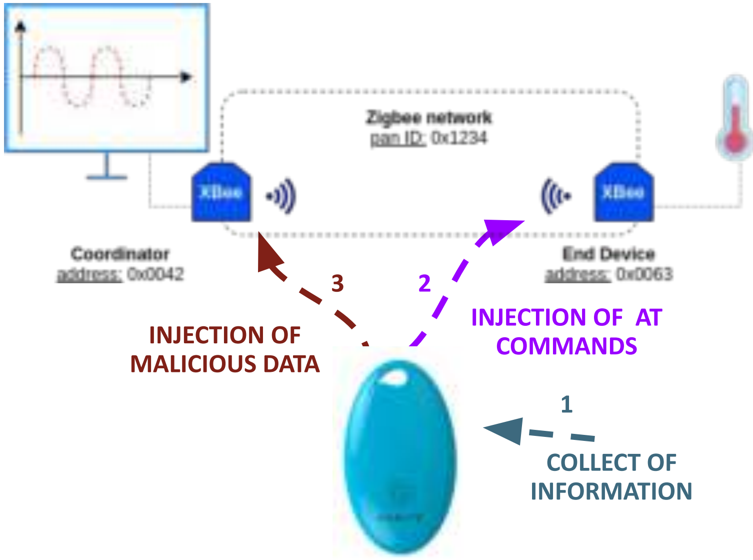




Injection result

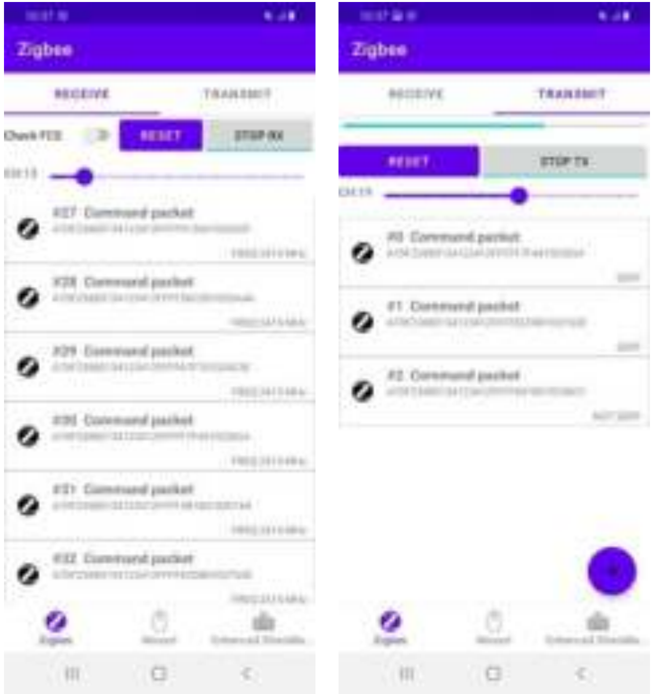
WAZABEE - Scenario 2

- Hijacking of a Zigbee sensor by a BLE connected keyfob (Gablys, nRF51822)
- Development of a malicious firmware, complete implementation (transmit and receive primitives)
- Remote AT command injection (sensor denial of service), followed by malicious data injection



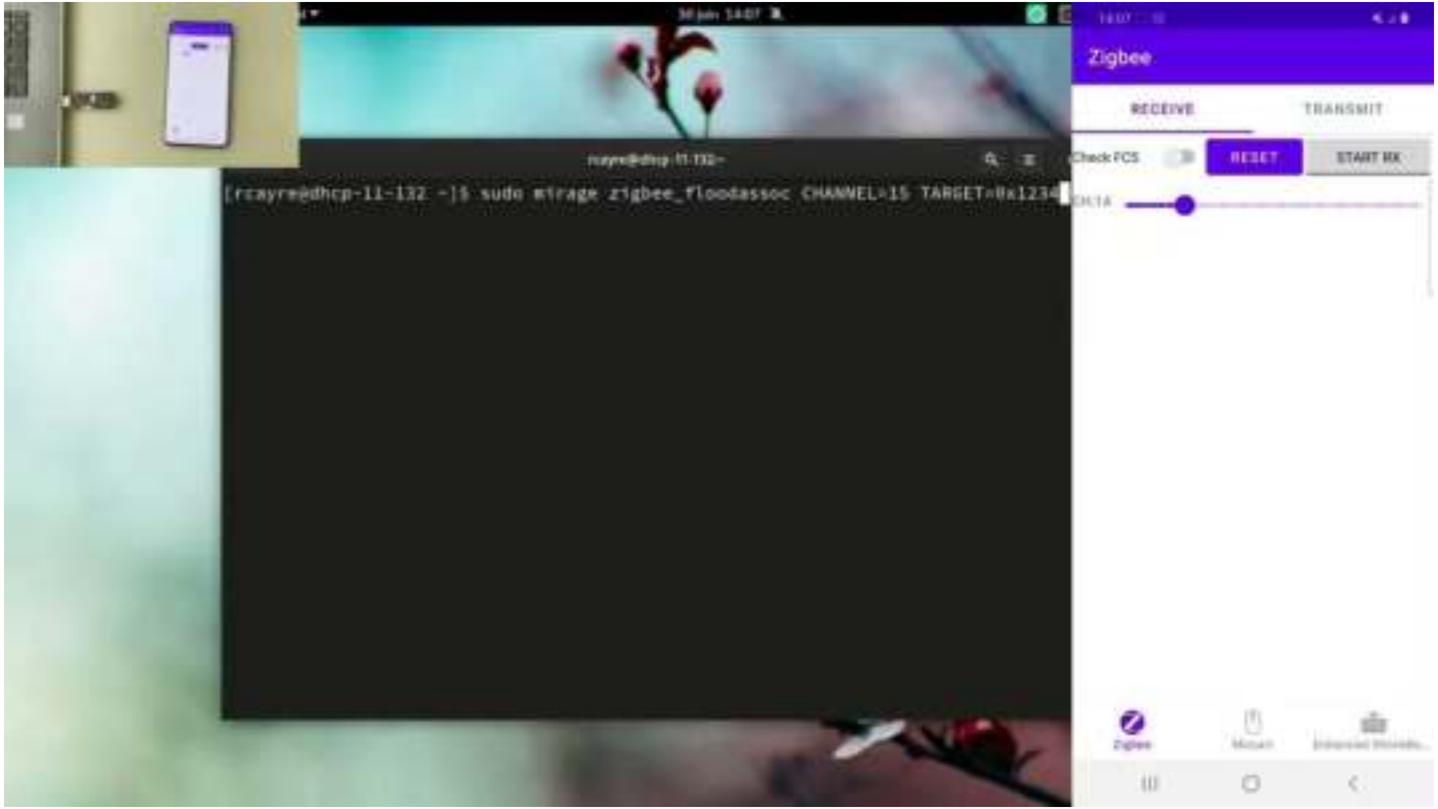
Hijacking result

WAZABEE - Smartphone Implementation



- Full implementation on Samsung Galaxy S20, access to low-level mechanisms (highly privileged attacker)
- Reverse engineering: black-box identification and hacking of software components, using InternalBlue framework (SEEMOO-Lab)
- Added support for proprietary wireless protocols (Enhanced ShockBurst, Mosart)
- Development of **RadioSploit**:
  - **patchs of the controller**
  - **Android application**

WAZABEE - Demo



WAZABEE - Conclusion

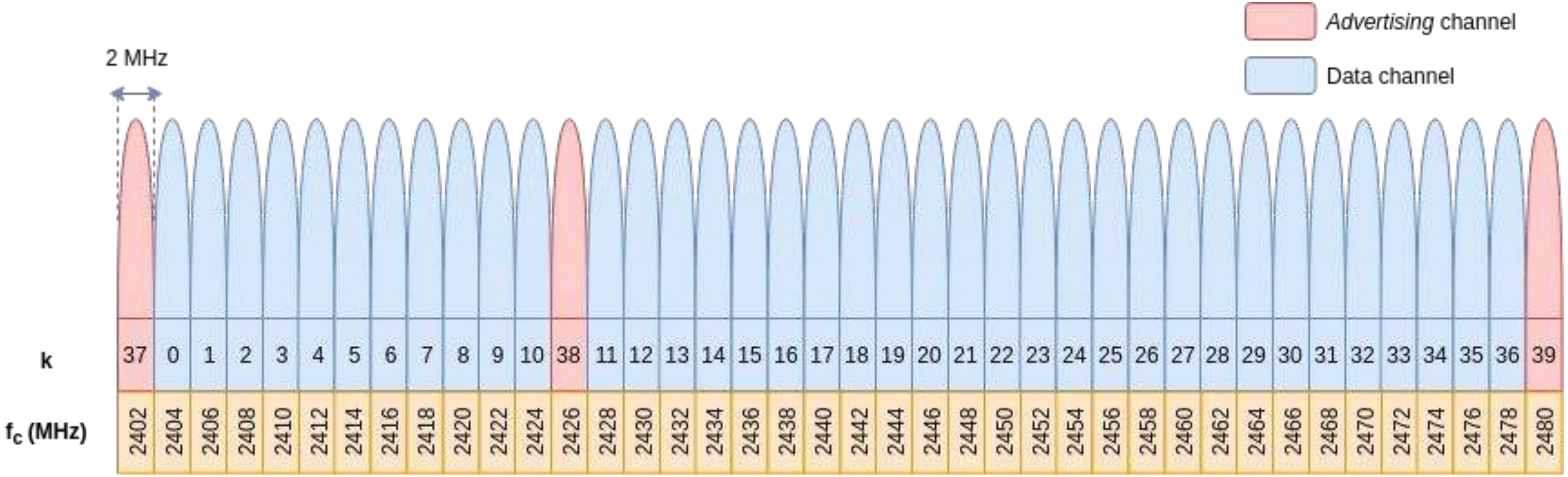
- A new cross-protocol pivoting attack exploiting similarities in incompatible modulations
  - very problematic in the IoT context (mobile devices, dynamic networks)
  - very hard to anticipate and mitigate

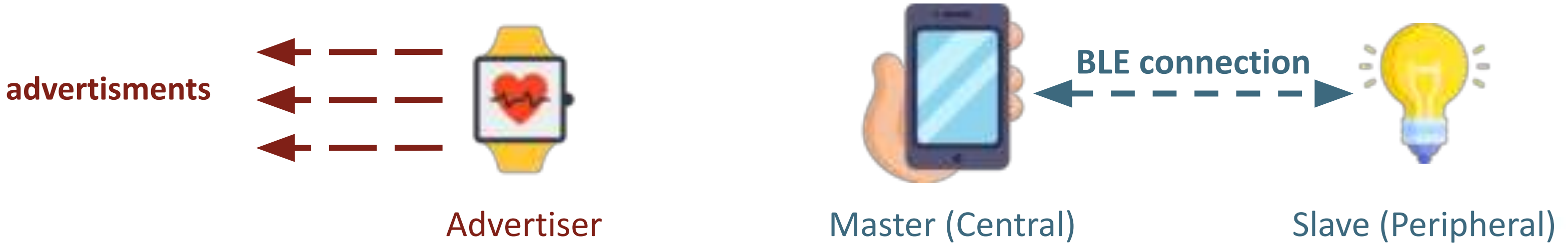
- We have the intuition that a lot of similar pivoting attacks exist
  - great : identify them
  - holy grail : formalize them

- Publications
  - [SSTIC 2020], [DSN 2021], [WiSec 2021] (“Best Poster award”)

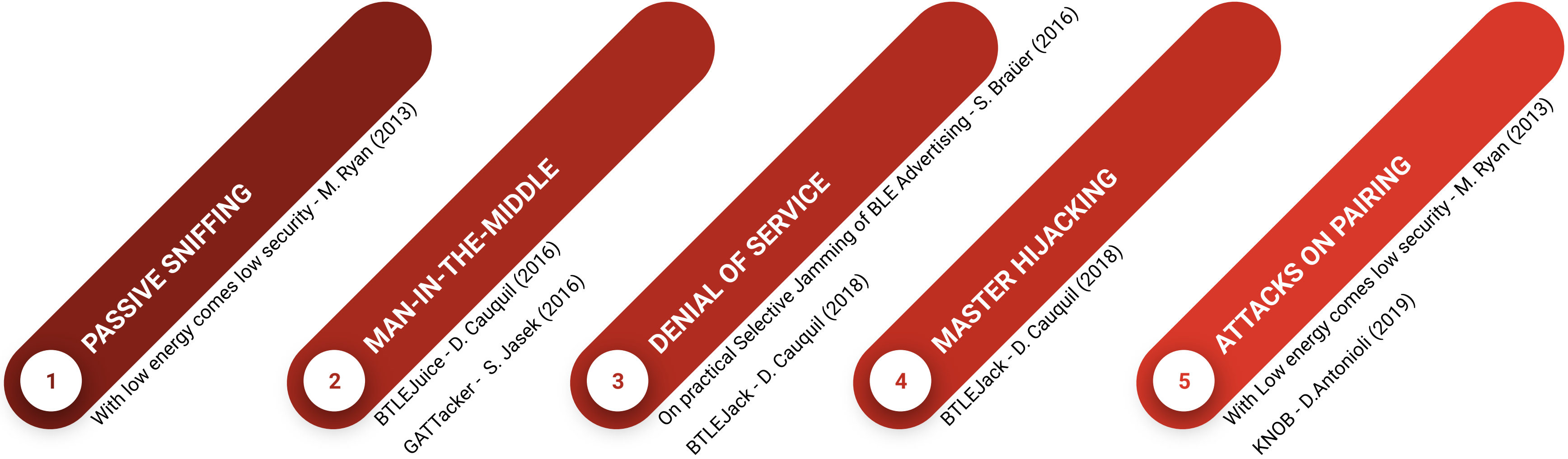
BLE vulnerability : InjectaBLE

INJECTABLE – Advertising and Connected modes

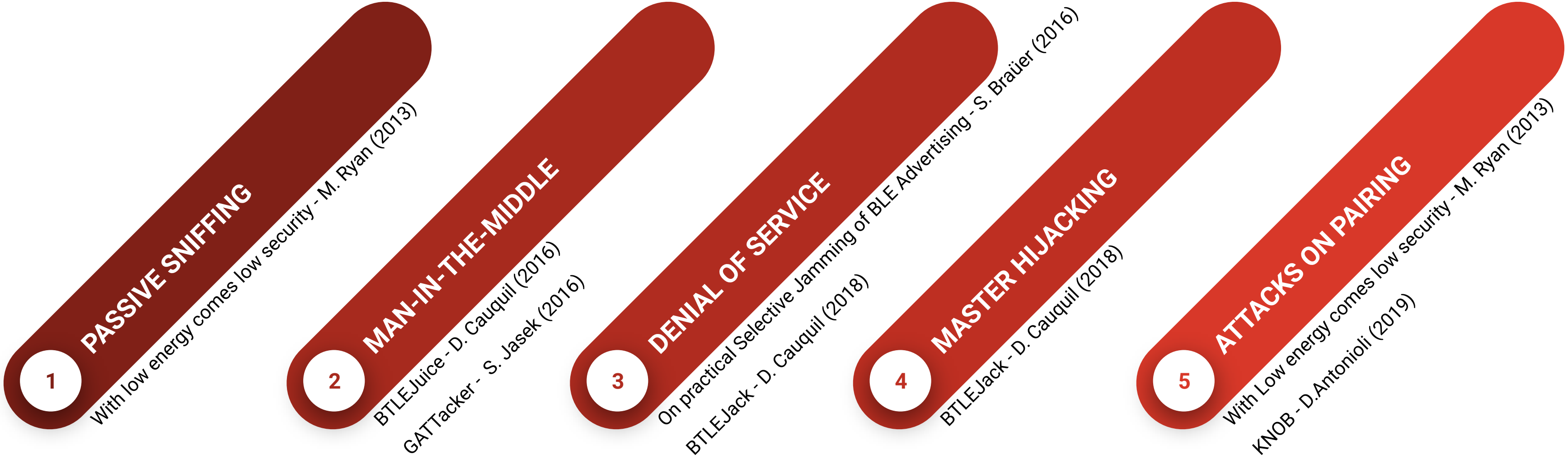




INJECTABLE – State of the art



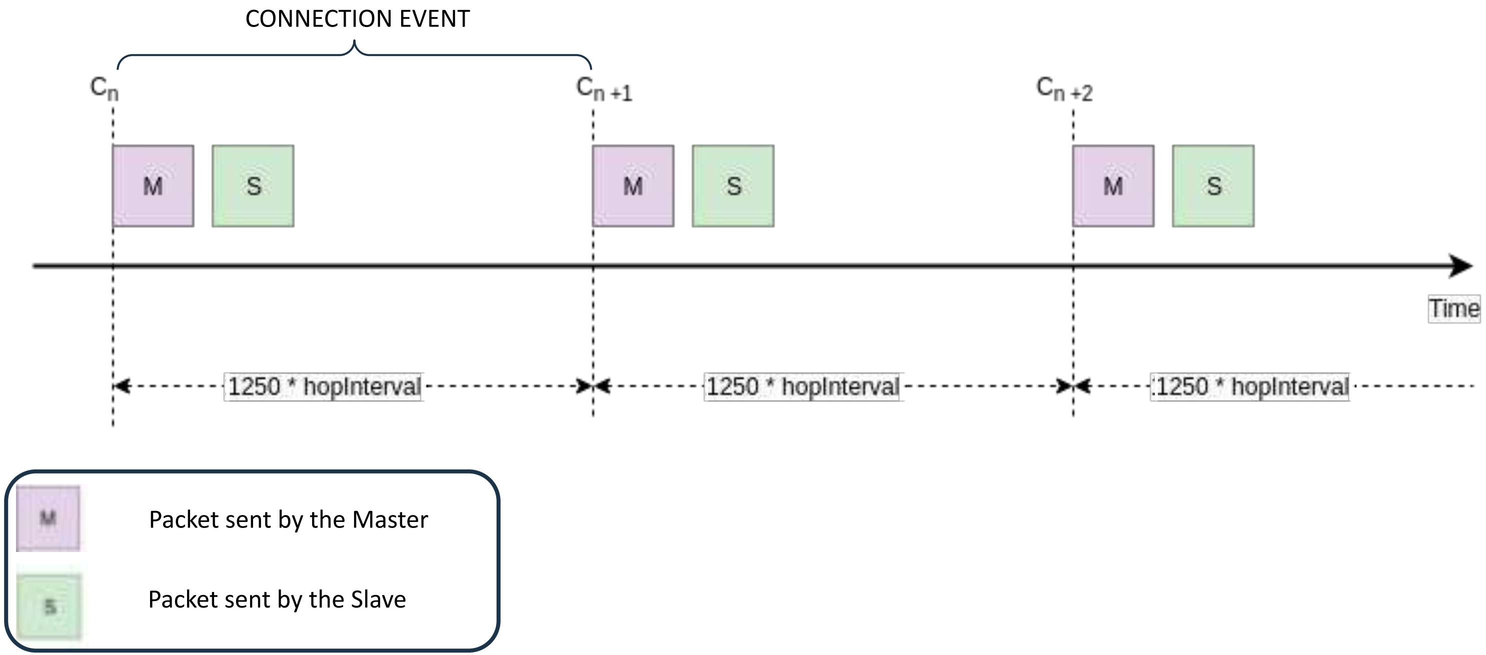
INJECTABLE – State of the art



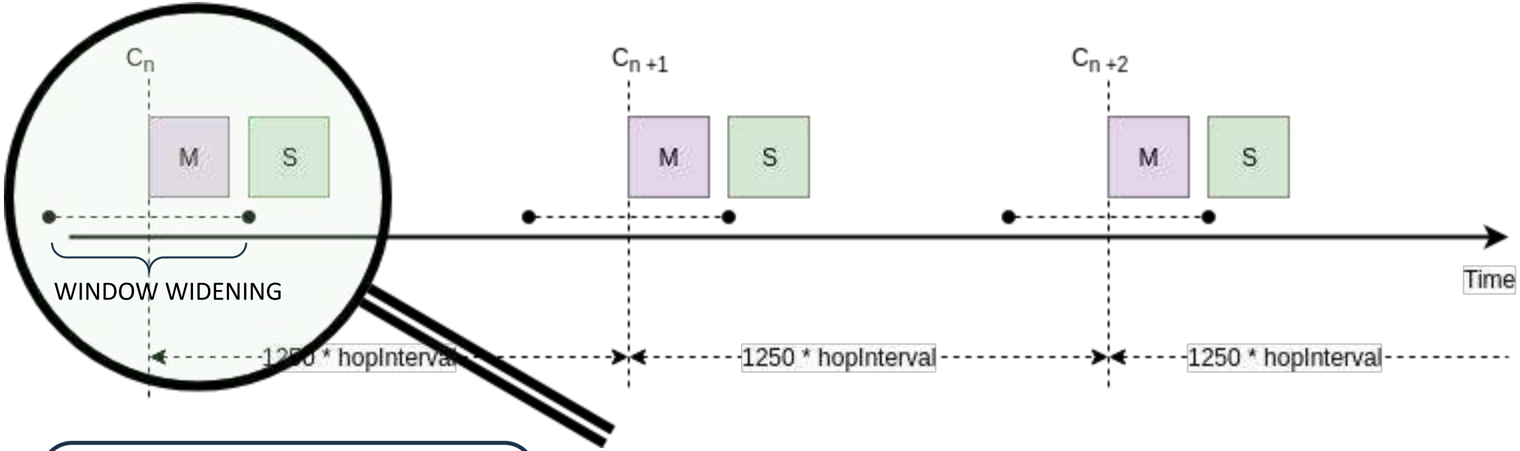
Our contributions

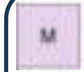
- Injecting malicious traffic into an established connection
- Performing Slave hijacking
- Performing a Man-in-the-Middle during an established connection

INJECTABLE – Connection event (ideal case)



INJECTABLE – Connection event (in reality)

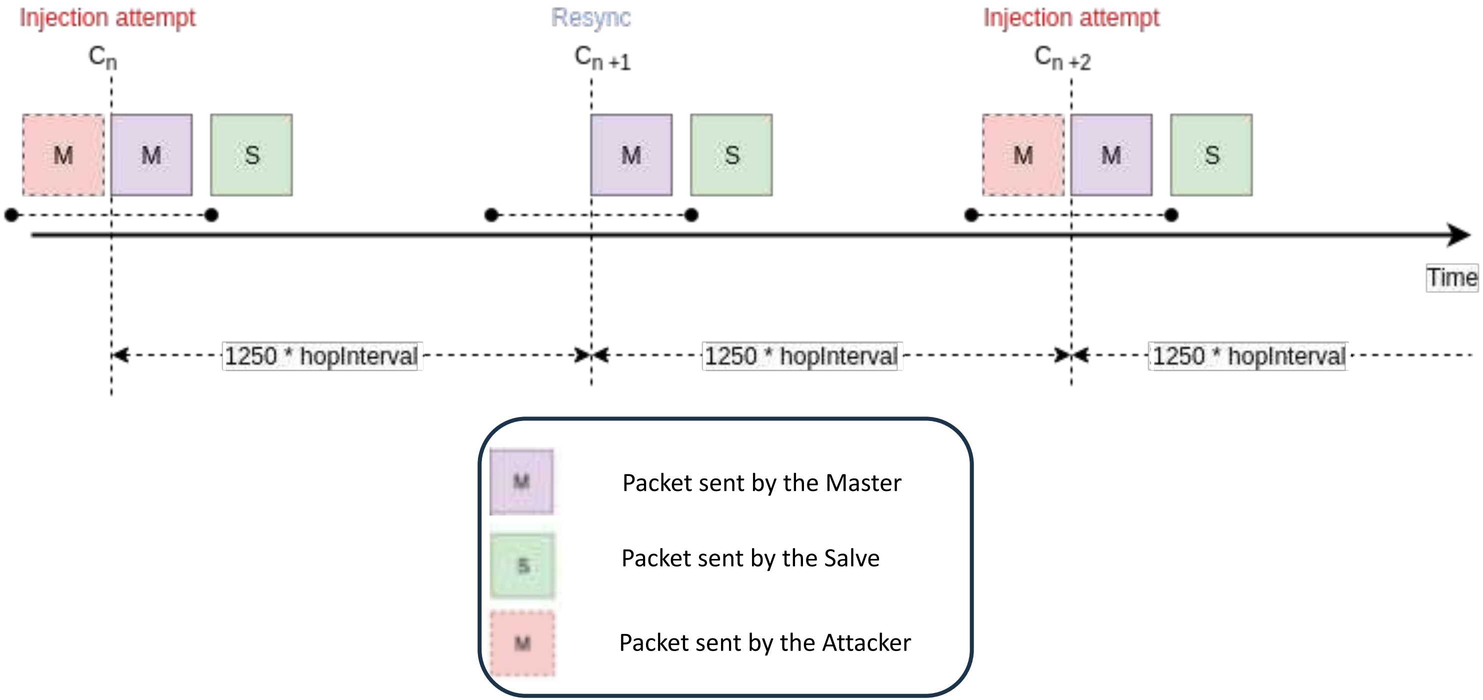




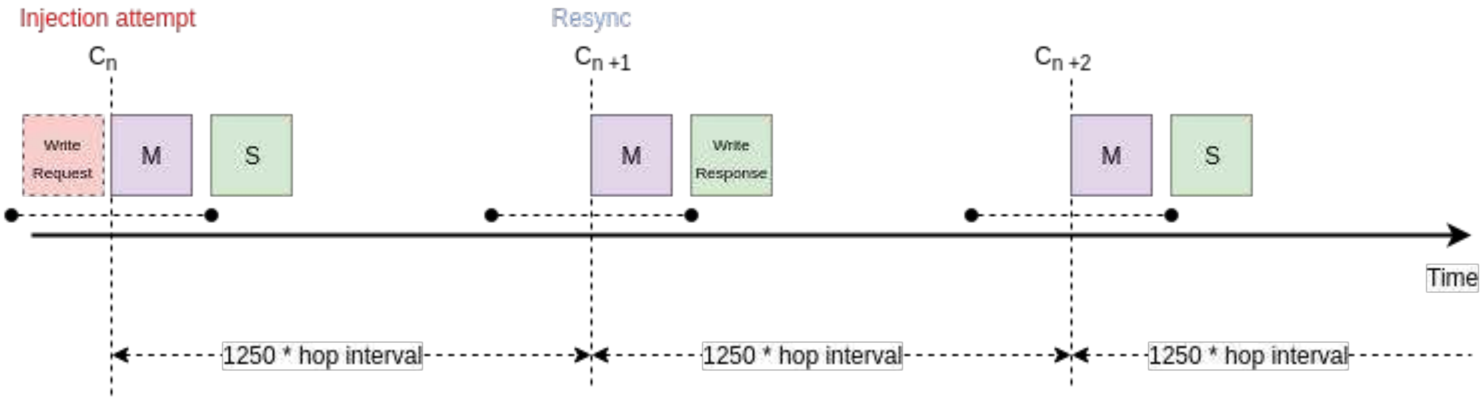
Packet sent by the Master

Packet sent by the Slave

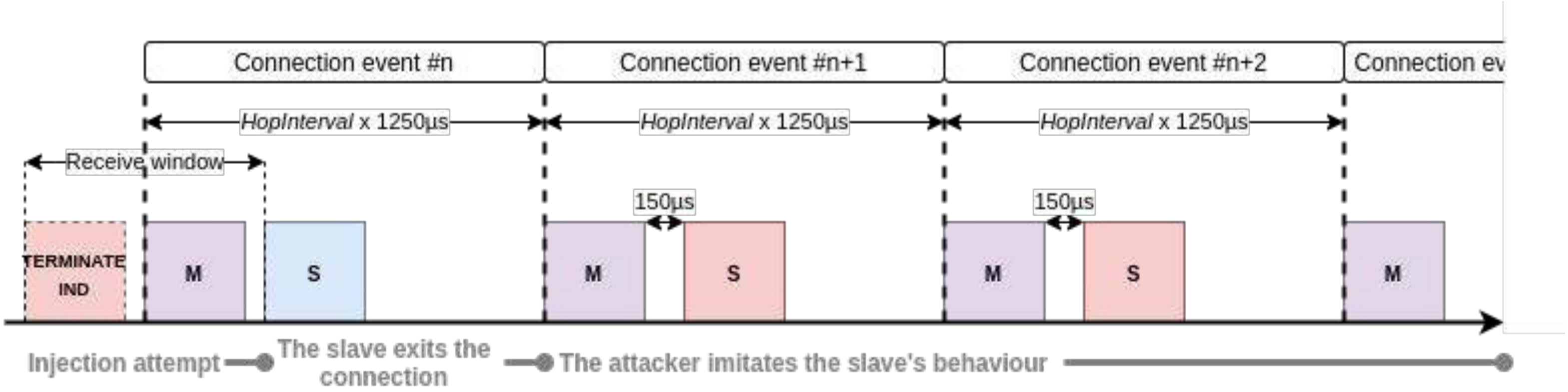
INJECTABLE – The vulnerability



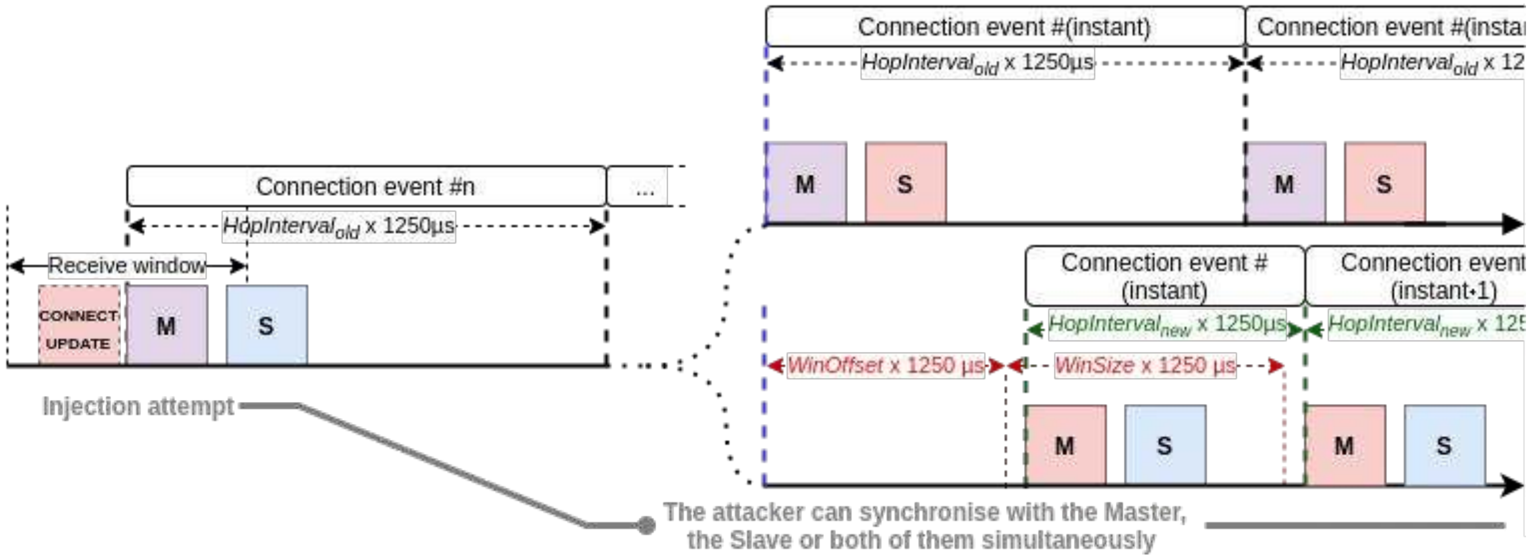
INJECTABLE – Scenario 1: triggering an unexpected behaviour



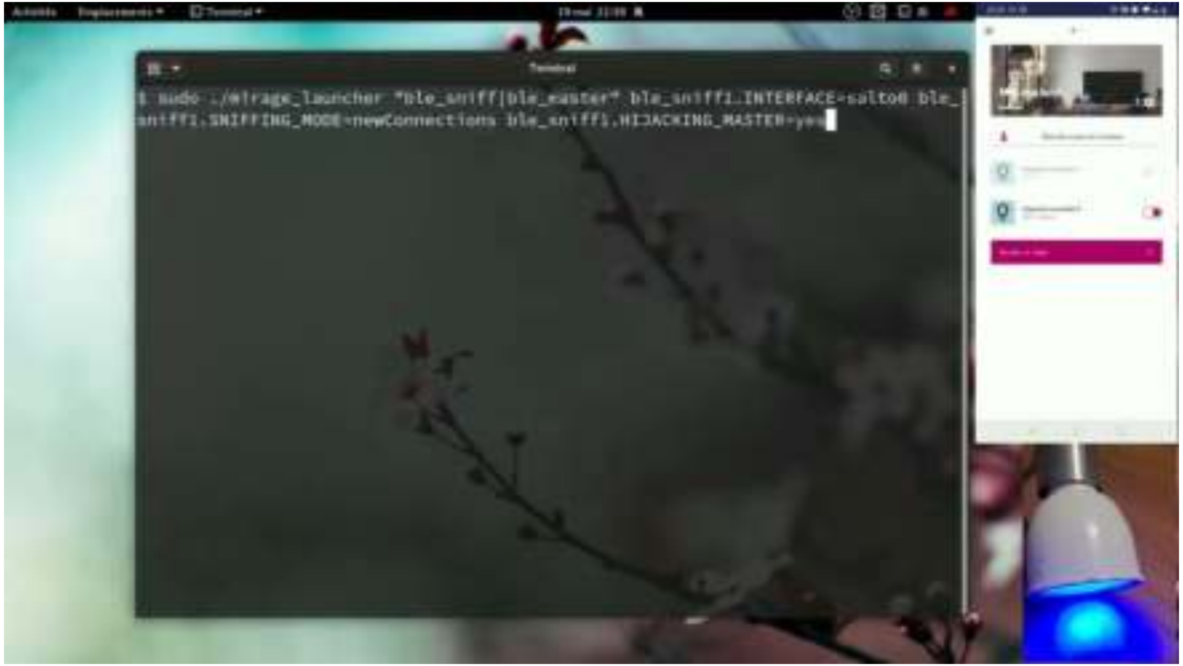
INJECTABLE – Scenario 2: slave hijacking



INJECTABLE – Scenario 3: hijacking & Man-in-the-Middle



INJECTABLE – Demo



INJECTABLE – Conclusion

- Functional proof-of-concept implemented on an nRF52840 dongle

- Vulnerability reported to Bluetooth SIG (CVE-2021-31615)

- Counter-measures
    - reduce the size of the listening window
    - activate Link Layer encryption
        - limits the impact but does not correct the vulnerability
        - most of BLE commercial devices do not use native encryption

- Publications
    - [SSTIC 2021], [DSN 2021]

Bad design : Padlock, keyboards and mices vulnerabilities

Smart Padlock - Security Analysis

- Proprietary protocol **built over Bluetooth Low Energy GATT** (applicative layer)
  - **Sniffing is not trivial**: channel hopping algorithm
  - **Security oriented device:** How to analyze **such a protocol ?**

- **Companion app** on smartphone to unlock / lock the padlock
  - Android (easier to analyze)
  - iOS

- We can combine an **OTA analysis** and **companion application reverse engineering**

Thanks to Orlaine Guetsa, Alexandre Goncalvez, Morgan Yakhelef (TLS-SEC Students) (presentation LeHack 2023)

Smart Padlock - OTA Analysis

```
[INFO] Write Request (from master) : handle = 0xb / value = 0100
[INFO] Redirecting to slave ...
[INFO] Write Response (from slave)
[INFO] Redirecting to master ...
[INFO] Write Command (from master) : handle = 0x8 / value = 100061f7c26ccaadfeec4e21041bd42e1eb4
[INFO] Redirecting to slave ...
[INFO] Handle Value Notification (from slave) : handle = 0xa / value = 300041fac4208386d82a2959bc12d95179db9539
[INFO] Redirecting to master ...
[INFO] Handle Value Notification (from slave) : handle = 0xa / value = edce8788d3717213e95e839558254d8dd69e0975
[INFO] Redirecting to master ...
[INFO] Handle Value Notification (from slave) : handle = 0xa / value = 0c1fdda8f325ac489a01
[INFO] Redirecting to master ...
[INFO] Write Command (from master) : handle = 0x8 / value = 20005046ef35f410a266e5f87e751ce768c7cd75
[INFO] Redirecting to slave ...
[INFO] Write Command (from master) : handle = 0x8 / value = f3d7fa285d02b1ef7b059fffaece
[INFO] Redirecting to slave ...
[INFO] Handle Value Notification (from slave) : handle = 0xa / value = 10001850dbbc66e66a842aa41875b3c523a0
[INFO] Redirecting to master ...
[INFO] Slave disconnected !
```

MITM approach to capture traffic during unlock command

LAAS CNRS

Smart Padlock - OTA Analysis

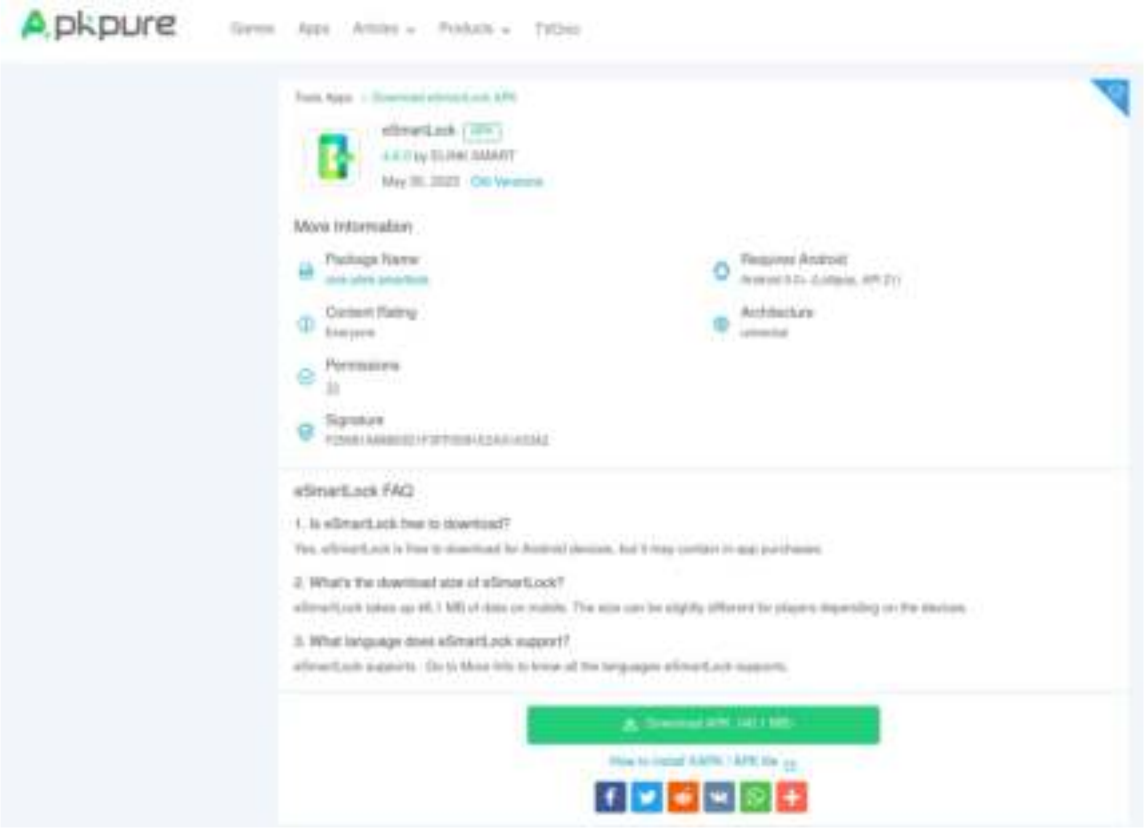
EURECOM

```
> (WriteCommand_1)  1000 dce19e8177d497e8628b72f1dd9db0e3
                    1000 56d65c330e9193cdd5419e372061ce20

> (Notification_1)  4000 5b0a21f6572b5f5162d58f916972c684fc7227b537b55c4d5742fd0ac200e04a2
f77ea077ee43db237b7bde1d890f6c152e6a9b8c162ac682db24024f9ceea6a0f030048756e74657273756e2d424c45
                    4000 ecfeb3d3e73ed4c199fa1978634372315d3c383ebfbcfd549d8aa2c659f772952
f77ea077ee43db237b7bde1d890f6c152e6a9b8c162ac682db24024f9ceea6a0f030048756e74657273756e2d424c45

> (WriteCommand_2)  2000 175dfa8885f9a743c047f5d2ca926645 76ac a673f0875b75b1279c92f91de0b7
                    2000 1249a7c8a56bd231a29a41ff95b5d255 76ac a673f0875b75b1279c92f91de0b7

> (Notification_2)  1000 1050dbbc66e66a842aa41875b3c523a0
                    1000 1050dbbc66e66a842aa41875b3c523a0
```

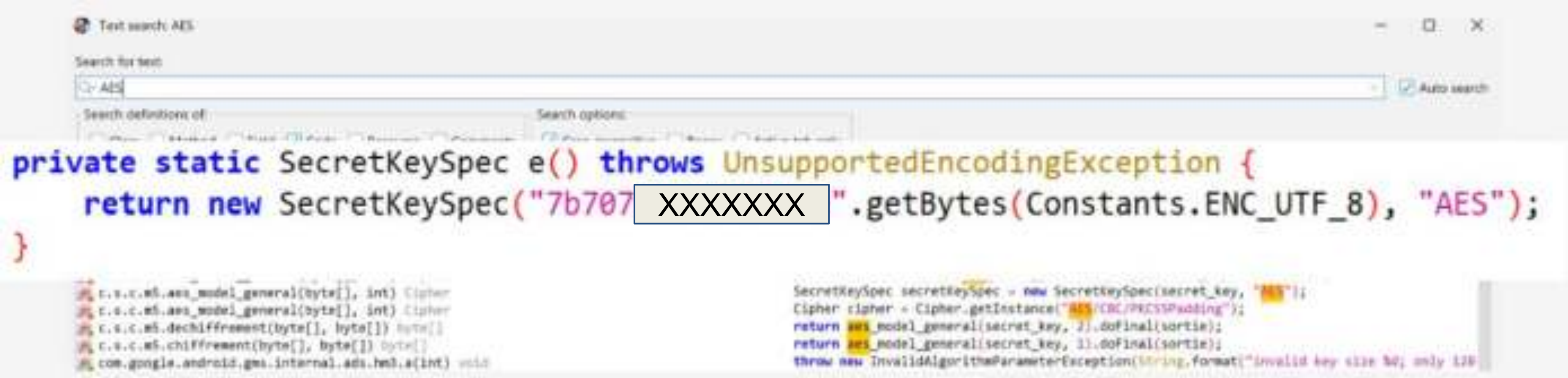
Two consecutive communications (unlock command)

recurrent field, length dependent => data looks random, but repeated pattern : AES-ECB encryption ?

Smart Padlock - App reverse engineering



Smart Padlock - App reverse engineering



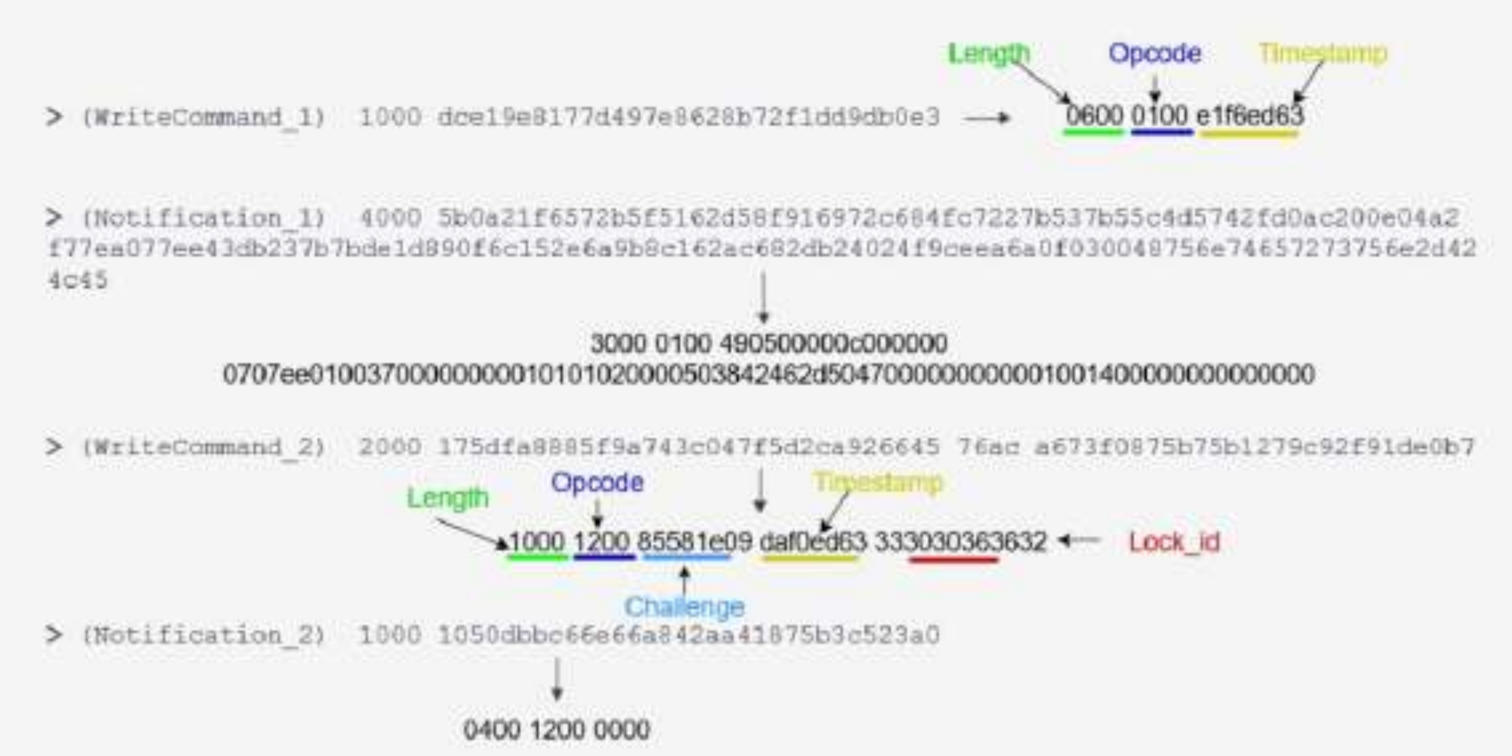
Hard-coded AES encryption key !!

Smart Padlock - App reverse engineering

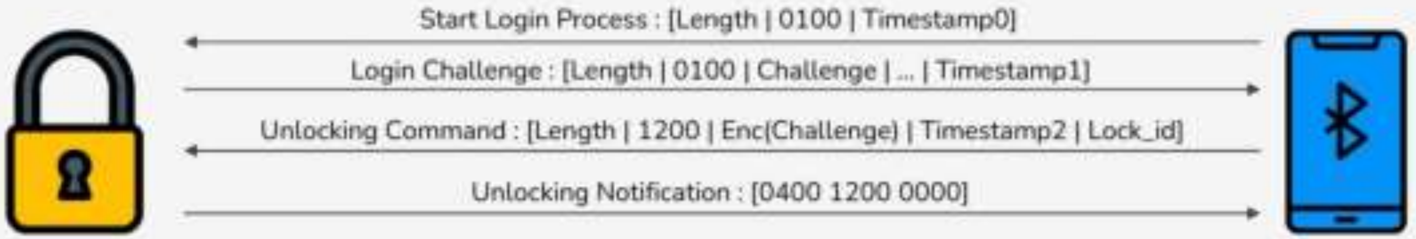
```
public static byte[] k(int i2, short s) {
    byte[] bArr = new byte[10];
    System.arraycopy(Packet.shortToByteArray_Little((short) 8), 0, bArr, 0, 2);
    System.arraycopy(Packet.shortToByteArray_Little((short) 21), 0, bArr, 2, 2);
    System.arraycopy(Packet.intToByteArray_Little(i2), 0, bArr, 4, 4);
    System.arraycopy(Packet.shortToByteArray_Little(s), 0, bArr, 8, 2);
    i f2 = c.n.a.f.f("BleProtocolUtils");
    f2.i("--packageDeleteFingerprint-- bDeleteFgp:" + c.g.a.a.s.a.a(bArr, ","));
    return n(bArr);
}
```

Structure of the plain text

Smart Padlock - App reverse engineering



Smart Padlock - App reverse engineering

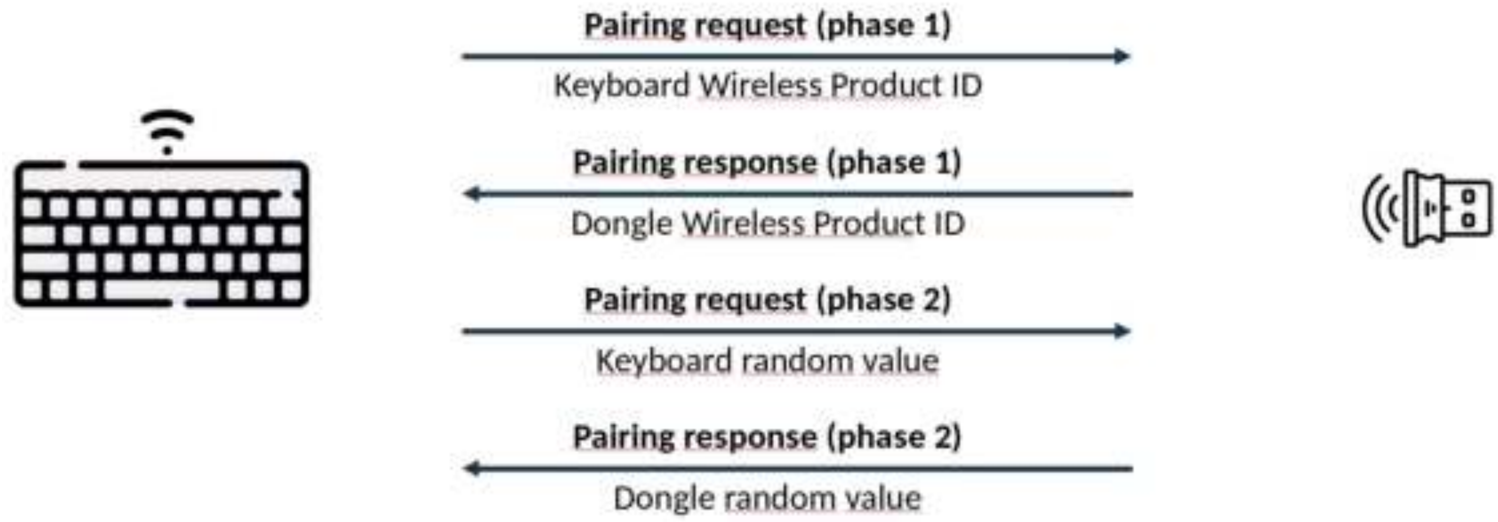


We need to:

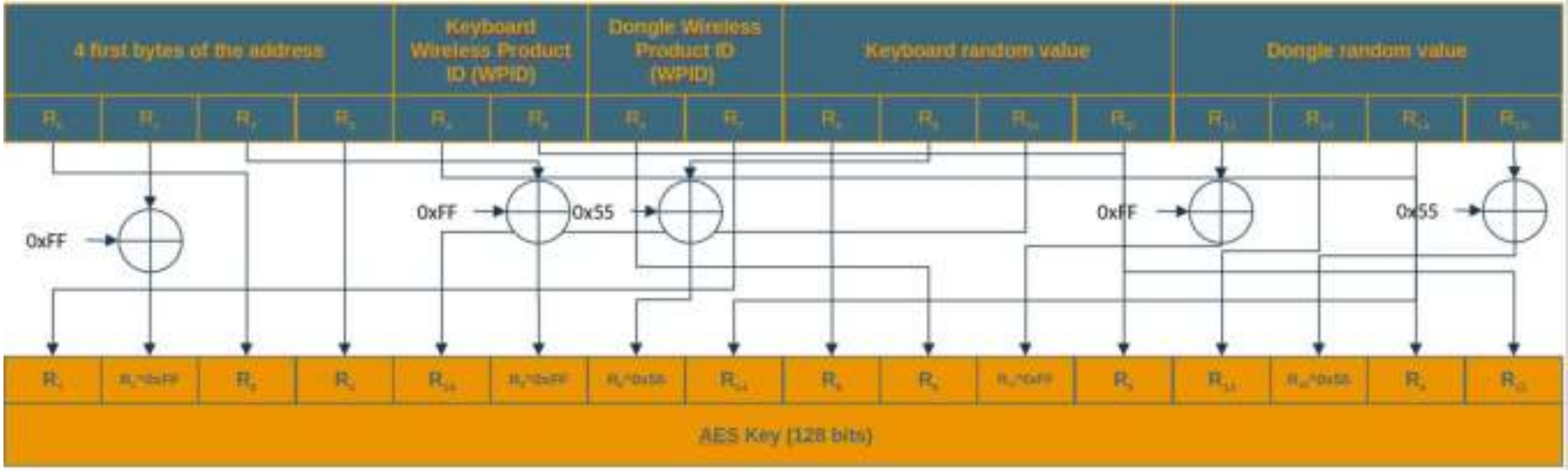
- Extract the challenge from the first message
- Send a valid message with the encrypted challenge, a "correct" timestamp (Timestamp0 < Timestamp1< Timestamp2) and the lock id

Keyboards and mices – Logitacker : Logitech Pairing process

- Generating an AES key during the pairing process
- A Diffie-Hellman could be a good idea ... too easy !
- A home-made obscure algorithm ... but vulnerable
- **Marc Newlin (2016), Marcus Mengs (2019)**

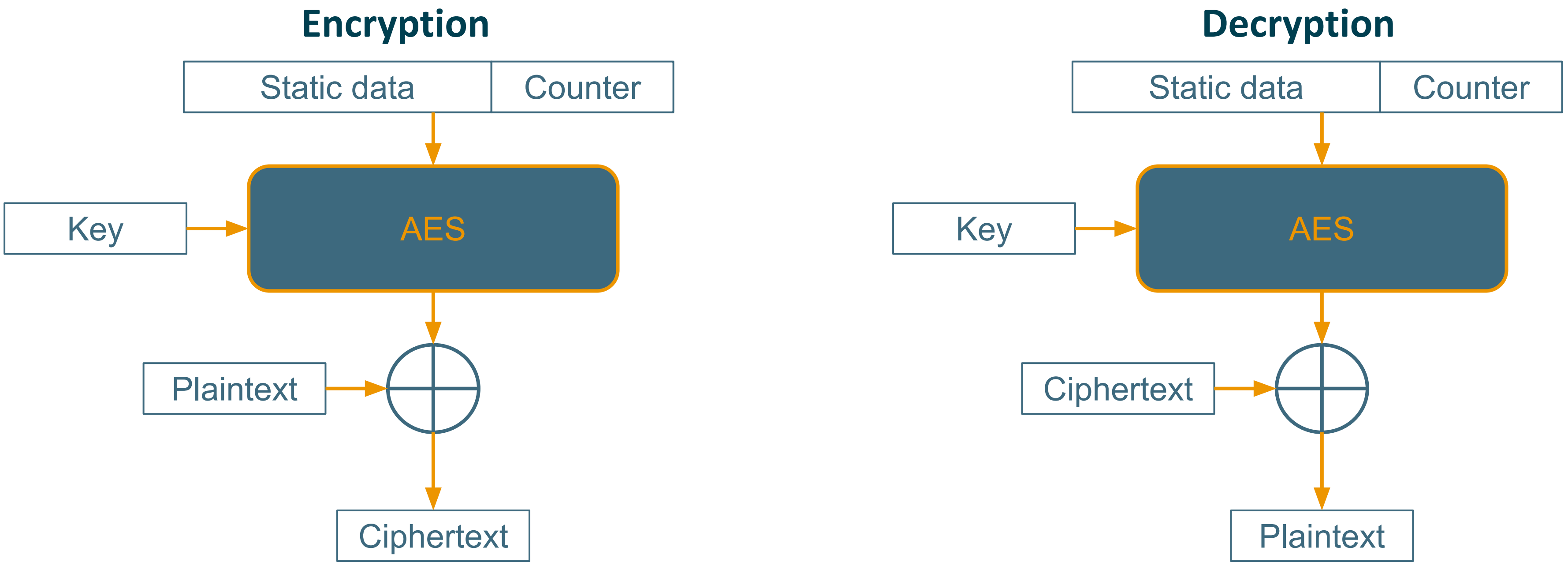


Keyboards and mices – Logitacker : Logitech Pairing process



With such an algorithm, trivial to calculate the key by observing the two phase 1 and phase 2 messages

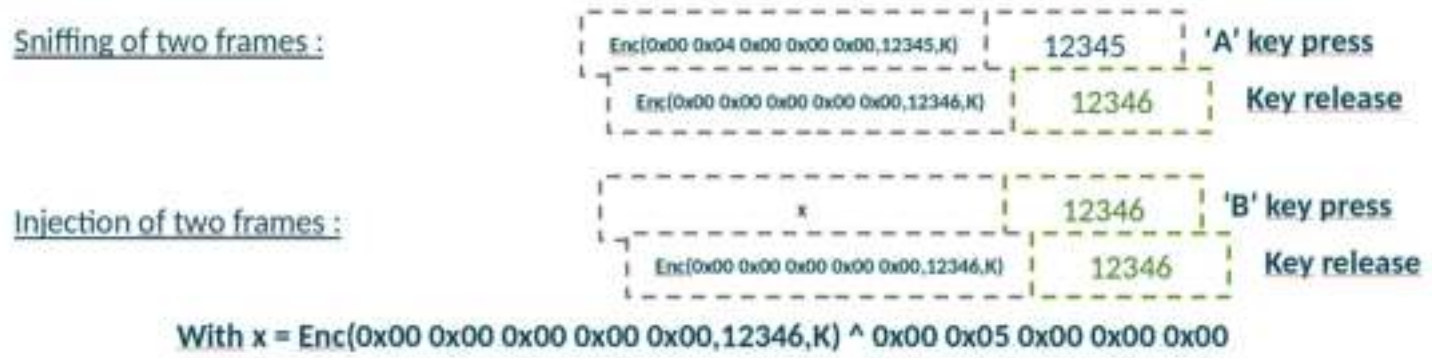
Keyboards and mices – Mousejack: Logitech encryption process



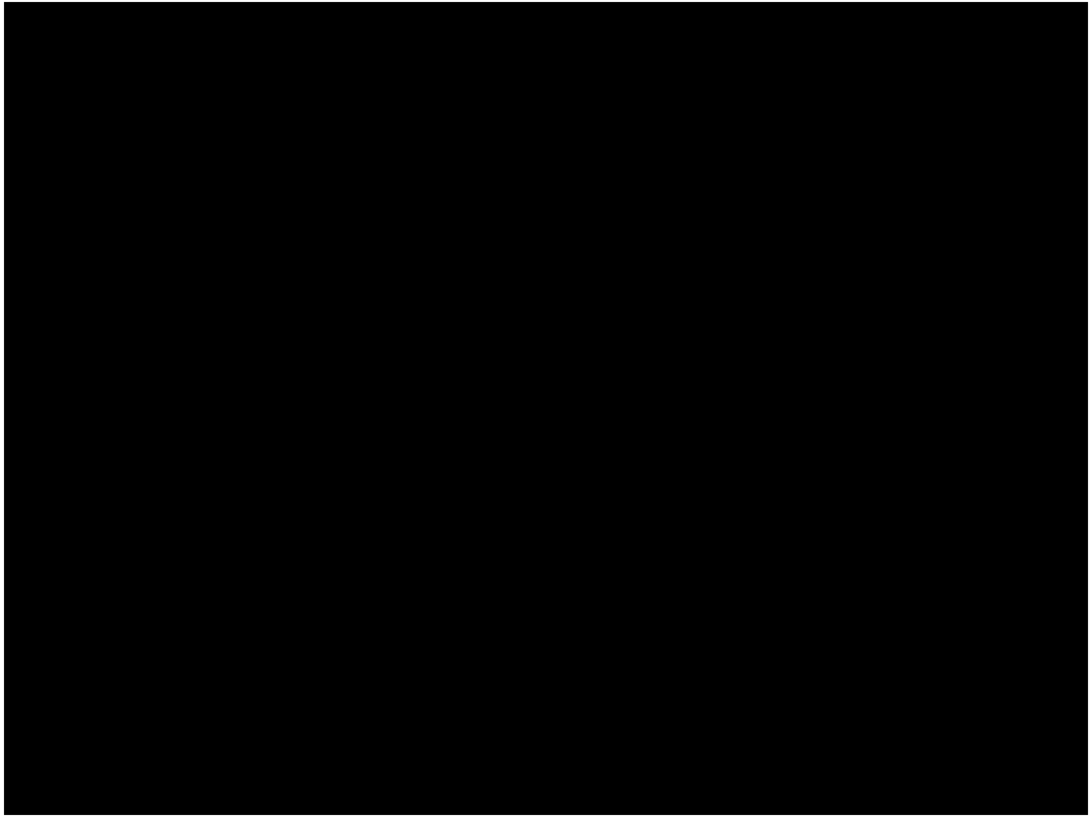
Packet format: | Enc(Plaintext,Counter,Key) | Counter |

Keyboards and mices – Mousejack: Logitech encryption process

- Plaintext messages are well known
  - Five bytes, representing the HID code of a key press 0x00 0x04 0x00 0x00 0x00 : A key press
  - Five bytes representing the release of the keystroke 0x00 0x00 0x00 0X00 0x00 : release
- Possibility to reuse a counter
  → Possibility to inject frames !



Keyboards and mices - Demo



Padlock, keyboards and mices - Conclusion

- High impact vulnerabilities linked to bad design decisions
- Really increases the discomfort about security of IoT objects
- We need more research on this topic to improve the situation

Conclusion and future work

Conclusion

- Security of connected objects not really addressed by manufacturers
  - lack of security knowledge and skills
  - time to market
  - heterogeneity of communications protocols

  → things do not seem to have changed for 10 years

  → many vulnerabilities easy to avoid, but others are tricky (WazaBee, InjectaBLE)

- Things must change ! :)
  - Continue to publish vulnerabilities and communicate
  - Build new defensive solutions
    - lightweight cryptography ?
    - embedded intrusion detection mechanisms ?
  - Need to automate the vulnerability analysis: requires a lot of skills (software, hardware, network, signal processing …)
  - Inform End users: associate a security label to each connected object ?

LAAS CNRS

EURECOM

Available tools

- **Mirage: a framework for wireless security audit**
  - Git: https://github.com/RCayre/mirage
  - Documentation: https://homepages.laas.fr/rcayre/mirage-documentation
- **ButteRFly: firmware for nRF52840 implementing InjectaBLE**
  - Git: https://github.com/RCayre/injectable-firmware
- **OASIS: a defensive framework for instrumenting BLE controllers**
  - Git: https://github.com/RCayre/oasis
  - Documentation: https://homepages.laas.fr/rcayre/oasis-documentation
- **WazaBee: a cross-protocol pivoting attack (BLE / ZigBee)**
  - Git for firmware nRF52: https://github.com/RCayre/wazabee_nrf52
  - Git for firmware TI-CC1352R1: https://github.com/RCayre/wazabee_ti
  - Git for python interface: https://github.com/RCayre/wazabee_cli
- **RadioSploit: cross-protocol attacks platform for smartphone**
  - Git for firmware patches: https://github.com/RCayre/radiosploit_patches
  - Gir for Android Application: https://github.com/RCayre/radiosploit

Thanks for your attention !