Experimental security study of WirelessHART

Kais Sellami[1] Romain Cayre[1, 2] Elies Tali[2] Pierre Ayoub[2]
Vincent Nicomette[1, 2] Guillaume Auriol[1, 2]

[1]Univ. Toulouse, INSA, France
[2]LAAS-CNRS, Toulouse, France

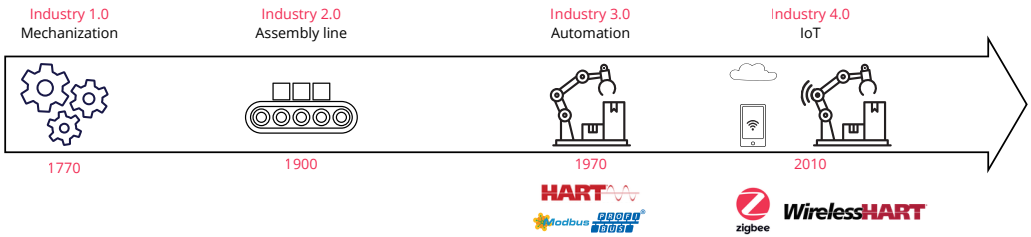
May 2026

Who am I?

- Final year INSA student
- TLS-SEC program: cybersecurity
- Last year internship: WirelessHART security analysis in LAAS-CNRS
- Focus on industrial security and IIoT

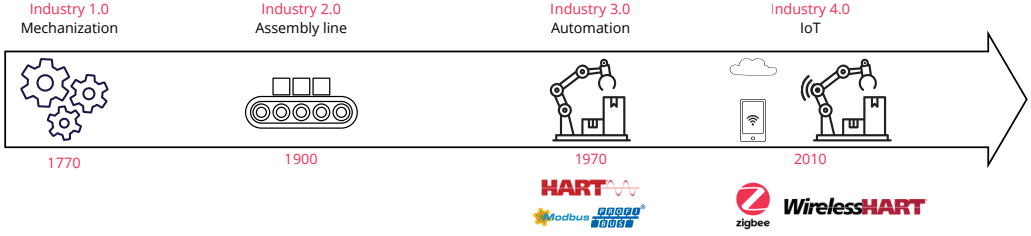
Industrial Context

- **Evolution of Industry**
  - From automation to industry 4.0
  - Digitalization and system interconnection
  - Emergence of Industrial IoT (IIoT)
- **Industrial Protocols**
  - Wired protocols: Modbus, PROFIBUS
  - Limited scalability



Industrial Context

- **Evolution of Industry**
  - From automation to industry 4.0
  - Digitalization and system interconnection
  - Emergence of Industrial IoT (IIoT)
- **Industrial Protocols**
  - Wired protocols: Modbus, PROFIBUS
  - Limited scalability
- **Adoption of Wireless Technologies**
  - WirelessHART, ISA100.11a, Zigbee
  - Flexible deployment, lower cost
- **Challenges**
  - Security concerns (cyberattacks, data integrity)
  - Coexistence with legacy systems



During this presentation

Focus on WirelessHART

- Sniffer
  - WirelessHART fundamentals
  - WirelessHART communications
  - Implementation
- Attacks
  - Mass-deauthentication
  - Time desynchronization

WirelessHART use cases

Pharmaceutical

WirelessHART use cases

Pharmaceutical

Petrochemical

WirelessHART use cases

Pharmaceutical

Petrochemical

Energy

WirelessHART use cases

Pharmaceutical

Petrochemical

Energy

Steel

Problem Statement

- Limited existing research
  - Less studied than other wireless protocols
  - Few experimental studies available

Problem Statement

- Limited existing research
  - Less studied than other wireless protocols
  - Few experimental studies available
- No open-source implementation
  - No fully accessible protocol stack
  - Difficult to reproduce and experiment

Problem Statement

- Limited existing research
  - Less studied than other wireless protocols
  - Few experimental studies available
- No open-source implementation
  - No fully accessible protocol stack
  - Difficult to reproduce and experiment
- Limited tooling
  - No low-cost sniffing tools
  - Research mostly relies on simulations

Problem Statement

- Limited existing research
  - Less studied than other wireless protocols
  - Few experimental studies available
- No open-source implementation
  - No fully accessible protocol stack
  - Difficult to reproduce and experiment
- Limited tooling
  - No low-cost sniffing tools
  - Research mostly relies on simulations

Objectives

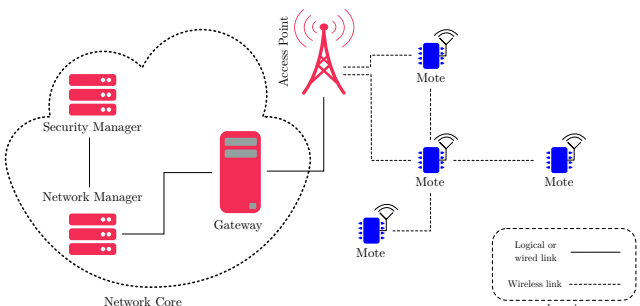
- Design a low-cost open-source sniffer
- Enable controlled traffic injection
- Evaluate practical attacks on WirelessHART

WirelessHART fundamentals

- 2.4 GHz ISM radio band
  → License-free
- Based on the IEEE 802.15.4 physical layer
  → Low power

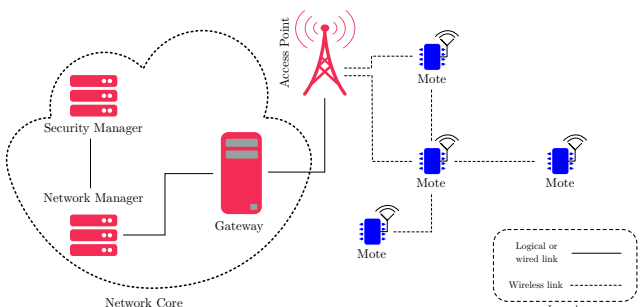
WirelessHART fundamentals

- 2.4 GHz ISM radio band
  → License-free
- Based on the IEEE 802.15.4 physical layer
  → Low power
- Mesh networking
  → Self-healing



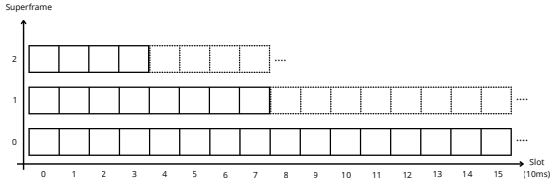
WirelessHART fundamentals

- 2.4 GHz ISM radio band
  → License-free
- Based on the IEEE 802.15.4 physical layer
  → Low power
- Mesh networking
  → Self-healing
- Time Division Multiple Access
  → Deterministic
- Frequency Hopping Spread Spectrum
  → Interference resilience



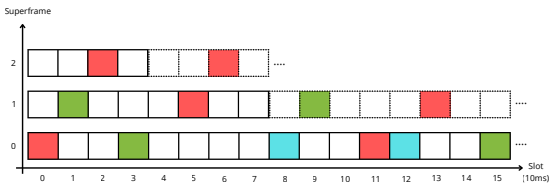
WirelessHART communications

- **Time Division:** Fixed 10ms slots grouped in *superframes*.



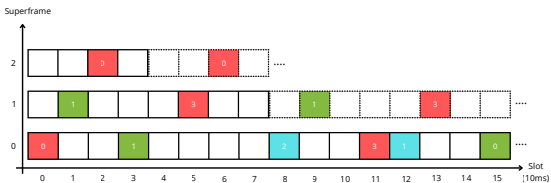
WirelessHART communications

- **Time Division:** Fixed 10ms slots grouped in *superframes*.
- **Links:** Plannified communications



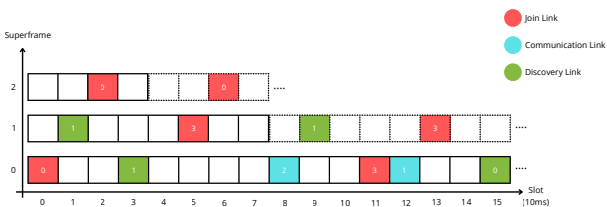
WirelessHART communications

- **Time Division:** Fixed 10ms slots grouped in *superframes*.
- **Links:** Plannified communications
  - Described by: Superframe ID, Slot index, **Offset**, Type and Options.



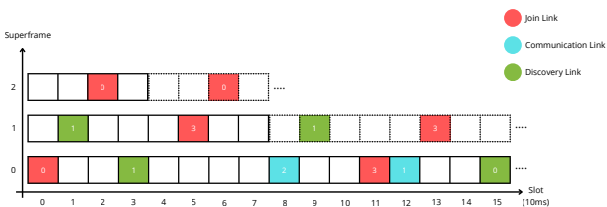
WirelessHART communications

- **Time Division:** Fixed 10ms slots grouped in *superframes*.
- **Links:** Plannified communications
  - Described by: Superframe ID, Slot index, **Offset**, Type and Options.
  - Types: Join, Broadcast, Discovery and Normal communications.



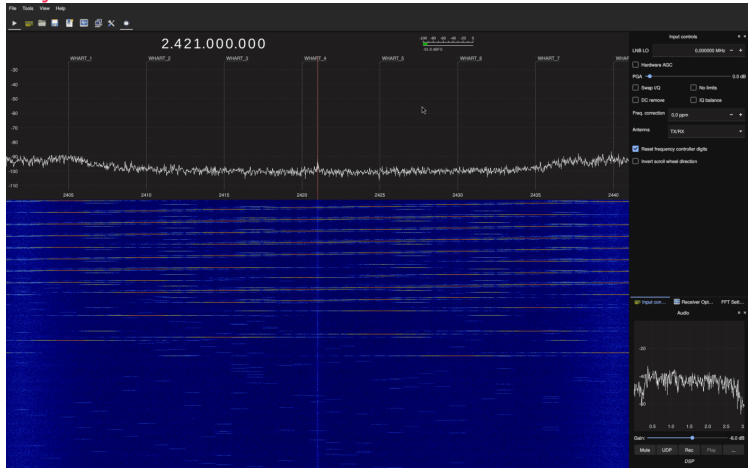
WirelessHART communications

- **Time Division:** Fixed 10ms slots grouped in *superframes*.
- **Links:** Plannified communications
  - Described by: Superframe ID, Slot index, **Offset**, Type and Options.
  - Types: Join, Broadcast, Discovery and Normal communications.
- **Management:** The *Network Manager* is the only responsible for the configuration of the network, scheduling communications and managing the routing table.



WirelessHART communications

Spectrum Analyzer



WirelessHART communications

- **Frequencies used:** Use of channels 11 to 25 (IEEE 802.15.4) declared by a two-byte *channel map*.
- **Frequency hopping algorithm:**

$$\text{index} = (\text{ASN}^* + \text{offset}) \bmod N_{\text{used}}{}^{\dagger} \tag{1}$$

*ASN=Absolute Slot Number

†N_{used} = Number of used channels

WirelessHART communications

- **Frequencies used:** Use of channels 11 to 25 (IEEE 802.15.4) declared by a two-byte *channel map*.
- **Frequency hopping algorithm:**

$$\text{index} = (\text{ASN}^* + \text{offset}) \bmod N_{\text{used}}{}^{\dagger} \tag{1}$$

$$\text{channel} = \text{activeChannelArray}[\text{index}] \tag{2}$$

*ASN=Absolute Slot Number

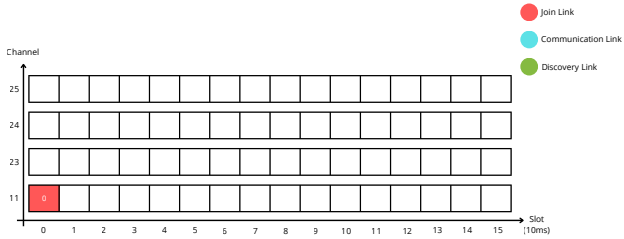
†N_{used} = *Number of used channels*

WirelessHART communications

- **Frequencies used:** Use of channels 11 to 25 (IEEE 802.15.4) declared by a two-byte *channel map*.
- **Frequency hopping algorithm:**

$$\text{index} = (\text{ASN}^* + \text{offset}) \bmod N_{\text{used}}{}^{\dagger} \tag{1}$$

$$\text{channel} = \text{activeChannelArray}[\text{index}] \tag{2}$$



*ASN=Absolute Slot Number

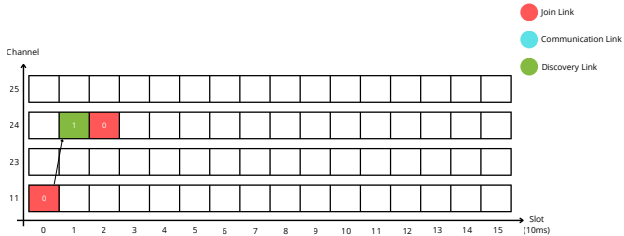
†N_{used} = Number of used channels

WirelessHART communications

- **Frequencies used:** Use of channels 11 to 25 (IEEE 802.15.4) declared by a two-byte *channel map*.
- **Frequency hopping algorithm:**

$$\text{index} = (\text{ASN}^* + \text{offset}) \bmod N_{\text{used}}{}^{\dagger} \tag{1}$$

$$\text{channel} = \text{activeChannelArray}[\text{index}] \tag{2}$$



*ASN=Absolute Slot Number

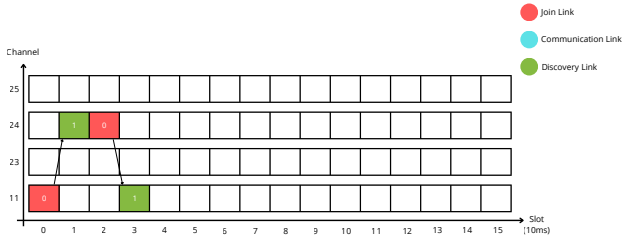
†N_{used} = Number of used channels

WirelessHART communications

- **Frequencies used:** Use of channels 11 to 25 (IEEE 802.15.4) declared by a two-byte *channel map*.
- **Frequency hopping algorithm:**

$$\text{index} = (\text{ASN}^* + \text{offset}) \bmod N_{\text{used}}{}^{\dagger} \tag{1}$$

$$\text{channel} = \text{activeChannelArray}[\text{index}] \tag{2}$$



*ASN=Absolute Slot Number

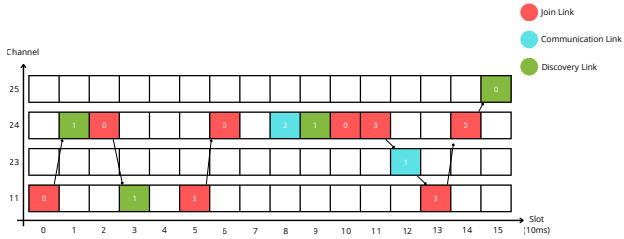
†N_{used} = Number of used channels

WirelessHART communications

- **Frequencies used:** Use of channels 11 to 25 (IEEE 802.15.4) declared by a two-byte *channel map*.
- **Frequency hopping algorithm:**

$$\text{index} = (\text{ASN}^* + \text{offset}) \bmod N_{\text{used}}{}^{\dagger} \tag{1}$$

$$\text{channel} = \text{activeChannelArray}[\text{index}] \tag{2}$$



*ASN=Absolute Slot Number

†N_{used} = Number of used channels

WHAD – Wireless HAcking Devices

Open-source wireless security framework

- Supports BLE, ZigBee, LoRaWAN...
- Unified host/hardware radio interface

Three components

- `whad-client` – host-side
- `ButteRFly` – dongle firmware
- `whad-protocol` – communication protocol

WirelessHART Sniffer

10 ms slots / 2 ms RX windows
⇒ host-side processing not feasible

ButteRFly (firmware)

- Active channel computation
- Radio channel switching
- Hardware timer for slot synchronization
- Time adjustment from ACK fields
- Packet capture & forwarding via `whad-protocol`

whad-client (Python)

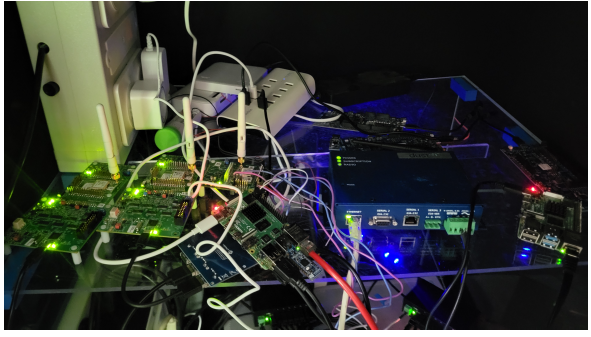
- Packet decryption & encryption
- Injection frame generation
- Unobserved link inference (exploration algorithm)
- WirelessHART dissection via **Scapy**
- Logging & capture export

Experimental Setup: Dust Network

- **Hardware Platform**
  - Dust Networks technology (Analog Devices)
- **Network Composition**
  - One gateway: Network manager + security manager + AP
  - Multiple motes

Sniffer evaluation

Experience	Duration	Environment	Number of motes	Traffic	Sniffer	**Total**
A	14 h	Lab	Gateway + 2 motes	Normal	Snif_4	14601
					Snif_{FH}	14593
					Loss rate	**0.05%**
B	45 mins	Domestic	Gateway + 5 motes	Continuous ping	Snif_4	25500
					Snif_{FH}	22288
					Loss rate	**12.6%**



From Sniffing to Attacks

- Next step: **Attacks**
- Performing attacks requires understanding:
  - How devices communicate
  - How security mechanisms are applied

→ Communication phases and security

Communication Phases

- **Advertisement Phase**
  - Access Point announces *join links*
  - Mote listens and synchronizes itself

Communication Phases

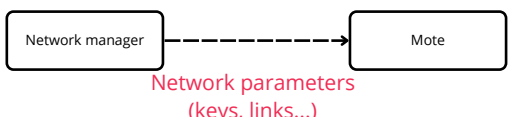
- **Advertisement Phase**
  - Access Point announces *join links*
  - Mote listens and synchronizes itself
- **Join Phase**
  - Mote sends a *join request* to the Network Manager

Communication Phases

- **Advertisement Phase**
  - Access Point announces *join links*
  - Mote listens and synchronizes itself
- **Join Phase**
  - Mote sends a *join request* to the Network Manager
  - Network Manager authenticates the mote

Communication Phases

- **Advertisement Phase**
  - Access Point announces *join links*
  - Mote listens and synchronizes itself
- **Join Phase**
  - Mote sends a *join request* to the Network Manager
  - Network Manager authenticates the mote
  - Allocation of communication links and network parameters



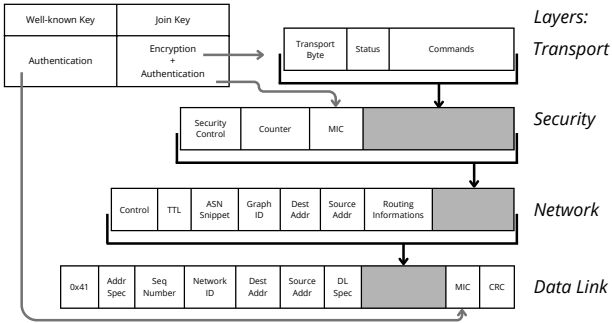
Communication Phases

- **Advertisement Phase**
  - Access Point announces *join links*
  - Mote listens and synchronizes itself
- **Join Phase**
  - Mote sends a *join request* to the Network Manager
  - Network Manager authenticates the mote
  - Allocation of communication links and network parameters
- **Secure Communication Phase**
  - Communications scheduled
  - Data encrypted and authenticated

WirelessHART security

Cryptographic keys

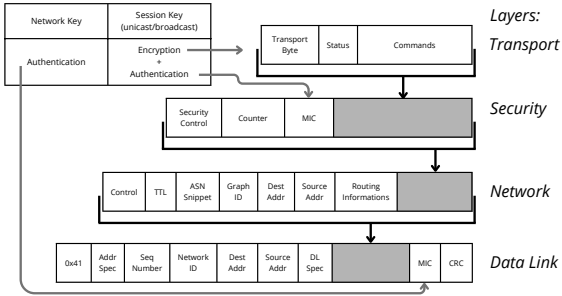
- **Well-known key:**
  - used for MIC calculation during advertising and joining at Data Link Layer (DLL)
- **Join key:**
  - pre-shared
  - secure the joining (handshake): encryption + authentication



WirelessHART security

Cryptographic keys

- **Network key:**
  - shared between all the network (Data Link Layer)
  - used to authenticate messages.
- **Session Keys:**
  - **Unicast:** confidentiality between nodes.
  - **Broadcast:** for broadcast messages.



WirelessHART security

Based on AES-128 CCM*

- **Confidentiality:** data encryption using session keys.

WirelessHART security

Based on AES-128 CCM*

- **Confidentiality:** data encryption using session keys.
- **Integrity:** use of MIC (Message Integrity Code)
  - **NWK-MIC:** integrity of encrypted data at Network Layer
  - **DL-MIC:** integrity of the whole PDU at Data Link Layer

WirelessHART security

Based on AES-128 CCM*

- **Confidentiality:** data encryption using session keys.
- **Integrity:** use of MIC (Message Integrity Code)
  - **NWK-MIC:** integrity of encrypted data at Network Layer
  - **DL-MIC:** integrity of the whole PDU at Data Link Layer
- **Anti-replay:** use of nonce derived from the ASN.

Threat model

Assumptions

- Passive & active access to the radio medium
- Association key compromised (trash-can attack, default/weak/predictable keys)
- Passive eavesdropping of an association to recover other keys

Threat model

Attacker goals

- **Confidentiality:** intercept sensor data & infer industrial processes
- **Integrity / Authenticity:** inject malicious frames by spoofing a legitimate node
- **Availability:** suspend one or more motes

Join key extraction

Trash can attack

OTA attacks

Mass-deauthentication

Time desynchronization

Mass-deauthentication

Principle

- Exploits the Suspend command (opcode `0x3CC`)
- Orders a mote to suspend between two ASNs
- Broadcast $\Rightarrow$ simultaneous suspension of multiple motes

Requirements

- K_{network}, $K_{\text{session\_broadcast}}$
- Knowledge of communication links

Mass-deauthentication

Impact

- Partial or total DoS of the wireless network
- Disruption of control loops & industrial processes
- May require **manual intervention** to recover

Evaluation

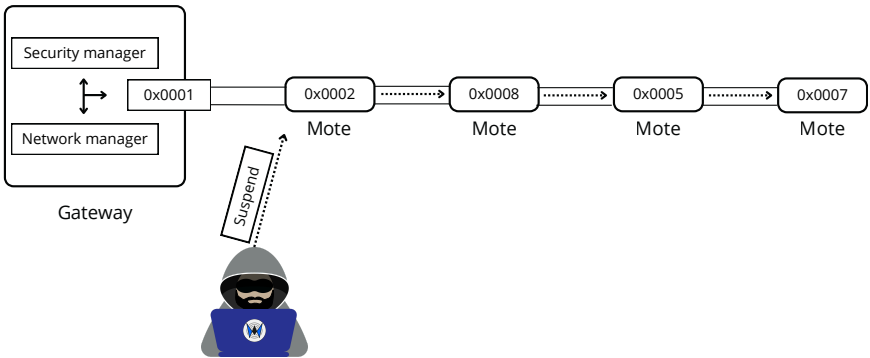
- Theorized in 2015[a]
- Never experimented due to lack of offensive tooling

[a] Duijsens, Master thesis

Mass-deauthentication scenarios

Denial of Service

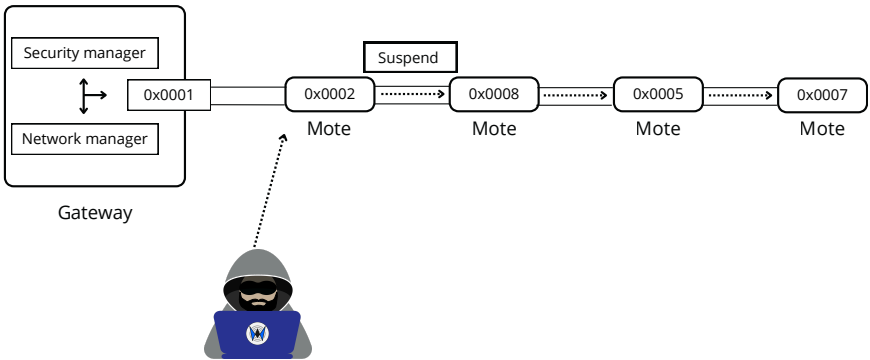
- Suspend of 1M slots $\triangleq$ 170 mins



Mass-deauthentication scenarios

Denial of Service

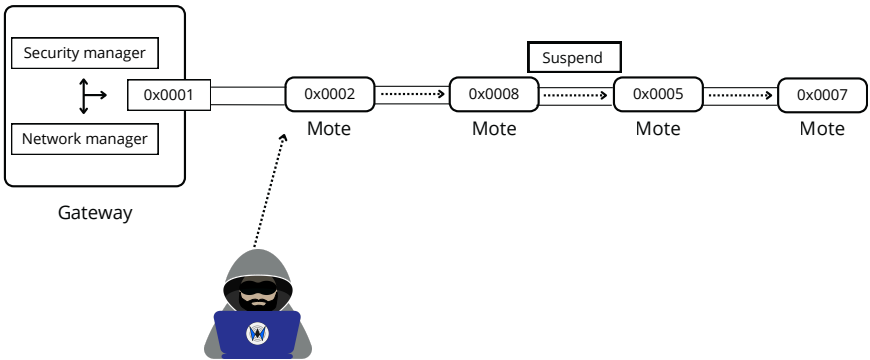
- Suspend of 1M slots $\triangleq$ 170 mins
- Routed command



Mass-deauthentication scenarios

Denial of Service

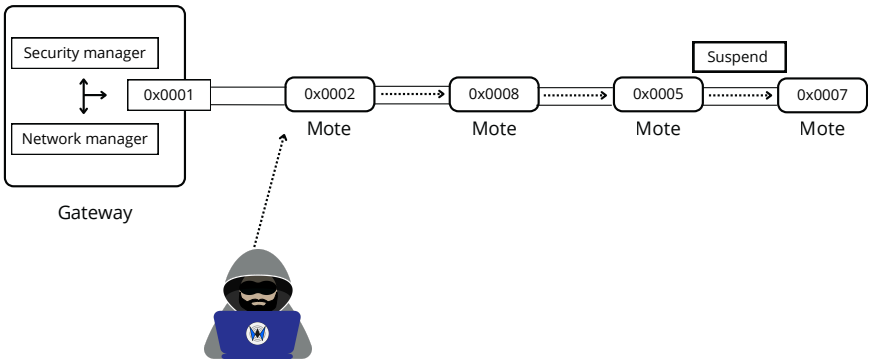
- Suspend of 1M slots $\triangleq$ 170 mins
- Routed command



Mass-deauthentication scenarios

Denial of Service

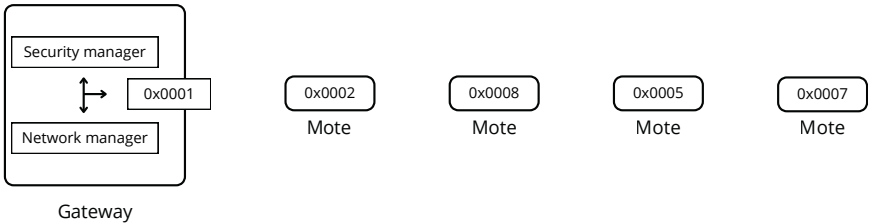
- Suspend of 1M slots $\triangleq$ 170 mins
- Routed command



Mass-deauthentication scenarios

Denial of Service

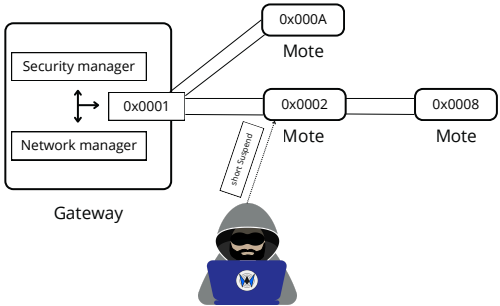
- Suspend of 1M slots $\triangleq$ 170 mins
- Routed command
- Disconnected network



Mass-deauthentication scenarios

Forced re-association

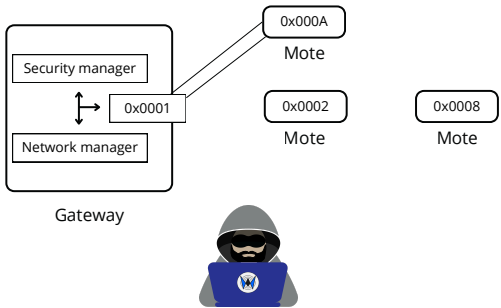
- Short suspend



Mass-deauthentication scenarios

Forced re-association

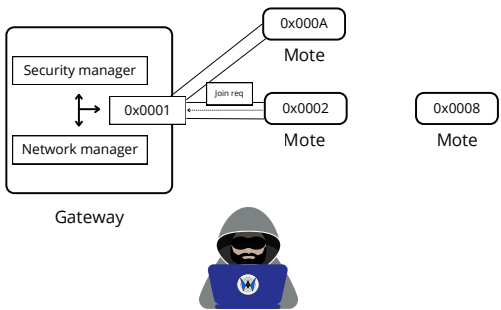
- Short suspend



Mass-deauthentication scenarios

Forced re-association

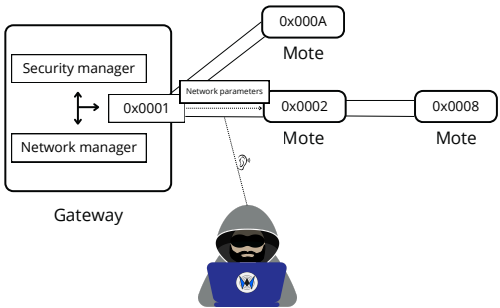
- Short suspend
- Motes in master mode reconnect automatically once the suspend finished: implementation dependent



Mass-deauthentication scenarios

Forced re-association

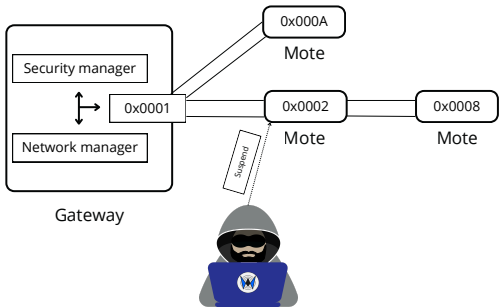
- Short suspend
- Motes in master mode reconnect automatically once the suspend finished: implementation dependent
- Revocate all of the missing keys of the attacked mote



Mass-deauthentication scenarios

Forced re-association

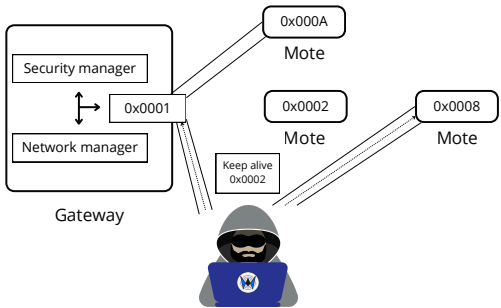
- Short suspend
- Motes in master mode reconnect automatically once the suspend finished: implementation dependent
- Revocate all of the missing keys of the attacked mote
- Impersonating by sending a Suspend command with a long delay



Mass-deauthentication scenarios

Forced re-association

- Short suspend
- Motes in master mode reconnect automatically once the suspend finished: implementation dependent
- Revocate all of the missing keys of the attacked mote
- Impersonating by sending a Suspend command with a long delay



Mass-deauthentication demo

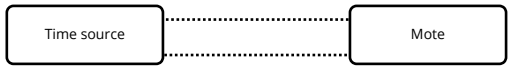
Mass-deauth attack

Time desynchronization

Neighbors

Two communicating nodes on DLL

For each peer, there exists one time source



Neighbors

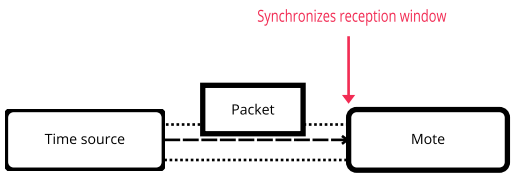
Time desynchronization

Neighbors

Two communicating nodes on DLL
For each peer, there exists one time source

Synchronization

- The reception time from time source



Neighbors

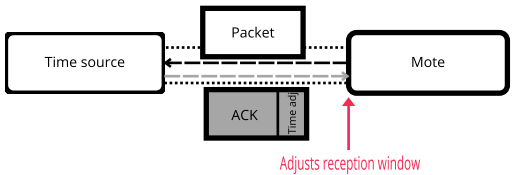
Time desynchronization

Neighbors

Two communicating nodes on DLL
For each peer, there exists one time source

Synchronization

- The reception time from time source
- Ack contain field **Time adjustment**



Neighbors

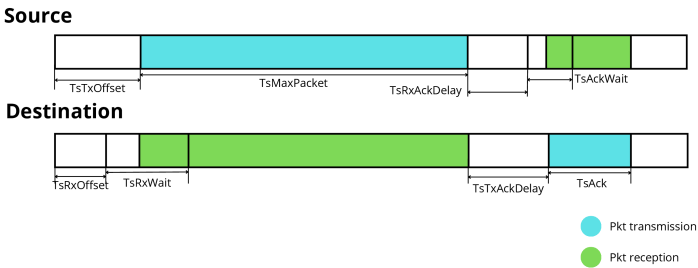
Time desynchronization

Principle

Spoof time source and send delayed traffic until desynchronizing the reception window (TsRxWait)

Impact

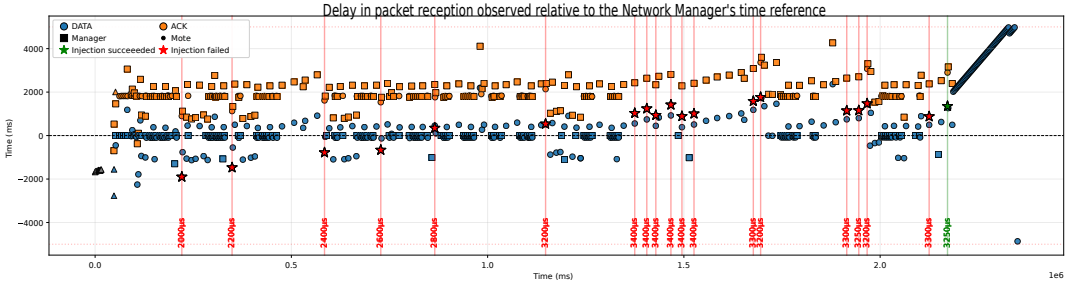
- Slot desynchronized
- Channel hopping delayed
- Denial of Service



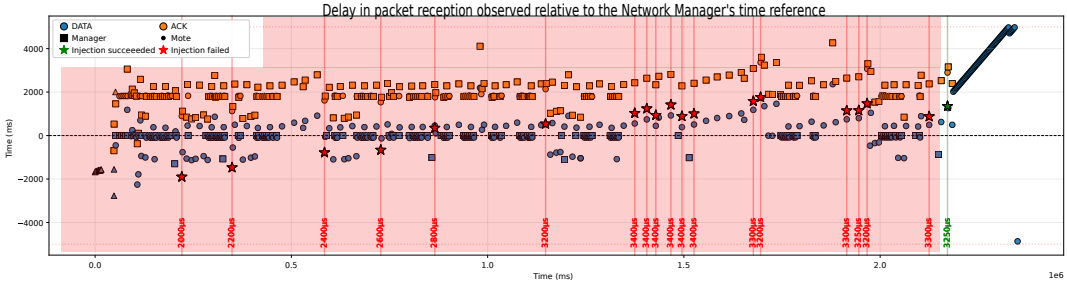
Time desynchronization demo

Time desynchronization attack

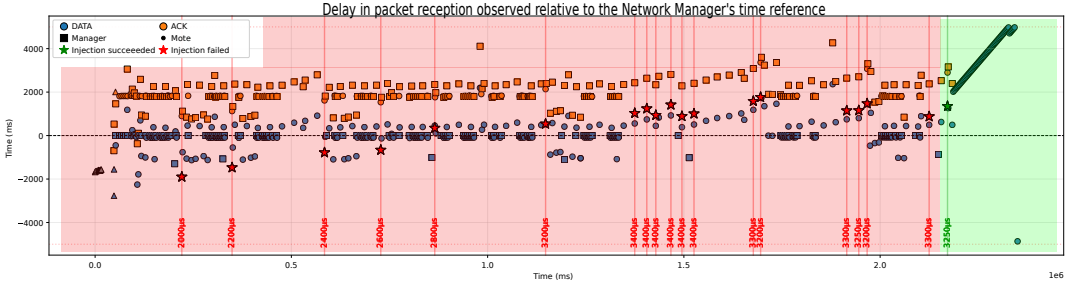
Time desynchronization evaluation



Time desynchronization evaluation



Time desynchronization evaluation



Countermeasures

Different K_{join}

- Enforce per mote Join key
- Limits compromise to a single mote

Countermeasures

Different K_{join}

- Enforce per mote Join key
- Limits compromise to a single mote

Make Suspend command unicast

- Low change to the network mechanisms
- Prevents network-wide shutdown from one node

Countermeasures

Different K_{join}

- Enforce per mote Join key
- Limits compromise to a single mote

Make Suspend command unicast

- Low change to the network mechanisms
- Prevents network-wide shutdown from one node

Detect abnormal time adjustments

- Detect abnormal values of time adjustment
- Reject large synchronization drifts

Conclusion & Perspectives

Conclusion

- WirelessHART remains insufficiently studied
- Open-sourced and low-cost experimental tool for analysis
- Practical attacks still feasible despite the protocol's security design

Conclusion & Perspectives

Conclusion

- WirelessHART remains insufficiently studied
- Open-sourced and low-cost experimental tool for analysis
- Practical attacks still feasible despite the protocol's security design

Perspectives

- Continue studying WirelessHART and test different attacks such as jamming
  - Propose an open-source stack of the WirelessHART network
  - Analyse dust network gateway mechanism
- Investigate IT/OT interactions
- Develop accessible and low-cost tooling for other wireless protocols

Thank you!

WHAD-client

https://github.com/whad-team/whad-client

ButteRFly

https://github.com/whad-team/butterfly

Bibliography

- A security analysis for wireless sensor mesh networks in highly critical systems
- Security analysis of WirelessHART communication scheme
- Security issue of WirelessHART based SCADA systems
- One for all and all for WHAD : Wireless shenanigans made easy
- Software-defined radio based measurement platform for wireless networks
- Cracking the channel hopping sequences in IEEE 802.15.4e-based industrial TSCH networks
- Revealing smart selective jamming attacks in WirelessHART networks
- Security considerations for the WirelessHART protocol
- WirelessHART : Applying wireless technology in real-time industrial process control
- WirelessHART : A Security Analysis
- FieldComm Group : WirelessHART documentation
- Formal verification of the WirelessHART protocol - verifying old and finding new attacks
- 802.15.4e in a nutshell : Survey and performance evaluation
- Reverse engineering WirelessHART hardware
- Formal security evaluation and improvement of WirelessHART protocol in industrial wireless network
- A comparison of WirelessHART and ISA100.11a
- A survey on the application of WirelessHART for industrial process monitoring and control
- Securing WirelessHART : Monitoring, exploring and detecting new vulnerabilities