TUM School of Computation, Information and Technology
Technische Universität München

Secure and User-Friendly Setup and Maintenance of Wirelessly Interconnected Embedded Systems

Martin Matthias Striegel

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr.-Ing. Wolfgang Kellerer

Prüfende der Dissertation:
1. Prof. Dr.-Ing. Georg Sigl
2. Prof. Dr. Dieter Kranzlmüller

Die Dissertation wurde am 20.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 04.12.2022 angenommen.

Abstract

Low-power wirelessly interconnected embedded systems (IES) have a wide range of applications such as body area networks (BANs) or industrial wireless sensor networks (WSNs). By now, even low-cost low-power devices are capable of using symmetric or asymmetric cryptographic operations to protect data at rest and in motion. However, due the lack of input/output (I/O) interfaces for user interaction, *setting up and maintaining* those devices still poses a challenge. Additionally, with increasing pervasiveness of those interconnected embedded systems, inevitably more and more users with no technical education will be operating and utilizing those devices.

In this thesis, I investigate *user-friendly* methods and appliances to permit even non-technical users to set up and run networks of embedded systems, enabling them to operate such networks securely along the lifecycle of the network. Contrary to previous contributions to usable security in those areas, I especially focus on *distributing firmware simultaneously to a large number of devices* and supervision of the network status *on-site* in realistic application scenarios.

I start this thesis by making the important distinction between IES and the more powerful device class commonly referred to as "Internet of Things (IoT) Devices". While they might be used in similar applications, they differ greatly in terms of setup, maintainability and the resulting attacker model. In this thesis, I focus on the former and continue by presenting the history of the research field usable security, its main themes and findings applicable to IES.

Then, I study user interaction and the resulting usable security problems in three networks of IES I have worked on. Two of those are BANs, the first being designed for work safety of firefighters utilizing IPv6 over Low power Wireless Personal Area Network (6LoWPAN), the second one for deducing a three-channel electrocardiogram (ECG) on elderly patients based on Bluetooth Low Energy (BLE). The third network utilizes screws equipped with preload-force sensors and ultra-high frequency radio-frequency identification (UHF-RFID) and MIOTY wireless connectivity to monitor whether the screws are still tightened sufficiently when attached to critical infrastructure. I model user behavior interacting with those systems and derive usable security problems in those networks, especially in secure setup and network monitoring.

I then propose a novel approach to distribute firmware and cryptographic secrets to a large number of IES simultaneously. I introduce a practical appliance and workflow using a smart Faraday Cage (FC) and over-the-air (OTA) firmware updates. User interactions are reduced to placing IES in the FC, following few and clear spoken instructions and removing and placing the embedded devices afterwards. All security-critical tasks are automatically conducted by the FC. I perform a user study which shows that removing the user from the security-loop is especially appealing to users with no or little technical background.

Based on those findings, I investigate, how this type of user can be empowered to monitor their wireless network without having to undergo extensive training. For this, I design a novel handheld network monitoring solution combining passive network observation with augmented reality (AR)-based traffic visualization. By pointing their hand held device, which is connected to a wireless traffic capture device, the user can see wireless network traffic as overlay on the camera image. By doing so, they can investigate multi-hop behavior in mesh network technologies such as 6LoWPAN and spot faulty devices. Unlike previous solutions, which require the traffic capture device to directly tap into a central place in the wireless network, my approach can be retrofit to already deployed networks without altering them, making this approach accessible to non-technical users.

Utilizing this architecture, I explore whether AR visualization can be used to help users identify attacks on a wireless network and thus educate them in wireless network security. I perform a user study with 25 participants, both non-technical users and with wireless security expertise and find that observing a 802.11 Wi-Fi (Wi-Fi) network under attack enabled even the non-technical users to not only notice, *that* an attack is taking place, but also describe the rough idea behind common attacks such as Denial of Service (DoS), Address-Resolution Protocol (ARP) spoofing and *Evil Twin*. I conclude that AR-based traffic visualization provides an accessible introduction to the hard-to-grasp topics of wireless security education and is suitable to spark the interest of students if used in school and study programs.

To summarize, I contribute results to understand user interactions with IES during the setup and runtime phases. Furthermore, I introduce two appliances and workflows to aid users in this process. Combining those results provides a complete and integrated workflow permitting even non-technical users to manage a secure wireless network of IES over its lifetime.

Zusammenfassung

Funkvernetzte eingebettete Systeme mit geringer Rechenleistung und geringem Stromverbrauch (low-power wirelessly interconnected embedded systems (IES)) haben vielfältige Anwendungen. So werden sie z.B. zur Messung von Vitaldaten am menschlichen Körper oder zur Datenerfassung in industriellen Anwendungen genutzt. Selbst diese günstigen IES können heute sowohl symmetrische als auch asymmetrische kryptografische Berechnungen durchführen und somit Daten während der Speicherung und der Übertragung absichern. Allerdings stellen die *Inbetriebnahme* sowie die *Wartung* dieser Geräte immer noch eine Herausforderung da, da es diesen Geräten an Wegen zur Ein- und Ausgabe von Daten fehlt. Zusätzlich werden diese Geräte durch ihre zunehmende Verbreitung unvermeidbar von immer mehr Nutzern, welche keinen technischen Hintergrund haben, bedient und verwendet.

In dieser Dissertation untersuche ich *nutzerfreundliche* Methoden und Apparate, welche es auch Nutzern ohne technischem Hintergrund ermöglichen, Netzwerke mit funkvernetzten eingebetteten Systemen in Betrieb zu nehmen und zu nutzen. Es wird ihnen ermöglicht, solche Netzwerke über den gesamten Lebenszyklus des Netzwerks hinweg *sicher* zu betreiben. Anders als frühere Beiträge zu nutzerfreundlicher Security in diesem Themenfeld behandle ich die *simultane Verteilung von Firmware an sehr viele Geräte.* Ebenfalls behandle ich die Überwachung des Netzwerkstatus *im Felde* in realistischen Anwendungsszenarien.

Ich beginne mit einer Gegenüberstellung von IES und der leistungsfähigeren Klasse von Geräten, welche typischerweise als Internet of Things (IoT)-Geräte bezeichnet werden. Beide Geräteklassen können zwar in Ähnlichen Anwendungen genutzt werden. Sie unterscheiden sich jedoch stark hinsichtlich der Inbetriebnahme, Überwachung zur Laufzeit und dem daraus resultierenden Angreifermodell. Diese Dissertation legt den Fokus auf die Geräteklasse der IES. Nach der Gegenüberstellung gebe ich einen Überblick über frühere Arbeiten aus dem Bereich der nutzerfreundlichen Security, der Themen dieses Forschungsgebiets sowie zentraler Erkenntnisse, die auf IES anwendbar sind.

Anschließend untersuche ich Nutzerinteraktionen und die sich daraus ergebenden Probleme hinsichtlich der nutzerfreundlichen Security in drei Netzwerken aus IES, an denen ich gearbeitet habe. Zwei von diesen Netzwerken erfassen Vitalparameter des Nutzers. Das erste Netzwerk erhöht die Sicherheit von Feuerwehrleuten im Einsatz und nutzt das IPv6 over Low power Wireless Personal Area Network (6LoWPAN) Kommunikationsprotokoll. Das zweite Netzwerk erfasst ein 3-Kanal-Elektrokardiogramm bei älteren Patienten und nutzt Bluetooth Low Energy (BLE). Das dritte Netzwerk besteht aus speziellen Schrauben, welche z.B. in kritischer Infrastruktur eingesetzt werden können. Sie können ihre Vorspannkraft selbst überwachen und mittels ultra-high frequency radio-frequency identification (UHF-RFID) und dem MIOTY-Funkprotokoll vermelden, wenn sie sich lockern. Ich modelliere das Verhalten der Nutzer bei der Interaktion mit diesen Netzwerken und leite davon Probleme aus dem Bereich der nutzerfreundlichen Security ab, die insbesonders während der Inbetriebnahme und der Überwachung des Netzwerks auftreten.

Anschließend präsentiere ich ein neuartiges Verfahren, um simultan Firmware und kryptografische Geheimnisse an IES zu verteilen. Hierzu habe ich einen intelligenten Faraday'schen Käfig (Faraday Cage (FC)) entworfen, welcher over-the-air (OTA) Firmware Updates durchfüehren kann. Die Nutzerinteraktion beschränkt sich auf das Platzieren der IES im FC, dem Befolgen von einfachen gesprochenen Anweisungen und der abschließenden Entnahme und Anbringung der IES am Zielort. Der FC führt alle security-kritischen Tätigkeiten automatisiert durch. Ich führe eine Nutzerstudie durch, deren Ergebnisse zeigen, dass es besonders für Nutzer mit geringem oder keinem technischen Hintergrund attraktiv ist, wenn sie durch Automatisierung aus der *Security-Loop* entfernt werden.

Basierend auf diesen Ergebnissen untersuche ich, wie diese nicht-technischen Nutzer in die Lage versetzt werden können, ihr Funknetzwerk zu überwachen, ohne dafür ausgebildet worden zu sein. Hierfür habe ich eine neuartige Lösung zur Überwachung von kabellosen Netzwerken entworfen. Diese kombiniert passives Mitschneiden des Netzwerkverkehrs mit einer Visualisierung in Augmented Reality. Der Nutzer zeigt mit der Kamera eines Smartphones oder Tablets (Handheld) auf ein Zielgerät. Das Handheld ist mit einem System zur passiven Erfassung von kabellosem Netzwerkverkehr verbunden. Auf dem Handheld wird der Netzwerkverkehr vom und zum Zielgerät als Überlagerung auf dem Kamerabild dargestellt. Damit kann der Nutzer das Netzverkverhalten beobachten und so z.B. in einem Mesh-Netzwerk wie 6LoWPAN das Hop-Verhalten der Pakete beobachten und defekte Geräte identifizieren. Frühere Lösungen erforderten, dass der Netzwerkverkehr an einem zentralen Punkt im Netzwerk mitgeschnitten wird, was Änderungen im Netzwerk erforderlich macht. Mein System kann hingegen durch die passive Erfassung des Netzwerkverkehrs auch in bereits operierenden Netzwerken angewendet werden, was meine Lösung auch für nicht-technische Nutzer zugänglich macht.

Mit dieser Architektur untersuche ich, ob die Augmented-Reality-basierte Visualisierung von Netzwerkverkehr auch dazu genutzt werden kann, Angriffe auf kabellose Netzwerke zu entdecken und dadurch den Nutzer im Bereich Security für Funknetzwerke zu schulen. Hierzu fëhre ich eine Studie mit 25 Probanden durch, darunter solche Probanden mit Erfahrung in der Security von Funknetzwerken sowie Probanden ohne technischen Hintergrund. Die Probanden beobachten ein 802.11 Wi-Fi (Wi-Fi) Netzwerk, auf das die typischen Angriffe *Denial of Service (DoS)*, *Address-Resolution Protocol (ARP) spoofing* und *Evil Twin* ausgeführt werden. Die Studie zeigt, dass auch nicht-technische Probanden durch die Beobachtung des Netzwerks die Angriffe sowohl entdecken als auch beschreiben köennen. Aus diesen Ergebnissen ziehe ich die Schlussfolgerung, dass die Visualisierung von Netzwerkverkehr mittels Augmented Reality eine leicht zugängliche Einführung in das komplexe und abstrakte Themenfeld der Security für kabellose Netzwerke darstellt. Daher kann dieser Ansatz auch in Schulen und Universitäten verwendet werden und ist geeignet, dort das Interesse der Schüler und Studenten zu wecken.

Zusammenfassend trage ich mit den Ergebnissen aus dieser Dissertation dazu bei, die Interaktion zwischen dem Nutzer und Funknetzwerken bestehend aus IES besser zu verstehen, insbesonders während der Inbetriebnahme und der Laufzeit der Netzwerke. Weiterhin präsentiere ich zwei Vorrichtungen und Vorgehensweisen, um den Nutzer bei der Inbetriebnahme und Überwachung der Netzwerke zu unterstützen. Zusammengenommen ermöglichen diese Ergebnisse es auch nicht-technischen Nutzern, ein sicheres Funknetzwerk aus IES über dessen gesamte Lebenszeit zu betreiben.

Acknowledgements

I would like to express my sincerest gratitude to Prof. Dr.-Ing. Georg Sigl for permitting me to pursue my Ph.D. at Technische Universität München (TUM) and for his past years of supervision. I would also like to thank Prof. Dr. Dieter Kranzlmüller for being my second examiner.

Thanks to Dr.-Ing. Johann Heyszl, former head of the Hardware Security department at Fraunhofer AISEC, for giving me the chance to pursue my research and for his scientific guidance.

I thank my amazing colleagues at Fraunhofer AISEC for their friendship, for valuable discussions and for providing feedback on my work. Thanks to the students who worked with me on this research.

Last but not least I would like to thank my parents and my wife for supporting me throughout working on this thesis.

Acronyms

6LoWPAN	IPv6 over Low power Wireless Personal Area Network.
ABE	Attribute-Based Encryption.
AES	Advanced Encryption Standard.
AP	access point.
API	application programming interface.
AR	augmented reality.
ARM	Advanced RISC Machine.
ARMS	augmented-reality based network monitoring system.
ARP	Address-Resolution Protocol.
BAN	body area network.
BLE	Bluetooth Low Energy.
BT	Bluetooth.
CGM	continuous glucose monitoring system.
COTS	commercial off-the-shelf.
CPS	cyber-physical system.
CPU	Central Processing Unit.
CS	Computer Science.
DDos	Distributed Denial of Service.
DH	Diffie-Hellman.
DHCP	Dynamic Host Configuration Protocol.
DoS	Denial of Service.
ECDH	Elliptic-Curve Diffie-Hellman.
ECG	electrocardiogram.
EEI	Electrical and Electronic Engineering.
EEPROM	electrically erasable programmable read-only memory.
EM	electromagnetic.
EUI	Extended Unique Identifier.
FC	Faraday Cage.

FOV	field of view.
GOD	gateway/onboarding device.
GUI	graphical user interface.
HCI	human-computer interaction.
HMAC	keyed-Hash Message Authentication Code.
HTTP	Hypertext Transfer Protocol.
HUD	head-up-display.
I^2C	Inter-Integrated Circuit.
I/O	input/output.
IC	integrated circuit.
ICD	implantable cardioverter defibrillator.
IEEE 802.15.4	IEEE 802.15.4 Low-Rate Wireless Networks.
IES	low-power wirelessly interconnected embedded system.
IIoT	Industrial Internet of Things.
IoT	Internet of Things.
IP	Internet Protocol.
IPC	industrial computer.
IS	Intelligent Screw.
IT	information technology.
JTAG	Joint Test Action Group Standard 1149.1-1990.
LED	light-emitting diode.
LLC	location-limited channel.
LPU	Local Processing Unit.
MAC	Medium Access Control.
MCU	microcontroller.
MITM	Man-in-the-Middle.
MMU	Memory Management Unit.
NFC	Near-Field Communications.
NTP	Network Time Protocol.
OOB	out of band.
OS	operating system.
OTA	over-the-air.
PCB	printed circuit board.
PGP	Pretty Good Privacy.

PKI	Public Key Infrastructure.
PPE	personal protective equipment.
PUF	physically uncloneable function.
QR	Quick Response.
RADIUS	Remote Authentication Dial-In User Service.
RAM	random-access memory.
RF	radio frequency.
RGB	red, green, blue.
ROP	Return-Oriented Programming.
RPL	Routing Protocol for Low power and Lossy Networks.
RSSI	received signal strenght indication.
RTOS	real-time operating system.
RTS	Request To Send.
SBC	single board computer.
SDR	Software Defined Radio.
SHA	Secure Hash Algorithm.
SMA	SubMiniature version A.
SNMP	Simple Network Management Protocol.
SNR	signal to noise ratio.
SPI	Serial Peripheral Interface.
SSH	Secure Shell.
SSID	Service Set Identifier.
SUS	System Usability Scale.
TCP	Transmission Control Protocol.
TETRA	Terrestrial Trunked Radio.
TRNG	true random number generator.
TTP	trusted third party.
UART	Universal Asynchronous Receive Transmit.
UDP	User Datagram Protocol.
UHF-RFID	ultra-high frequency radio-frequency identification.
UI	user interface.
USB	Universal Serial Bus.
Wi-Fi	802.11 Wi-Fi.
WPA3	Wi-Fi Protected Access version 3.
WSN	wireless sensor network.

List of Figures

List of Tables

Contents

1 Introduction

Computers are actually easy machines to secure: Just turn them off, lock them in a metal-lined room and throw away the key. What you end up with is a machine that is very secure, just not very usable. – [1]

1.1 Motivation

Advances in low-power sensing, embedded computing and wireless protocols drive the dissemination of low-power wirelessly interconnected embedded systems (IES) in industrial, agricultural, medical or household appliances. They collect data or drive actuators and thus interact with the physical world [2]. Failure or corruption of those devices and the data they send can have severe consequences ranging from impaired industrial productions to injury or death (Figure 1.1). For example, if medical devices such as implantable cardioverter defibrillators (ICDs) or insulin pumps accept packets forged by an attacker, the patient might be harmed severely [3, 4, 5]. Those examples demonstrate that IES must be secured, e.g., by authenticating data transmissions.

Securing IES is challenging due to both technical reasons and the users and their interaction with those systems. For example, during the *setup* of large wireless sensor networks (WSNs), both firmware images and cryptographic keys must be distributed among many IES with few or no input/outputs (I/Os). A security-conscious user would like to do this in-house to reduce the likelihood of the keys being stolen along the supply chain. However, supplying firmware and keys manually via a programmer and cable to a large number of IES is tedious and does not scale. Additionally, access to the physical programming interfaces might be restricted due to sealed enclosures. Hence, due to technical limitations, even for security-conscious users setting up a secure WSN is challenging.

During run time, networks must be *maintained*. For example, sensor node behavior must be observed to discover attacks on complex multi-hop routing protocols [6]. However, due to the lack of remote management protocols such as Secure Shell (SSH) in IES, network behavior can only be observed *passively*. This comprises, for example, captur-

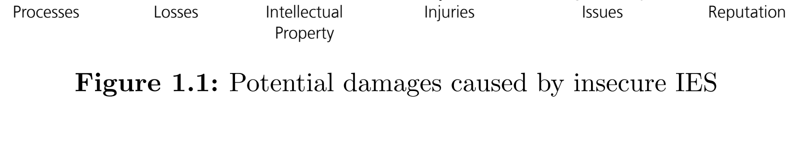


Figure 1.1: Potential damages caused by insecure IES

ing and inspecting network traffic using tools such as `wireshark` or `tcpdump` which are challenging to master [7, 8]. Those network monitoring tools merely provide an abstract view of the network by providing text, topological graphs or diagrams which contain lots of information and require extensive training to make use of. Thus we can draw the dire conclusion that wireless connectivity increases the attack surface of IES without providing defenders adequate capabilities to counter attacks.

In addition to technical limitations, *users and their interaction with IES* impair secure operations, too. With increasing performance of IES and the connection of more and more devices, inevitably more non-technical people operate IES. For example, in the medical domain, IES are used by elderly patients without technical background and, yet, their lives depend on the correct and secure operation of the medical device.

Unfortunately, securing IES involves additional manual actions, whose necessity is often not understood by users. For example, having to read a passkey from one device and manually entering it in a second device to perform device pairing, as in Bluetooth Low Energy (BLE) *Passkey Pairing*, is tedious [9]. If possible, users will go the path of least effort and try to ignore or bypass security measures in favor of convenience. With BLE, users might prefer user-friendly but insecure setup mechanisms like *Just Works Pairing*, where Elliptic-Curve Diffie-Hellman (ECDH) with no authentication is performed. This is prone to Man-in-the-Middle (MITM) attacks, however, non-technical users are not concerned about such attacks [10]. As a dire conclusion, SCHNEIER stated that in securing a system humans are the weakest link [11].

To remedy this, the human element in security has been investigated in the research field *usable security*, which joins methods from human-computer interaction (HCI) and information security. Researchers learned that users play a crucial role in securing computer systems. They found that attackers focus primarily on the human link in a security chain, while the designers of those systems did not take this into account. Slowly, engineers understood the role of the user in the security loop of a system, i.e., all security-related activities in the lifecycle of a system. They learned that they have to design systems such that users actually can achieve their primary goal (e.g., read e-mail) securely without security controls getting in their way [12, 13, 14].

Previous research has investigated the *device pairing problem* and *security administration*, which correspond to the security-critical phases *setup* and *maintenance* in the lifecycle of networks of IES. The former deals with establishing trust between two parties which have no prior notion of each other. The latter is concerned with keeping devices secure through their lifecycle.

The challenge of making IES both secure and usable is their lack of I/O such as buttons or displays. I found, that previous works on device pairing are often based on unrealistic assumptions about the capabilities of IES, e.g., the IES having auxiliary I/O interfaces. Further I found that most results on security administration were derived from studies not targeting IES.

1.2 Research Objective and Approach

With this thesis I contribute to conquering above shortcomings by proposing user-friendly mechanisms to set up and maintain secure networks of IES. I investigate the hypothesis "given the right tools, even non-technical users are capable of setting up and maintaining networks of IES". More specifically, I exlore the central research questions raised in previous literature on usable security research [15]:

1. How to remove the user from the security loop?
2. How to permit intuitive interaction with IES?
3. How to educate users in security?

To answer each of those questions, I propose a novel method and accompanying apparatus and implement those in real hardware. I evaluate the feasibility of those approaches by performing structured user studies utilizing established usability testing methods. These studies were published and presented at international conferences after having been subject to a peer review. Results from one study lead to a European patent grant (EP3557897A1)[1].

1.3 Structure of this Thesis

In Chapter 2, I define the scope of this thesis and give a brief history of usable security and the main findings of this research field. The subsequent chapters comprise content from my original contributions presented at academic conferences.

- In Chapter 3 I discuss practical security and usability problems in two body area networks (BANs) and a structural integrity monitoring network I worked on.
- In Chapter 4 I propose and evaluate a novel approach to set up a network of IES by leveraging a Faraday cage and over-the-air (OTA) updates. With this, IES can be configured intuitively while removing the user from the security loop.
- In Chapter 5 I demonstrate, how to interact with networks of IES intuitively using an augmented-reality based network monitoring system (ARMS). I propose an architecture leveraging multiple distributed traffic capture devices, show how those information can be joined and how they can be presented such that they are understandable by non-technical users.
- Finally, in Chapter 6 I show, how the ARMS application can be used to educate users in wireless network security.

Chapter 7 concludes this thesis.

[1]Please see Appendix A.2 for a complete list of publications.

2 Background

2.1 Scope and Specific Challenges: Low-Power Interconnected Embedded Systems vs. "Internet of Things Devices"

In this thesis, I focus on low-power wirelessly interconnected embedded systems (IES) and differentiate those from the devices commonly referred to as "Internet of Things (IoT) devices". I compiled the main differences of those device classes in Table 2.1. In the following, I discuss the key takeaways from the differentiation and how they relate to this thesis.

Table 2.1: Comparison: IES vs IoT devices

	IES	**"IoT device"**
Platform	ARM Cortex-M-based MCU, e.g., nRF52 series (*Nordic Semiconductor*), CC1352 series (*Texas Instruments*)	ARM Cortex A9, A53 or A72-based processors, e.g., BCM2837B0 (*Broadcom*), i.MX series (*NXP Semiconductors*)
	Hybrid: Xtensa LX7-based ESP32 (*Espressif Systems Inc.*)	
Computational resources	Single core at 64 MHz, 512 kB Flash memory, 64 kB RAM	Quad core up to 1.5 GHz, up to 64 GB Flash memory, up to 4 GB RAM
	ESP32: Single or dual core up to 240 MHz, up to 320 kB RAM, up to 16 MB Flash memory	
Power supply	battery-powered	mains powered
Operating system	None or RTOS	UNIX-based
Connectivity	BLE, IEEE 802.15.4, LoRAWAN, MIOTY. Connection to TCP/IP network via gateway, often unidirectional communications	Direct bidirectional connection to TCP/IP network over Ethernet/Wi-Fi
	ESP32: BLE and Wi-Fi	
Control interfaces	LED, button if applicable	Command shell exposed over SSH, keyboard and display if applicable
Applications	WSN for environmental monitoring; medical wearable devices	Smart home and IIoT appliances

Platform IES typically comprise an Advanced RISC Machine (ARM) Cortex-M3 or M4 MCU, while IoT devices have an MCU from the ARM Cortex-A series of processors. The Xtensa LX7 architecture-based ESP32 MCU resides in between those performance ranges.

Computational resources and power supply IES have less computational resources than IoT devices, i.e., they have only single-core processors and their Flash and RAM memory size is in the range of kilobytes. In contrast, IoT devices might have quad core Central Processing Units (CPUs) and Flash memory and RAM in the range of gigabytes. The ESP32 with its dual core CPUs and several megabytes of Flash memory lies in the middle.

IES typically run on battery instead of mains power. To extend battery life IES need to reside in a deep sleep state most of the time. Thus they **should avoid performing expensive calculations**, such as those involved in, e.g., asymmetric encryption schemes or signal processing. If such operations have to be performed nevertheless, hardware accelerators should be employed to perform calculations efficiently and quickly and thus let the IES return to a deep sleep state quickly.

Operating system Due to low computational resources and the lack of a Memory Management Unit (MMU) IES do not use a Linux-based operating system (OS). Instead, they typically utilize a RTOS such as *Zephyr OS*[16] or *FreeRTOS*[17] which typically include a wireless stack and a scheduler.

These OSes typically lack the *Command Shell* typically used to administer Linux-based systems remotely. While implementing a shell in a RTOS-based application is possible [18, 19], this is typically not done to preserve scarce memory. Additionally, as will be discussed in the next paragraph, due to limited connectivity of the IES a shell is often of little use for controlling the IES remotely. Thus, while IES may employ SSH as secure transport layer protocol, **commonly there is no mechanism to control the IES remotely.**

Connectivity To preserve energy, IES utilize low-power wireless protocols such as IEEE 802.15.4-based (e.g., ZigBee), LoRAWAN, MIOTY and BLE. To connect those devices to TCP/IP-based networks, gateways are used.

As stated before, IES are typically battery-powered. To extend battery life, their energy consumption must be minimized. Sending messages of the wireless channel contributes significantly to the overall energy consumption of the IES, despite using one of the above-mentioned low-power wireless protocols. Thus, to preserve energy we want the IES to send messages at low transmission power and to minimize the number of messages sent. **Hence, sending status or control messages should be avoided to preserve battery.**

To further preserve energy, IES should turn off their radio as often as possible. During this they can not receive *downlink* messages, i.e., messages sent from the gateway to the IES. On top of that, protocols such as LoRAWAN and MIOTY have only limited downlink-capability by design. Hence, due to energy and protocol constraints, unidirectional connectivity from the IES to the gateway is preferred over bidirectional communications.

IES can be deployed in environments with harsh radio frequency (RF)-conditions causing fluctuating connectivity to the network control server. E.g., in the vicinity of the human body, propagation of radio waves in the 2.4 GHz band are hindered by the high moisture content of the human body [20]. In industrial environments, strong RF interference can block communications by flooding receivers with too high noise levels.

Thus, as discussed in the previous paragraph, while running a shell on the IES is feasible, energy constraints, utilizied communication protocols, and harsh deployment environments **prevent reliable remote control of IES**.

Control interfaces As discussed in the previous paragraph, IES typically do not have remote control interfaces. Depending on the application, they might also have limited to no other interfaces for the following reasons: For applications such as large-scale WSNs, e.g., to supervise factory operations, IES should be cost-efficient. Thus, adding input/output (I/O) such as buttons and displays to IES is unfeasible from an economic point of view.

If deployed remotely in uncontrolled environments, e.g., in habitat monitoring, IES require tightly sealed and robust enclosures. Otherwise, moisture might find its way inside the enclosure [21]. Furthermore, IES might be subject to unexpected treatment by humans unplugging or animals gnawing at them [22, 21]. Even if there were I/O, due to the remote placement physically approaching the IES and reading its status would be unfeasible.

If used in a body area network (BAN), IES must be disinfectable and thus sealed tightly as well. Membrane keys are prone to become leaky and should be avoided. Access to wired interfaces such as Universal Asynchronous Receive Transmit (UART), Serial Peripheral Interface (SPI) or Inter-Integrated Circuit (I^2C) is blocked by the enclosure. **Thus, wireline I/O interfaces are often not available or unsuited for interacting with IES.**

Applications Embodiments of IoT devices are typically larger smart home devices, such as the *Amazon Echo Show Gen 5*, a "smart display" or a "smart fridge" by *Samsung*, [23, 24]. We also find them in industrial applications ("IIoT") to, e.g., connect legacy wireline sensors to the Internet over a wireless link [25].

The integration of IoT devices in existing networks is easy, because they natively use TCP/IP-based communications and this infrastructure, such as switches, routers and wireless access points (APs), are already present in many facilities and the homes of private users.

In contrast, IES and the low-power transmission protocols utilized by those such as LoRAWAN and MIOTY typically require a dedicated gateway, which converts those transmission protocols to TCP/IP-based communications. Due to being practically unidirectional communication protocols, they are typically used in sensor applications, where sensor nodes push metered data to a cloud application [26, 27].

Medical devices worn at the body are often Bluetooth-based IES. Examples include insulin pumps, continuous glucose monitoring system (CGM) or electrocardiogram (ECG) recorders [28, 29, 30, 31]. Those devices typically meter physiological parameters such as blood glucose levels or the heart rate, or deliver drugs, such as insulin. This type of devices send data over the Bluetooth-link to receiving devices such as the smart phone of the patient. In some applications, the smart phone acts as a gateway and forwards received data, e.g., to the cloud. Because Bluetooth is ubiquitous in modern smart phones and tablets, this type of IES can be integrated easily in existing infrastructure and thus, those applications are more wide-spread than the LoRAWAN and MIOTY-based IES.

Summary In this section, I have shown, how the IES on which I focus on in this thesis differ from the devices commonly referred to as "IoT devices". I have discussed, how and why IES typically lack I/O capabilities. Hence, novel ways of interacting with IES are needed. Later in this thesis, I introduce to interaction mechanisms to securely set up and maintain networks of IES.

As IES interact with the physical world[1], they also interact with their *users*. In fact, more and more applications such as BANs collecting critical medical data have a deep influence on the life of their users. This pervasiveness inevitably causes the number of non-technical users interacting with IES to increase. These users lack the technical background to understand the security threats that lie within these technologies, but their data need to be protected nevertheless. Hence IES need to be designed such that non-technical users can interact with them securely. Else, the user must be considered an (involuntary) attacker. With respect to this, in the next section I am going to introduce attacker models applicable to networks of IES.

2.2 Attacker Model: Users can be Attackers, too

The "classical" attacker model Generally, in IES we consider the *Dolev-Yao* attacker model, in which the attacker has full control over the wireless channel [32]. This means, that the attacker can eavesdrop, replay and modify any message transmitted. Also, he can inject arbitrary messages. Further, he can jam the wireless channel on the physical and upper layers or delay messages in order to launch a denial of service attack. The attacker is capable of performing computations in polynomial time and we assume him to be present during setup and runtime of the wireless network.

Attack surface: IoT devices vs. low-power IES In practice, the attack surface of IoT devices and low-power IES differs. Firstly, as IoT devices are directly reachable from a TCP/IP network and because they expose remote control interfaces such as a command shell over SSH, they can be **hijacked more easily**. Secondly, because they

[1]Due to this, they are also called *cyber-physical systems (CPSs)*, however, I will not use this term for the remainder of this thesis.

are more powerful than the low-power IES, the attacker has **more options how to utilize a hijacked device**. Because the Linux OS on IoT devices already brings a large number of tools, scripting capabilities and interpreters (such as the `Shell` or the `Python` interpreter), the attacker can **tailor the hijacked device towards their needs** conveniently. For example, he can use the device to perform mining of cryptographic currencies. He might also make the device part of a botnet, which launches Distributed Denial of Service (DDos) attacks[2]. Because cryptographic currencies can be exchanged for real money and botnets can be rented, hijacking an IoT device often yields direct financial benefits to the attacker.

In contrast, due to the absence of a network-accessible `Shell`, IES might be hijacked, e.g., by a Return-Oriented Programming (ROP) attack or by replacing its firmware using, e.g., over-the-air (OTA) firmware updates. Taking the first approach, the attacker would need to turn the captured IES in a weapon by utilizing ROP gadgets. This is very tedious compared to having full access to a IoT device and, at the time of writing this dissertation, we have yet to see a practical implementation. The second approach using OTA updates has been demonstrated and hence must be considered a realistic threat [33].

As IES are often part of a control loop and either sense data or drive actuators, they might be subject to attacks such as **eavesdropping, packet replay, packet injection and Denial of Service (DoS)**. The attacker might utilize eavesdropping to learn a secret production recipe, inject packets to alter a production process [34] or to halt the process by overloading the IES [35, 36].

In conclusion, IES are typically exposed to other attacks executed with different motifs compared to attacks on IoT devices. Many of those attacks on IES can be thwarted by employing well-known security controls such as message encryption and authentication and signed OTA updates. However, due to being utilized by non-technical users and due to the lack of I/O capabilities, IES must deal with security issues introduced by the legitimate, but potentially unskilled user.

Extended attacker model: The user as attacker Users must be considered both *voluntary* and *involuntary* attackers. If users sense a reward, they might be tempted to exploit their own devices. For example, health insurance companies increasingly offer financial benefits for policyholders who demonstrate a healthy life-style. Policyholders can demonstrate this by, e.g, wearing a fitness tracker which tracks and shares the distance walked per day with the insurance company. However, [31] have demonstrated ways to manipulate fitness trackers directly at the *sensor*, e.g., by mounting them to the tires of cars. With this fitness trackers can be tricked into counting additional "steps", which the user never walked.

This type of attacks can be executed even by non-technical users. Defending IES against this type of attack is extremely challenging, as the user can be considered an

[2]See, for example, this script for hijacking a Raspberry Pi over SSH and creating a botnet: `https://github.com/zachRudz/60467_raspPiBotnet`, last accessed 24 August 2021

authenticated attacker with physical access and generally unlimited time to conduct an attack.

Additionally, the user can cause *involuntary* attacks by making errors, which lead to security issues. Those can cause similar damages as a dedicated attack would.

For example, [37] found that users send unencrypted e-mails because they do not understand the encryption tool Pretty Good Privacy (PGP) and its user interface [38]. The authors conclude that a computer system needs to communicate a clear conceptual security model to the user so they are able to understand security functionalities.

As a second example, during the Bluetooth pairing procedure, devices are enumerated by their friendly name, e.g., *AHeadset* and sorted by their received signal strenght indication (RSSI). Due to the lack of input capabilities at the headset, the smart phone and the headset utilize Bluetooth (BT) *Just Works* association model. Here, an unauthenticated Diffie-Hellman (DH) key exchange is performed. In densely populated environments, multiple BT headsets with the same friendly name might be present and the user typically chooses to connect to the device shown at the top of the device list. Before actually using the headset, they have no way of knowing whether they connected to the correct headset. They made (reasonable) assumptions, i.e., that the device enumerated at the top is in fact the device they want to connect to, and their device did not provide assistance in supporting or challenging those assumptions. Hence, in this case the user made an error which could lead to a security issue if the device they accidentally connected to is controlled by an attacker.

This shows, that when designing a system to be truly secure, we have to take into account user errors and their potential impact on security.

Summary In this section, I have introduced an attacker model applicable to IES. This attacker model incorporates the user as involuntary attacker, if the computer system they are dealing with is overly complex and non-comprehensible. Fortunately, this can be remedied by appropriately designing systems. The interaction between users, devices and software with the goal of providing a secure and user-friendly experience is studied in the field of *usable security*, which I am going to introduce in the next section.

2.3 Usable Security

What constitutes a *usable* system? ISO 9241 defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in the specified context of use" [39].

The research field *usable security* is hence concerned with building computer systems, which fulfill above criteria while also being secure in the sense that they protect data and resources "from accidental or malicious acts, usually by taking appropriate actions" [40]. To do so, it combines methods and findings from human-computer interaction (HCI), information security and privacy. It dates back to 1975, when Saltzer and Schroeder identified the *psychological acceptability* as crucial for building and maintaining secure computing environments [12][3]:

> It is essential that the human interface be designed for ease of use, so that **users routinely and automatically apply the protection mechanisms correctly**. Also, to the extent that the user's **mental image of his protection goals matches the mechanisms he must use**, mistakes will be minimized. If he must translate his image of his protection needs into a radically different specification language, he will make errors.

In 2003, the National Academy of Engineering requested to "give computer end-users **security they can understand** and privacy they can control" [14, p.23]. And in 2009, the Department of Homeland Security demanded [41, p.90]:

> Security must be **usable by persons ranging from non-technical users** to experts and system administrators. Furthermore, systems must be **usable while maintaining security**. In the absence of usable security, there is ultimately no effective security.

Those quotes can be summarized to the following statement: If applied in a comprehensible and well-aligned manner, security does not impair usability.

In this section, I summarize key findings from previous works from the usable security domain. Most of those results stem from studies, where study participants interacted with desktop computers or smart phones. This is no surprise as previously those devices were more prevalent than IES. In Chapter 3 I am then going to show that although the models of users and their actions are applicable to networks of IES, actually establishing such user-friendly networks is challenging.

2.3.1 Modelling Human Behaviour for Usable and Secure Systems

Understanding how users interact with technical systems is crucial in designing systems that are both secure and usable. Whitten and Tygar found five properties which explain, why building usable and secure systems is challenging [37]:

Property 1: Unmotivated Users For most users, security is a secondary task: "People do not generally sit down at their computers wanting to manage their security; rather, they want to send email, [...] and they want security in place to protect them while they do those things." [37]. Users always face a conflict between being secure (e.g., having to use long and complex passwords) and getting their work done (e.g., not spending tremendous amounts of time to unlock their computer) [42].

Property 2: Abstraction Users find it difficult to deal with security properties, which are very abstract sets of rules.

[3]Highlighting in the quotes was done by the author of this thesis.

Property 3: Lack of feedback Users need good feedback to help them make good security decisions and prevent security errors. However, providing good feedback for security management is difficult, as the type and amount of information shown to users must be considered carefully. As I am going to discuss in Chapter 3, this is especially challenging with IES, as they lack I/O capabilities and thus means to display meaningful information to the user.

Property 4: Barn Door It must be assumed, that if a secret has been left unprotected once, it has already been utilized by an attacker. For example, if a secret key is transmitted in clear text only once, it could have been eavesdropped and this secret must be considered compromised.

Property 5: Weakest Link The security of a system is only as strong as its weakest component. As security software can not always decide as the user would do, users needs to make security decisions for themselves. For example, a user might decide to proceed browsing a web site despite the web browser having issued a warning that the connection to this web site is insecure. The user might also choose to open an office document despite the office software warning that this document contains executable macros which might be potentially harmful. This is especially grave, as system designers thus must consider an *authorized user* as adversary, who, while not intending to cause harm, has bypassed perimeter defenses at least.

2.3.2 Designing Usable and Secure Systems

With these five properties in mind, general guidelines for designing systems that are secure, usable, and minimize the damage the user can do as involuntary attacker were developed. In the following, I am going to discuss those before showing, why it is difficult to apply those on IES.

Successful designs promote users to achieve their desired security goals, while flawed designs impede even very motivated users to operate systems securely [43]. As stated before, users want to achieve a certain goal while forced to make security decisions at the same time. CRANOR coined the term *security loop* for security decisions or activities, of which the user inevitably is part of [15]. To design a secure and usable system, they proposed a three-tier approach:

Principle $\boxed{1}$: Remove the user from the security loop Whenever possible, a system should be designed such that the user is removed from the security loop and thus can not make security errors. To this, the role of non-malicious humans in a secure system needs to be analyzed and potential failure modes must be identified. A common fallacy here is that developers do not take into account the wide range of users, their skill levels and their expectations. In turn, developer oftentimes design systems which they consider to be usable for themselves, but not for the actual user. As a baseline approach fail safe default configurations, e.g., unique and random passwords, should be applied to every system. User interaction beyond this is covered by Principle $\boxed{2}$.

In Chapter 4 I demonstrate a practical solution to remove the user from the security loop while setting up a network of IES.

Principle $\boxed{2}$: Permit intuitive interaction As fully removing the user from the security loop can not be achieved for most systems, a system should be designed such that users can interact with it intuitively.

A usable system lets users achieve their goal without placing barriers, as those lead to errors [44, 45]. The two main types of errors are *slips* and *mistakes*. Slips occur when the intention of the user was correct but the error was made during execution [46]. For example, due to cost and space constraints a MCU system might only have a single LED and a single pushbutton as I/O. The user must be able to derive the state of the MCU (e.g., connecting, connected, measuring, idle) from the LED and remember the effect a button press has in this particular state (e.g., move to the next state or take a measurement). A slip occurs, if the user misinterprets the state the system is in. In this case, the user intends the button to do perform a particular action (e.g., take a measurement) but due to the system being in another state, the system does not respond as the user intended.

A mistake is made when the intended action itself was an error [46]. This happens, for example, if due to a lack of knowledge the user makes errors in the security configuration of a system, leaving the system vulnerable.

The probability for errors inevitably increases in complex systems, which are hard to configure, manage, maintain and implement correctly, and the complexity of systems increases constantly [47]. Developers can either expose this complexity to the user or try to hide it using abstractions. As users have shown to not understand the underlying security principles of, e.g., e-mail encryption [37], typically abstractions are used which, however, bring their own challenges: To be considered intuitive, the actual process of user interaction must match the user's perception (or mental image) of that process [12]. Users must be able to understand that a particular event resulted from a particular action (task-action-model). This means, that user actions must be both *visible* and *undoable*. If a user action leads to a security issue, warnings must be issued by the system. Those warnings must be directly relatable to user action, i.e., those warnings must appear in the right context and comprise clear wording. Here, the challenges of using too much abstractions become apparent: How to issue warnings on actions the user is not aware of having committed, as the security-related activity is hidden by the abstraction?

In Chapter 5 I propose an architecture for a novel augmented reality (AR)-based network traffic visualization system, which permits intuitive interaction with IES.

Principle $\boxed{3}$: Educate users In addition to the above, users must be taught how to perform security-critical tasks. Users have to be treated as partners in the endeavor to securing systems. In practice, however, they are often treated as enemies, e.g., supposedly undermining the security policies of their company [13]. It was found, that this is caused by users being provided little to no information on the rationale behind security measures, as, e.g., company security departments consider users inherently insecure.

[13] thus demanded, that the rationale behind security measures is *explained* to users to make them partners in defending information technology (IT) systems.

This is challenging because more and more non-technical users, i.e. users who do not have received training in wireless networking and/or security, are operating IES [1, p. 48], for example:

- WSNs in factories are likely to be maintained by servicemen, who are trained mechanics
- WSNs used in agriculture and forestry are tended by farmers and forest rangers
- BANs used to monitor physiological conditions are likely be used by elderly patients

While security training might be provided to those user groups, we are yet to see attacks on IES in the field. Hence, users do not expect to be attacked, thus lack security awareness. Even if there were programs to teach, e.g., wireless network security to the public, currently, there is a lack of affordable and intuitive setups to teach those in an accessible way.

In Chapter 6 I demonstrate, how users can be educated in wireless network security utilizing a novel AR-based network traffic visualization.

2.4 Summary

In this chapter I differentiated between low-power wirelessly interconnected embedded systems (IES) and the more powerful device class commonly referred to as "IoT devices", which is out of scope of this thesis. Next, I introduced the research field of *usable security*, its history, and the current state of knowledge concerning the modeling of human behavior and how to design systems, which are both secure and usable. Most of those design guidelines stem from previous works, in which interactions between users and smart phones or desktop computers were studied. Unlike IES, those have plenty of I/O capabilities to support user interaction. IES bring their specific challenges, which I am going to discuss in the next chapter.

3 Usable Security in Networks of Interconnected Embedded Systems

In this chapter, I present three networks of low-power wirelessly interconnected embedded systems (IES) I have worked on. Two of those are *body area networks (BANs)* and were published at academic conferences [1] [2]. The first BAN monitors physiological parameters of firefighters and must be ready to use with *no user interaction at all.* The second BAN records an electrocardiogram (ECG) and is designed to be used by elderly patients.

The third network consists of *Intelligent Screws (IS)* which monitor the preload force of the screw while being attached to critical infrastructure such as bridges and wind turbines. This is work in progress and has not been published before.

I briefly present the original content of those publications on BANs including advancements in system design after the original publication. I further present the current state of the design of the IS network. Subsequently, I revisit the design of those three networks under the usable security lens and study user interactions with the IES. Special challenges constituted by those three networks are non-technical, possibly sensory and motor impaired users, as well as challenging and labor-cost intensive environments.

Based on those observations, I motivate the need for solutions for user-friendly setup and maintenance of IES, which I am then going to introduce in later chapters of this thesis.

3.1 Case-Study 1: A Body Area Network for Protecting Firefighters in Indoor Operations

System Description and Application Performing heavy physical work in burning buildings while wearing personal protective equipment (PPE) and additional gear exerts extreme stress on the cardiovascular system of the human body (Figure 3.1.). Outfitting the PPE with connected sensors creates a BAN. This BAN continuously monitors the

[1]

[20]: *Marco Dietz, Martin Striegel, Robert Weigel, and Amelie Hagelauer. A new heat-warning-system based on a wireless body area network for protecting firefighters in indoor operations. In* 2018 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNet)*, pages 34–37, Anaheim, CA, January 2018*

[2]

[30]: *Georg Bramm, Matthias Hiller, Christian Hofmann, Stefan Hristozov, Maximilian Oppelt, Norman Pfeiffer, Martin Striegel, Matthias Struck, and Dominik Weber. CardioTEXTIL: Wearable for Monitoring and End-to-End Secure Distribution of ECGs. In* IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 2021*

Figure 3.1: Firefighter squad preparing an indoor assault under medium smoke. The author of this thesis is to the bottom right.
Image courtesy: Freiwillige Feuerwehr Freising, license: CC BY-NC-ND 3.0 DE

vital parameters of the firefighter and the remaining pressure in the compresed air cylinders and transmits those information to the command vehicle. Under an impending circulatory collapse or lack of oxygen, the firefighter can be ordered to withdraw from the mission by the group leader using spoken communications over radio.

A novel aspect of the original work is the utilization of four heat sensors, which are distributed over different positions of the body. I found that if only some temperature sensing nodes measure a significant increase in temperature an external heat source must be nearby. This provides a coarse directional detection of heat sources and by this the firefighter can be made aware of dangerous situations early.

The second novelty of the original work is the thorough investigation of wireless communications in the vicinity of the human body. Through simulation and measurements we found optimal positions for the sensors such that for all communication paths received signal strenght indication (RSSI) is maximized and thus reliability of communications optimized.

Communication Scenario The BAN consists of a *concentrator*, which also reads sensors, and five additional sensor nodes placed around the human body as shown in Figure 3.2. Data from sensor nodes are transmitted to the concentrator using IPv6 over Low power Wireless Personal Area Network (6LoWPAN) over IEEE 802.15.4 Low-Rate Wireless Networks (IEEE 802.15.4) operated at 2.4 GHz. At the concentrator, data are processed and transmitted to the command vehicle in the 868 MHz band.

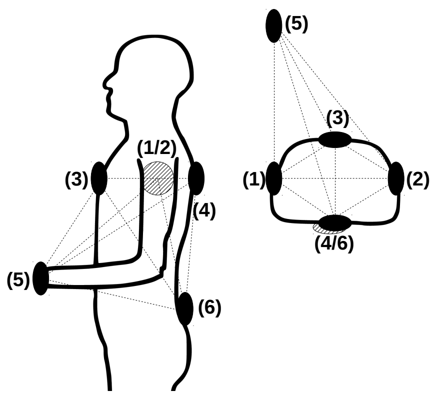


Figure 3.2: Device placement at the human body and possible transmission paths: Sensor nodes (1) to (4) measure skin temperature. (3) doubles as concentrator. (6) measures pressure in the oxygen cylinder. (5) is an auxiliary sensor placed in the glove.

The efficiency and reliability of the network depends on the positioning of the sensor nodes and the concentrator at the body of the firefighter. For electromagnetic (EM) wave propagation at 2.4 GHz the human body is challenging due to its high water content and its stacked structure of tissues (e.g., skin, muscle, fat). We purposely chose this frequency to model and measure path losses by creating an abstract geometrical body model in conjunction with an empirical model for the transmission losses. From this, we found that the optimum position of the concentrator is at the chest and that this position permits reliable communications to all sensor nodes. We conclude that by careful layout of the devices constituting the BAN even communications at 2.4 GHz can be used reliably.

Security Requirements *Availability* of the network is the paramount security goal, on a par with *authenticity* of transmitted and stored data: The firefighter might be ordered to withdraw because the transmitted physiological data are missing or were manipulated such that they appear alarming. Those data are part of an audit trail, which must be secure against manipulation. Otherwise data could be manipulated *after* a mission, e.g., to cover up wrong decisions made during the mission.

Transmitted data contain health information. Based on those, decisions are made, which can have severe medical consequences. Thus law enforces that those data are kept *confidential* both in transit and storage.

User Interaction After unboxing, the sensor nodes and the Local Processing Unit (LPU) must be supplied with configuration data and cryptographic material. Next, for every firefighter, five sensor nodes must be assigned to the network created by one LPU.

The sensor nodes and LPU could be configured by the manufacturer, however, this means entrusting cryptographic material such as keys to the manufacturer. Additionally, those information must be transferred to the fire station securely. Hence, it were more desirable to configure devices in-house, however, the servicemen at the fire station are no

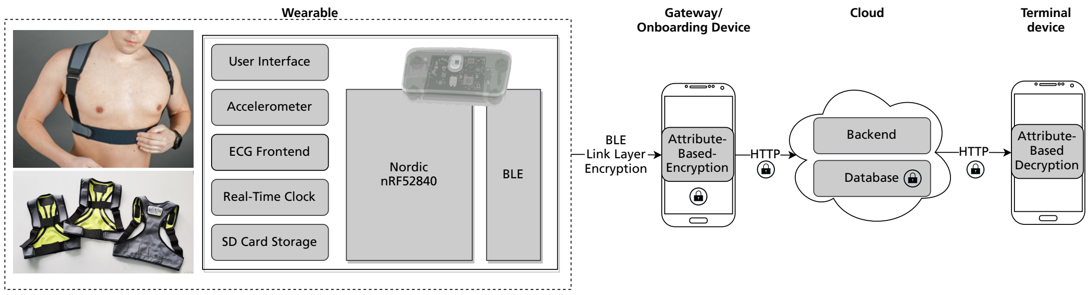


Figure 3.3: Wearable ECG: Data sharing architecture

security experts. If they make errors during setup, security of the system is compromised because the serviceman are part of the security loop.

The primary task of the firefighter is the actual mission, e.g., search and rescue. The BAN has thus been designed to be transparent to the firefighter, i.e., to not distract them from this primary task. Hence, after being placed in the PPE, sensor nodes remain in deep sleep until they sense motion, then they start taking measurements and transmitting them to the command vehicle and the group leader.

Because firefighters do not interact with the sensor network themselves directly, sensor nodes do not require input/output (I/O). Rather, the group leader receives and analyzes data sent by the network and uses voice radio (e.g., Terrestrial Trunked Radio (TETRA)) to provide information to the firefighter based on those data. If the sensor network fails or reports suspicious data, only the group leader will notice and instruct firefighters to fall back to manually reporting their status via voice radio.

Not displaying sensor readings to the firefighter directly is intentional. Due to environmental stimuli such as pursuing the primary task under interference such as stress, heat, noise, and darkness, firefighters would likely not be able to process information from the sensor network issued over an additional user interface such as a head-up-display (HUD). For this reason we utilize spoken commands issued over radio from the group leader, as this is the traditional communications channel firefighters were trained to use and follow orders heard there.

3.2 Case-Study 2: Wearable for Recording and End-to-End Secure Distribution of Electrocardiograms

System Description and Application For long-term monitoring and recording of cardiac events portable sensor units, so-called Holter ECGs, are used. They utilize adhesive electrodes and cables to record the ECG signals. This leads to skin irritations and severe restrictions in the patient's movement.

To overcome those limitations several Fraunhofer institutes designed a wearable ECG integrated into a t-shirt. The wearable device comprises an adjustable vest with four dry electrodes and an attached sensor module with an ARM Cortex-M4 Bluetooth Low

Energy (BLE)-enabled microcontroller (MCU) (Figure 3.3). It can be used for convenient long-term medical-grade acquisition of ECGs at a sampling rate of 400 Hz.

Communication Scenario The sensor module attached to the adjustable vest transmits data secured on the link layer to a gateway/onboarding device (GOD) over BLE. At the GOD data are encrypted on the application layer using Attribute-Based Encryption (ABE) and then sent to a cloud storage using Hypertext Transfer Protocol (HTTP). From there, authorized physicians and the users themselves can retrieve encrypted data via HTTP and decrypt them at their terminal devices (e.g., smart phone or laptop) to view and analyze those data.

The theory behind ABE is not in scope of this thesis and the reader is kindly referred to the seminal paper [48]. The key idea of ABE is that there are *multiple* private keys for a single public key used to encrypt data. Users specify who is permitted to view their medical records. For example, if the user specifies that only themselves and their attending cardiologist are permitted to view the data, those data are encrypted under this *access control policy*: `user` $\vee$ `(is_cardiologist` $\wedge$ `attending)`. If a physician can prove that he is the user's attending cardiologist and thus satisfies all attributes, he is issued a decryption key.

Using ABE in this application has the benefit that the patient is in charge of their data. Data only need to be encrypted once and every legitimate recipient has their *own private key* used to decrypt those data. Most importantly, data are encrypted between the data producer and the data consumer: In case of a breach of the cloud database, the attacker can only retrieve encrypted data.

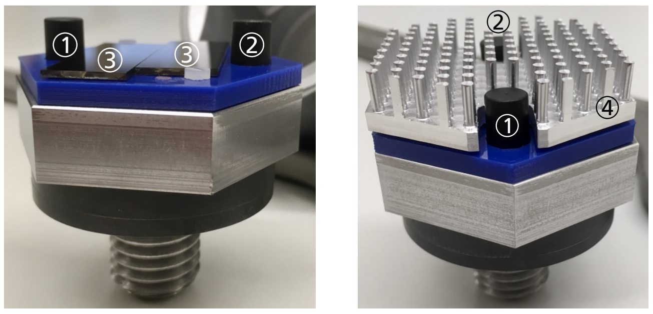
Security Requirements As transmitted data contain health information they must be kept *confidential* during transmission and at rest.

Authenticity and *Integrity* of the transmitted data are paramount. If data could be manipulated to indicate a cardiac failure while the patient actually is in good condition, an ambulance could be dispatched to the patient unnecessarily. More severely, if an actual cardiac failure is not detected because manipulated data show normal heartbeats, the health of the patient is threatened.

User Interaction The wearable ECG is designed to be used by elderly patients which might not have a technical background. After unboxing, the user needs to establish trust between the wearable device and the GOD. This is done by performing BLE out of band (OOB) pairing. A secret is exchanged between the wearable and the GOD via encoding the secret in the flashing of the red, green, blue (RGB) light-emitting diode (LED) at the wearable device and capturing and decoding the transmitted secret at the GOD. This secret is used to encrypt and authenticate transmissions between said devices. The user has to initiate the pairing process by pressing a button at the wearable and at the GOD and then arrange the devices such that the camera of the GOD can capture the RGB LED. Then, the user needs to enter their credentials into the GOD to be able to utilize ABE.

At run time the wearable ECG does not require user interaction except being recharged every 48 h. If technically fit, the user might use their terminal device (e.g., smart phone) to retrieve ECG data from the cloud and visualize those. However, if the user does not possess such a device, data are transmitted to the attending physician and inspected by them. The wearable device can also feed collected data into machine learning algorithms which attempt to automatically detect cardiac failure. For high risk patients this automated evaluation could be hooked to the rescue directing center. In case the wearable device reports cardiac failure or stops reporting data an ambulance could be automatically dispatched to the patient.

3.3 Case-Study 3: Monitoring Critical Infrastructure with the Intelligent Screw



① UHF-RFID antenna

② MIOTY antenna

③ Solar cell

④ Heat sink of thermo generator

Figure 3.4: *Intelligent Screw.* Powered by solar cells (left) or thermo generator (right).

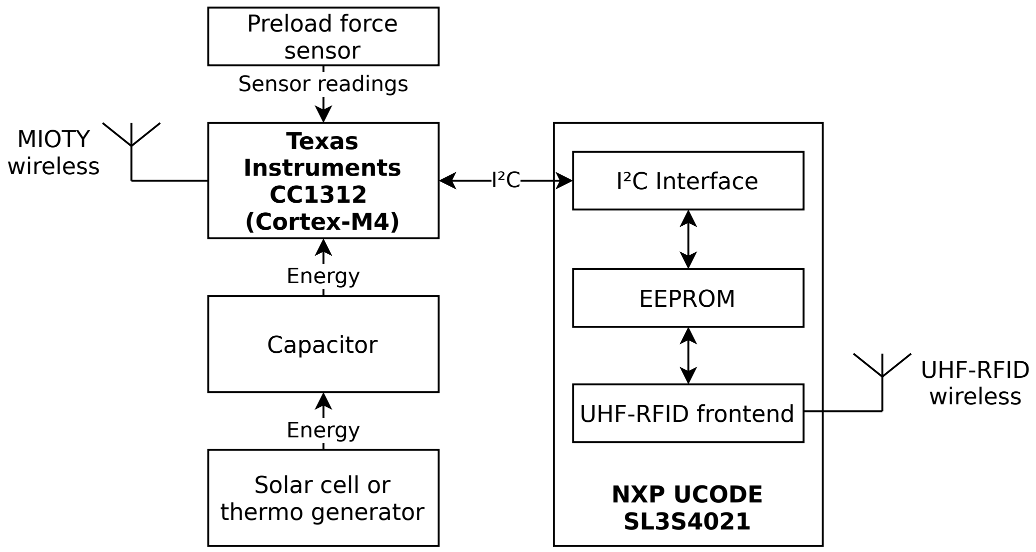


Figure 3.5: *Intelligent Screw* electronic components

System Description and Application The *Intelligent Screw (IS)*, which is jointly developed by several Fraunhofer institutes, is an IES housed in a common metric screw, as shown in Figures 3.4 and 3.5. The IS continuously meters its preload force to determine, whether it is still tightened sufficiently. It transmits those measurements to a backend over a wireless link. Upon detecting a loose screw, a technician is dispatched to re-tighten or replace it. Fields of application include critical infrastructure such as bridges or wind farms. To be able to work autonomously for a long period of time, this IES utilizes solar cells or thermo generators which harvest energy and charge a capacitor. The CC1312 MCU remains in deep sleep, until a sufficient amount of energy has been collected. Then, the MCU wakes up, reads the preload force sensor, transmits the sensor reading and goes back to sleep.

Communication Scenario The IS has two wireless interfaces. The first interface utilizes the *MIOTY* wireless protocol to transmit sensor readings using the built-in wireless interface of the CC1312 MCU [49]. It operates at 868 MHz and permits transmitting packets with a small payload over a range of multiple kilometers to a gateway. The gateway in turn forwards packets over a wired link to the MIOTY backend.

The second interface is ultra-high frequency radio-frequency identification (UHF-RFID) operating at 868 MHz. This interface is part of the *UCODE SL3S4021* integrated circuit (IC) by NXP Semiconductor and is connected to a electrically erasable programmable read-only memory (EEPROM) [50]. The EEPROM can be written or read via UHF-RFID and via a Inter-Integrated Circuit (I^2C) interface, which is connected to the CC1312 MCU.

Before initial use, the IS is provisioned with calibration data for the preload sensor and a symmetric key used to secure MIOTY transmissions. Those data are transferred to the EEPROM via UHF-RFID. As the UCODE IC is powered by the electromagnetic field of the UHF-RFID transmission, the CC1312 MCU can still be in deep sleep state during provisioning. From the EEPROM, data can be read into the CC1312 MCU via I^2C as soon as the CC1312 has collected sufficient energy to leave the deep sleep state. During network operation, the CC1312 MCU utilizes the symmetric key to encrypt data and create the keyed-Hash Message Authentication Code (HMAC), before sending them to the MIOTY backend. The backend is run by the operator of the IS and data are decrypted and checked for authenticity there. The gateway only sees encrypted data and can thus be operated by a third party.

Security Requirements Being used in critical infrastructure dictates that wireless messages sent by the IS are *authentic* and *available*: If screws are tightened sufficiently, but messages forged by an attacker state the screw came loose, a maintenance technician would be sent to investigate integrity of the infrastructure component. The same holds true in case of the absence of messages from the IS. The superfluous dispatch of the technician causes financial losses and loss of reputation at the manufacturer of the IS.

If, however, the screw came loose unnoticed due to a forged message stating its secure hold this can cause safety hazards.

Table 3.1: Comparison of the three networks and IES used therein

	Firefighter BAN	**ECG Wearable**	**Intelligent Screw**
Number of IES in sensor network	dozens to hundreds	one	hundreds
Wireless interfaces	6LoWPAN/IEEE 802.15.4	BLE	MIOTY (measurements), UHF-RFID (configuration data)
Data flows	N producers, 1 consumer	1 producer, N consumers	N producers, 1 consumer
Physical interfaces	LED, not accessible during use	RGB LED, pushbutton	none
Parties directly interacting with IES	firefighter, group leader, serviceman	patient	serviceman, operator of infrastructure
Setup activities per IES	Supply network configuration data and cryptographic material, place IES in PPE	Exchange cryptographic material (via BLE pairing), enter e-mail address, put on shirt	Supply calibration data and cryptographic material, attach screw to structure
Maintenance activities	Locate faulty IES among others, service or replace	Perform troubleshooting or call support	Locate faulty IES among others, tighten or replace

User Interaction The IS must be provisioned with calibration data and cryptographic keys before use. Similar to the BAN for firefighters, the IS can be provisioned by the manufacturer, in term making the cryptographic keys known to them. Then the keys must be transferred to the operator of the MIOTY backend securely. Additionally if a provisioned IS is stolen during transport the attacker is in possession of a valid secret key which then can be used to forge sensor readings, unless the particular IS can be identified and blacklisted in the MIOTY backend.

To overcome those issues, the IS could be provisioned right before deployment at the infrastructure component. As with the firefighters, the servicemen tasked to do this are no security experts. Because they are part of the security loop, errors made during setup can cause security issues during the operations phase of the network.

During network operation a particular IS might report insufficient preload force over MIOTY. A serviceman is dispatched to investigate if the structural integrity of the infrastructure component the screw is part of is endangered. The position of every individual IS at the infrastructure component, e.g., a bridge, might have been recorded during network setup. However, as all IS look the same and lack an optical identifier it is still hard to localize a particular IS in the field. As of writing this thesis, the localization issue has not been solved.

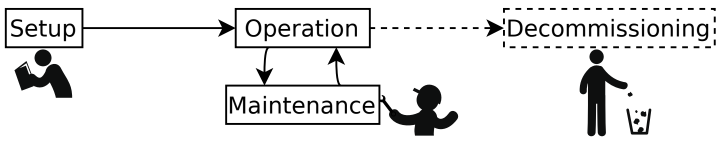


Figure 3.6: Network life cycle and direct user interaction.

3.4 Key Findings: User Interactions in the Network Lifecycle

Table 3.1 compares key aspects of the three networks of IES discussed above: The three networks **vary in their size**. A single firefighter carries six devices in total. This number times the number of firefighters in a particular fire station means that there are dozens to hundreds IES to be managed. For the IS, depending on the dimensions of the structure to be monitored, hundreds of IS might be in use. The wearable for recording an ECGs in contrast only consists of one IES. The type and size of the network obviously has large impact on the total effort needed to operate the network.

Analyzing those networks we find that direct user interaction dominantly occur in the *setup* and *maintenance* phases of the network (Figure 3.6). *Decommissioning* can occur either automatically, e.g., by providing IES with a fixed expiry date, or manually by simply unplugging devices. As the network will not be used after decommissioning, security errors made there are less critical than those made during setup or maintenance. Thus, decommissioning will not be taken into account for the remainder of this thesis.

During **setup**, users actively interact with the IES. Users need to distribute configuration data and establish a trusted relationship between communication partners, i.e., among IES or IES and a gateway or backend system. The latter lays the foundation for the security of a system throughout its life cycle. As the devices do not have a prior notion of each other, the *user* being in possession of the devices must be considered the trusted third party (TTP). By distributing cryptographic information such as keys to the IES, the user establishes trusted relationships among devices. While this is necessary, it contradicts property 5 introduced in Section 2.3.1 considering the user as the weakest link in the security chain.

As seen at the BAN for firefighters and the IS use cases, the user has to deal with medium to large-size networks. Evidently, provisioning devices in those networks by transferring a shared secret over a physical connection such as a wire, as suggested in early works, is unfeasible [51]. Not only is this process slow, it is error-prone due to the user having to perform security-related actions and thus being part of the security loop.

Device activity in the network can be monitored remotely. However, as already stated in Chapter 2, IES lack control interfaces for remote troubleshooting such as Secure Shell (SSH). Thus, **maintenance activities** in large networks of IES typically require localizing the faulty device among others which look the same before the actual troubleshooting at the physical device can be initiated. If network connectivity of the IES was interrupted and the cause of the error could not be deduced from wireless messages (or the lack thereof), finding the cause at the device is challenging due to the lack of physical I/O such as a display. For example, the wearable for recording ECGs has a RGB LED which can visualize error codes using blink patterns. The BAN for firefighters has a status LED. However, as the IES are integrated inside the PPE of the firefighters, the devices can not be accessed during a mission. The IS does not have any I/O capabilities apart from the wireless transmissions. Thus, without auxiliary means, all three networks lack means to provide feedback to the user and thus fail to permit intuitive user interaction (Property 3 from Section 2.3.1).

The networks vary concerning **who performs setup and maintenance tasks**, but all of those parties must be considered unmotivated users (Property 1 from 2.3.1) as their primary objective is making the network operational rather than managing security. The BANs for firefighters could be designed to not require any interaction from the firefighters themselves. Setup and maintenance of the network still need to be carried out, but this could be shifted to other parties (servicemen and group leader) which are available in this particular application scenario to aid the firefighter (the actual user). Hence the firefighter, whose primary mission is stressful and thus potentially harmful to the security of the system, is removed from the critical activity loops.

For the wearable device and the IS shifting those tasks to another party is not possible, as typically the users themselves (patients respectively serviceman) are the only parties having access to the networked devices. Thus, efficient troubleshooting in those networks remains an open challenge.

3.5 Summary

In this chapter I discussed three networks of IES I have worked on. I studied user interactions with those and analyzed the role of the user in the security loop. One key take away is that all three networks require user interaction in the setup and maintenance phases of the network. For some applications, those tasks can not be shifted to auxiliary users with a technical background and thus must be carried out by the actual nontechnical user. This user then faces a lack of I/O (such as a display) at the devices, which renders carrying out those tasks very challenging.

Based on those findings, in the subsequent sections I introduce novel approaches to aid users in setting up and maintaining a network of IES both securely and user friendly. To this, in Chapter 4 I propose an approach which removes the user from the security loop while setting up a network and thus satisfies design principle $\boxed{1}$. In Chapter 5 I demonstrate how monitoring a network using augmented reality (AR) permits intuitive interaction with the network, hence fulfilling design principle $\boxed{2}$. Finally, in Chapter 6 I explore means to educate users in wireless network security using the AR-based traffic monitoring (design principle $\boxed{3}$).

4 Secure and User-Friendly Setup of IES in a Faraday Cage

As stated in Chapter 3, users interact with networks of IES in all stages of the IES life cycle. Especially errors made during setup of the network can cause security issues later on. For example, if secret keys are not distributed properly, or non-unique keys are used, the security of the network is impaired right from the start of the network life cycle (*barn-door* Property 4). Obviously, the person setting up the network is in the security loop, which violates CRANOR'S design principle $\boxed{1}$.

In this chapter, I propose and evaluate an approach to perform *initial setup* of networks of IES using an intelligent Faraday Cage (FC) and over-the-air (OTA) Firmware Updates[1]. This approach permits even non-technical users to set up a network of IES in a user-friendly and secure manner.

In chapter 3 I have shown, that the initial setup phase of a network of IES usually comprises *provisioning*. Provisioning means, i.e., assigning the devices to a particular network and establishing trust between communicating parties, which is typically done by exchanging cryptographic material such as keys or certificates.

From the user's perspective, the classic way to interface IES is manually attaching a programmer and a cable. However, doing is while provisioning a large number of IES is tedious and does not scale well. Additionally, when IES are deployed in harsh environments or close to the human body, they are enclosed in a sealed casing to protect electronics inside from dirt and moisture. This limits access to the physical programming interfaces. Lastly, the person tasked with setting up the wireless sensor network (WSN) will likely not be a security expert, thus human error must be expected and dealt with.

Rather than using wired interfaces, in this chapter I propose a device and work-flow called *Box*, which permits exchanging sensitive information such as secret keys over the wireless channel inside a FC. The EM-shielding of the FC protects this process against eavesdropping, preventing an attacker from overhearing the secrets being transferred.

Except for closing and opening the Box and placing and removing IES, this procedure runs automated. This effectively removes the user from the security loop and thus fulfills CRANOR'S design principle $\boxed{1}$. I conducted a user study with 31 participants to

[1]

This chapter is based on the following publication:
[52]: *Martin Striegel, Johann Heyszl, Florian Jakobsmeier, Yacov Matveev, and Georg Sigl. Secure and user-friendly over-the-air firmware distribution in a portable faraday cage. In* Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '20, July 2020*

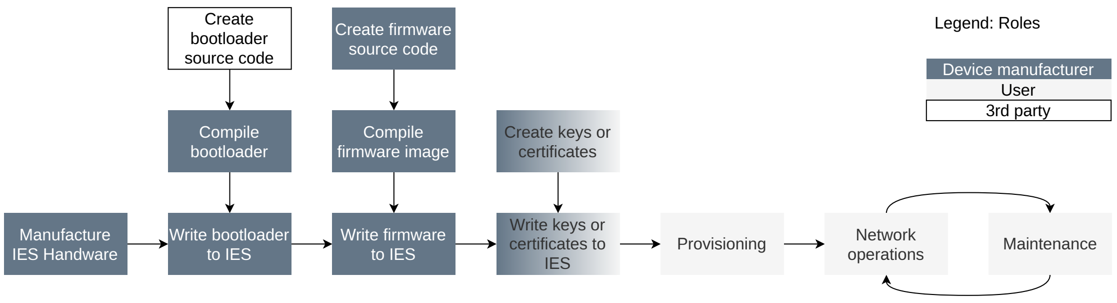


Figure 4.1: Overview: Tasks involved in manufacturing IES and creating a network

evaluate this approach and found that in addition to high usability the Box surpasses wired distribution in terms of speed and user satisfaction.

My original research focused on how the Box can be used by the end user, i.e., the party who uses the IES. While re-visiting the original publication for this dissertation I found that the Box also provides benefits for the manufacturer of the IES when the Box is being used during device manufacturing. Hence, in this Chapter I investigate both use cases *device manufacturing* and *provisioning*, as they are intertwined and both profit from using the Box.

I start by introducing tasks and roles involved in industrial manufacturing and provisioning IES and discuss challenges associated with those in 4.1. In Section 4.2 I describe key ideas how my novel Box approach helps to resolve those challenges and how the Box and its ecosystem are designed. Then, I outline the implementation in Section 4.3, followed by the user study in Section 4.4. I summarize this Chapter in Section 4.5.

4.1 Manufacturing and Provisioning IES

In this section I discuss how IES are manufactured and how they are provisioned in order to create a operational network. Based on those observations, I introduce my Box-approach and discuss, how it fits in this overall process.

I start this section by defining actors and tasks involved in manufacturing and setting up a network of IES (Section 4.1.1). Next, I discuss challenges in manufacturing and provisioning (Sections 4.1.2 and 4.1.3). Those challenges motivate my Box-approach, which is introduced in Section 4.2.

4.1.1 Scenario, Task and Role Definitions

For the remainder of this chapter I assume the following scenario: A medium-sized company wants to monitor their manufacturing plant by utilizing IES. We call this company the *end user* of the IES. The company buys commercial off-the-shelf (COTS) IES from a second company, which manufacturers the IES. We call the second company the *device manufacturer*. At both companies we find specific *roles* which will be introduced upon their first appearance in the sequence of tasks.

Figure 4.1 shows tasks to be done to create IES and establish an operational network with those and which of the above mentioned entities performs those tasks. In the following, I discuss those tasks in detail: The device manufacturer assembles the IES by mounting MCU chips, sensors, wireless connectivity, power supply, and additional components on a printed circuit board (PCB)[2]. Next, the device manufacturer compiles the bootloader and writes it to the IES. The bootloader is the first software which is launched after a MCU boots. It checks if the memory of the MCU contains an executable firmware image and if this firmware image is authentic, i.e., has been signed by the device manufacturer. If so, the bootloader launches this firmware. Else, the bootloader does not continue the boot process.

This authenticity check that only their own firmware is used on the IES and prevent the use of - potentially malicious - third party firmware. To enable the bootloader to verify the authenticity of the firmware image the device manufacturer adds their public key, which corresponds to the private key used for signing firmware images, to the bootloader. The device manufacturer then compiles the bootloader and the public key into the bootloader image and writes it to the non-volatile memory of every MCU via the debug interface of the MCU (e.g., the Joint Test Action Group Standard 1149.1-1990 (JTAG) interface).

Next, the device manufacturer compiles a firmware image from their source code. This firmware image includes run-time functionality of the IES such as reading sensors and wireless connectivity. The firmware image is signed using the private key of the device manufacturer. The device manufacturer then writes the firmware image to the IES, typically utilizing the debug interface or a Universal Asynchronous Receive Transmit (UART) connection to the bootloader.

Together with the firmware image the device manufacturer might also create keys or certificates and write those to the IES. The end user can utilize them later to integrate the device in their network without having to create their own secrets[3].

After having written the bootloader, the firmware, and the cryptographic secrets, the device manufacturer disables the debug interface of the IES. This prevents reading the memory of the IES and overwriting the firmware or the bootloader.

Next, the IES are shipped to the end user who wants to utilize the IES to monitor their manufacturing plant. If the device manufacturer supplied the IES with secret keys, those must be made known to the end user. For example, the secret keys might be printed on the enclosure of the IES. Else, the IES must offer an interface from which the keys can be read.

At the end user, typically a *serviceman* is tasked to set up and operate the IES. The serviceman now provisions the IES, i.e., introduces new devices to an existing network. The procedure depends on the type of wireless protocol used and procedures for protocols commonly used with IES are discussed in detail in Appendix A.1. In any case, this

[2]I deliberately omit the role of the *chip manufacturer*, as they do not play a role in the following considerations.

[3]This is typically done with IES which communicate using LoRaWAN. Please refer to Appendix A.1 for a detailed discussion on key management in common wireless protocols

involves the serviceman performing manual actions, e.g., manually exchanging secret keys and network addresses between the IES and the network.

After the network is operational, IES need to be maintained, which involves monitoring network operations and applying firmware updates to the devices, until the IES are decommissioned.

Attacker Model An attacker could pursue two goals which intersect: Firstly, the attacker might want to harm the device manufacturer directly by launching attacks on the manufacturing plant. As shown in Figure 1.1 in the introduction of this thesis, such attacks can lead to interrupted processes, financial losses, the theft of intellectual property and safety issues and injuries. Secondly, the attacker might want to harm the end user utilizing the IES which, however, also harms the device manufacturer indirectly. The end user might suffer from interrupted processes, financial losses, theft of intellectual property, safety issues and injuries. However, it must be expected that if the end user is victim to an attack the device manufacturer is blamed for having sold insecure devices to the end user and face legal issues and a loss of reputation. This shows that securing IES through their life cycle must be considered a joint responsibility between device manufacturer and end user: The device manufacturer must develop and manufacturers such that they are usable securely by the end user. The end user, in turn, is responsible for following the security instructions provided by the device manufacturer. At this interface we can assume friction and due to this it must be expected that the attacker targets this interface [13].

I assume the attacker to be present at the manufacturing plant during all stages of device manufacturing. The attacker could target the availability of the manufacturing plant, which is the most important security objective [53]. However, this would lead to IES simply not being manufactured and thus would not cause security issues at the end user of the device. As providing a holistic security concept for manufacturing plants is out of scope of this thesis, I focus on attacks where the attacker attempts to sabotage the production of IES with the intention that devices are manufactured insecurely, introducing security issues at the end user.

Ultimately, the attacker wants to harm the end user by *eavesdropping* on their network traffic with the intention to learn confidential information or *manipulate* or *forge messages* such as sensor readings so trick the unser into making wrong decisions based on those data. Assuming the *Dolev-Yao* attacker model initially introduced in Section 2.2 the attacker has full control over the wireless channel [32]. While a manufacturing plant typically employs physical access controls an attacker might bypass those to gain access to the facility and move there freely. Hence, the attacker must be considered an insider who operates in proximity to the wireless devices and can eavesdrop, replay, inject, jam and modify any message transmitted. To reduce the risk of being detected the attacker utilizes those wireless attacks instead of tampering with the hardware of contraptions used in the manufacturing, e.g., modifying adapters used to program IES. I assume that the attacker does not have physical access to the servers which compile the bootloader and firmware and sign those, as those servers are typically located outside the manufacturing facility.

In the next section I discuss challenges faced by the device manufacturer while producing IES and challenges in provisioning devices faced by the end user when setting up a network of IES. Those challenges motivate my novel approach which I am then going to introduce in Section 4.2.

4.1.2 Challenges in Manufacturing IES

Security Starting at 2024 European law demands that IES meet extensive security requirements [54, 55, 56]. The device manufacturer is responsible for securing devices in the early stages of their life cycle and later when firmware updates are used to remedy security problems. Besides secure design and implementation of the IES, they need to be manufactured in a secure facility so no security issues are introduced during production. For example, the device manufacturer must be able to create and store cryptographic secrets in the manufacturing plant and to distribute those to the devices. Especially the private key used to sign firmware images must be stored securely, while the corresponding public key must be transferred to the IES so they can verify the authenticity of firmware images.

Securing manufacturing environments is impeded by legacy equipment which has not been designed with security in mind and the desire for availability, which means that shutting down a manufacturing line for updating or replacing equipment is unfeasible. With manufacturing plants becoming increasingly connected, they can not longer be considered "air-gapped", i.e., isolated from the outside world. Treating the inside of manufacturing plants as secure environment protected by perimeter defenses does not hold true any more and security standards demand a multilayered approach to security instead ("defense in depth" [53]).

Besides those challenges, staff at the production plant has typically little to no background in security. We find the following roles: *Industrial engineers* plan the manufacturing of IES. They develop and build electromechanical contraptions to aid in manufacturing those devices, such as programming adapters used to interface multiple devices. They also might develop the software used to run those contraptions. *Production staff* utilize those contraptions to manufacture the IES. Typically, they have received no or very limited technical training.

Manufacturing For utilizing the wired interfaces to write the bootloader and fimware images, test points are placed on the PCB, which expose, e.g., JTAG or UART. To contact those test points, *programming adapters* must be developed and built by the industrial engineers. Those programming adapters consist of a base plate and a lid, to which pushrods and needle adapters are attached to. Production staff place a number of PCB are placed in fixed positions on the base plate. Closing the lid presses pushrods on the PCB to fix it in place. The needle adapters are also pushed down and touch the test points on the PCB, establishing electrical contact. Bootloader and/or firmware are written to the PCB and afterwards those are removed from the programming adapter.

Both test points and programming adapters have several drawbacks. Placing the test points on the PCB requires space, which contradicts the desire for building ever smaller devices. Programming adapters must be engineered and customized for every PCB, which is expensive. This also enforces that the design of the PCB is fixed at an early stage of the device development so the industrial engineers can develop and manufacture the programming adapter on time. The maximum number of PCB which can be programmed simultaneously directly depends on the size of the programming adapter. Thus, programming a large number of devices simultaneously requires large and costly programming adapters. Small series production and customization of products is hence unfeasible due to the high cost associated with building the required adapters and adjusting the software used to control those. Traditionally, this software did not need to handle customized firmware images, but IES demand individual cryptographic keys per device and thus their firmware has to be customized. This also means, that the production environment must ensure that new keys are generated for every device to be programmed. It must further be recorded which key was assigned to which device in order to provide this information to the end user of the device so they can utilize those keys in provisioning their devices.

The device manufacturer wants to write the firmware image to the IES *as late as possible* during manufacturing, because this maximizes the time available for firmware development and permits customization of the devices at a late time in the manufacturing process. However, depending on their field of application, IES might utilize sealed enclosures so they can be used in harsh environments. After the PCB is mounted in the sealed enclosure, the test points are not accessible to the programming adapter anymore. This dictates, that if the device manufacturer wants to ship the IES with the most recent firmware, mounting them in the sealed enclosure must be the very last step during production. This dictates a specific sequence for manufacturing, impeding flexibility.

Summary Device manufacturers face challenges by new security requirements imposed on both their production environments and on the produced IES themselves. Management of cryptographic keys and the limitations of programming adapters make the distribution of customized firmware to IES challenging.

In the next section I introduce additional problems faced by the *end user* of the IES during provisioning those devices. Then, in Section 4.2 I introduce my novel approach which helps both the device manufacturer and the end user to overcome those challenges.

4.1.3 Challenges in Provisioning IES

As described in Section 4.1.1, the end user (in our example a medium-sized company) bought a number of IES which shall be used to monitor their manufacturing plant. IES are mounted on machines and meter, e.g., vibrations. Those sensor readings are transmitted from the IES to a networked application, which collects, stores and analyzes received data.

Security The user wants their sensor readings to be *authentic* and *illegible* to other parties. To this, IES utilize cryptographic algorithms for authenticating and encrypting messages. The security guarantees of those algorithms depend on the confidentiality of the secret keys used and thus access to those must be limited. For performance reasons, symmetric encryption is preferred over asymmetric encryption techniques. Because IES can be deployed such that the attacker has physical access to the devices, keys must be stored securely at the IES. As IES typically lack a secure key storage such as a, keys are stored in their non-volatile memory. To limit access to this memory, communication interfaces, especially the JTAG interface of the IES, need to be locked. To reduce the damage in case a key is leaked, an *individual* secret key per IES is preferred.

Provisioning Inherent to provisioning is the *bootstrap problem*: To be able to exchange sensitive information a secure communications channel is needed. This secure communications channel can be established using cryptography. In order to utilize cryptography, keys must have been exchanged between communicating parties. To secure this exchange, a secure communications channel is needed, which does not exist yet.

In case of provisioning IES, secret keys, device identifiers, and network information must be exchanged between the IES and the Backend, which shall receive the metered data over a secure channel[4]. The device manufacturer might have written secret keys to the IES to take this burden from the user, but this brings the following problems: The end user must entrust the device manufacturer that the cryptographic secrets were created correctly (i.e., using a true random number generator (TRNG)) and handled securely at the manufacturing plant. Keys need to be transferred to the end user over a secure channel, so the user can integrate the IES in their network infrastructure. This means, that keys are stored in the firmware of the IES and that this firmware must be protected, as otherwise the attacker might read it from the devices and learn the keys.

Next, the end user needs to learn those keys and make them known to his Backend to be able to secure communications cryptographically. If keys were not provided to the end user, e.g., as part of the documentation of the device, they need to retrieve these keys from the IES. How to do this? The wired interfaces at the IES might not be usable any more, as due to security reasons the JTAG interface might have been locked to protect the firmware. If the IES has a tightly sealed enclosure designed to withstand environmental impact, the wired interfaces might not be accessible any more. Lastly, even if a wired interface were accessible, without dedicated equipment reading information from a large number of devices would be a tedious undertaking for the user, as the user would most likely need to plug a cable into every IES.

I conclude that users lack user-friendly interfaces to communicate with the IES. Lastly, even if the user managed to interface the IES, they would need a secure channel between the IES and their Backend to exchange those sensitive information. Such a channel is only present after secret keys have been exchanged, but for exchanging the secret keys, this secure channel does not exist yet.

[4]This is handled differently for different communication protocols and the reader is referred to Appendix A.1 for details.

Location-limited channel (LLC) for Provisioning To provide a way to interface IES and create an initial secure channel between the IES and the Backend, researchers proposed *LLCs* [51, 57]. By their nature, LLCs enforce physical proximity and/or directivity and thus they provide authenticity and, in many implementations, confidentiality as well. They also have the property that an attacker sending on those channels will be detected by the legitimate user. As mentioned by [51], wired connections are an instance of a LLC, but due to their limited usability, previous literature proposed other LLCs which can be used to exchange sensitive data. Those include channels using light [58, 59, 60, 61, 62, 57, 63] or sound [64, 65, 66] to transmit information.

Users themselves can also be considered LLCs. For example, users can manually enter a secret in two devices which shall establish a secure connection [9]. Another idea comprises the user shaking devices similarly and simultaneously. By doing so, the user injects shared randomness in devices, from which shared secret keys can be derived [67, 68, 69, 70].

Another class of approaches utilizes unauthenticated asymmetric key-exchange protocols. After the key exchange, devices then display an artifact of the key exchange to the user. This artifact can be a visual or audible digest of the exchanged secret. The user establishes authenticity by manually comparing the data displayed and confirming equality [71, 72, 73, 74, 75, 9].

Implementing those types of LLCs has drawbacks: Firstly, IES must be equipped with additional hardware, increasing their cost and size. Secondly, letting the user perform security-critical tasks, e.g., comparing audio patterns and confirming their equality, is dangerous, as minor operating errors can turn into critical security errors [15].

[69] briefly mentioned the idea to use a FC for provisioning devices, which was further explored by [76, 77]. The main idea is that due to its EM shielding properties the FC creates a LLC inside in which secret keys can be exchanged without an external attacker being able to eavesdrop or disturb the key exchange. Compared to other LLCs, inside the FC the IES can utilize their primary wireless communications protocol to exchange secrets securely. Thus, no additional hardware needs to be added to the IES. Also, by using wireless communications a large number of IES can be interfaced simultaneously, speeding up the provisioning process. Additionally, using a FC permits automating security-critical tasks and removes the user from the security loop. With my Box-approach I build on those works by providing an optimized design for the FC and demonstrating its use in additional applications.

Summary While provisioning their devices users face security and usability problems, as they need to exchange secret keys between their IES and their Backend without having established a secure communications channel yet. I presented LLCs designed to simplify provisioning and then discussed, how using a FC-based LLC is a promising approach for provisioning large numbers of IES in a efficient and user-friendly way.

To stick to the order of tasks presented in Figure 4.1, in the upcoming Section 4.2 I initially show how my FC-based approach can be utilized by the device manufactuer to

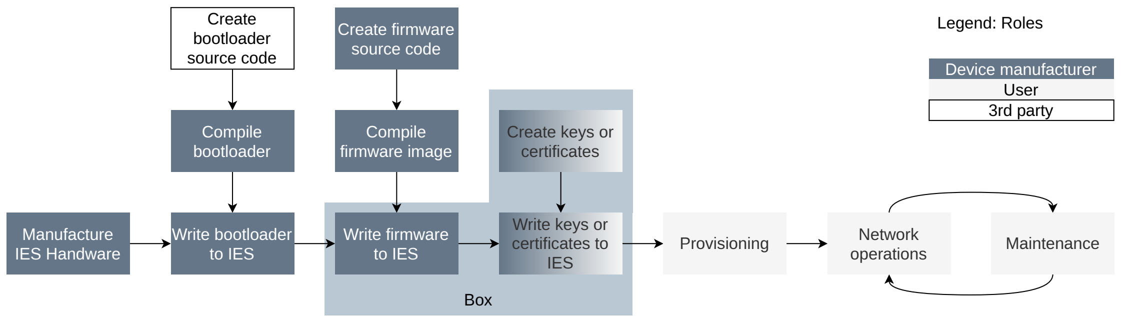


Figure 4.2: Using the Box in Manufacturing of IES: Tasks covered by the Box

produce IES. Then, I discuss how the end user can also use my approach to provision IES.

4.2 Box-Based Solution: Usage and Design

In this section, I introduce my Box approach. I start by presenting how the Box can be used by device manufacturers to optimize their manufacturing process (Section 4.2.1). Next, I describe how the Box aids the end user in provisioning their IES (Section 4.2.2). Then I detail the design of the Box (Section 4.2.3) and its ecosystem, particularly the role of the Backend and the IES (Sections 4.2.4 and 4.2.5).

4.2.1 Using the Box in Manufacturing of IES

Figure 4.2 shows how the Box is utilized in manufacturing IES. The Box is utilized after IES have been supplied with a bootloader. I require that MCUs are supplied with an OTA-capable bootloader. This type of bootloader is able to receive firmware images over a wireless communication link.

My system has three entities: One Box, one Backend, and an arbitrary number of IES. The Box is a Faraday Cage (FC) which shields internal communications from the outer world. This essentially creates a LLC, which prevents an attacker outside the Box from overhearing or interfering with communications which occur inside. The Box is part of the IES manufacturing line and operated by production staff. The Backend of the device manufacturer is, e.g., a server which controls the manufacturing process.

Figure 4.3 shows the workflow using the Box, divided into steps ① to ④. Initially, the Backend creates a firmware image and a number of unique secret keys using a true randomness source. The Box is connected to the Backend using e.g., a cable (step ①). The Box requests a firmware image and secret keys from the Backend and stores them into its memory. This can be done automatically, e.g., at the beginning of a production shift.

To deploy firmware images to IES, the production staff powers them on, places them inside the Box and closes its lid (step ②). Upon detecting that it is shut tightly, the Box creates a wireless network within. Now each IES utilize its wireless transceiver to

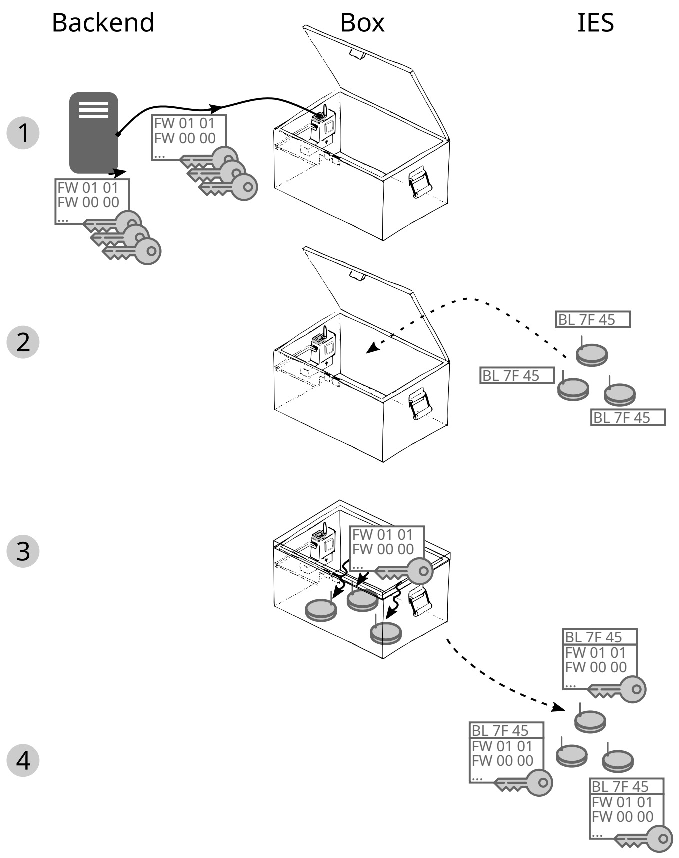


Figure 4.3: Using the Box in Manufacturing of IES: Workflow with the Box

join the network and request a firmware image OTA from the Box. Upon receiving such requests, the Box patches an individual secret key into every firmware image by writing the secret key into the firmware binary file. Then, the Box cryptographically signs the now unique firmware image and sends it to every IES (step ③) This process is shielded from the exterior of the Box as it blocks EM waves and operates at low power.

Afterwards production staff removes the IES from the Box. Every IES now contains the runtime firmware image which includes an individual secret key and functionality to be configured by the end user, to connect to the network of the end user, and to read sensors (step ④).

Discussion: Using the Box in Manufacturing of IES The key idea of the Box-approach is to utilize an OTA-capable bootloader for simultaneously deploying firmware images to many IES during device manufacturing. Can we assume an OTA-capable bootloader to be present at the IES so the Box can be utilized?

The de-facto standard bootloader for IES is the open-source `MCU boot`, which supports OTA firmware updates [78]. OTA firmware updates are supported by many commonly used transmission protocols and MCU platforms used in IES [79, 80, 81, 82, 83, 16]. In addition, having OTA update capabilities will be a mandatory requirement for IES as of 2024, as IES must be able to receive firmware updates for remedying security issues [54, 55, 56]. Thus, I conclude that the assumption of having a OTA-capable bootloader is reasonable and that the Box can be applying in device manufacturing. With this in mind, I am now going to discuss the benefits of utilizing the Box.

Secure and user-friendly workflow: To be able to cryptographically sign firmware updates, the Box can either be part of the Public Key Infrastructure (PKI) of the production plant or have a permanent network connection to the Backend. In the former case, the Backend delegates signing firmware updates to the Box. In the second case, the Box sends customized firmware images over the network to the Backend and receives the signed firmware images. The Box encapsulates security-critical tasks such as transferring secret keys to IES within its shielded enclosure and produces a user-friendly workflow to production staff, who merely have to place IES in the Box, close the lid of the Box, re-open the lid, and remove the IES.

The EM shielding of the Box provides an additional layer of security, contributing to the defense-in-depth approach demanded by industrial security standards [53] and hindering an attacker, who gained access to the manufacturing plant from overhearing or interfering with wireless transmissions,

Reducing EM interference: As a side-effect besides protecting the internal communications against eavesdropping, the EM shielding provided by the Box also reduces interference from external communications or EM noise, which is especially present in industrial environments. By doing so, communications inside the Box experience less packet loss due to interference, speeding up the firmware update process. In addition, the shielding ensures that wireless communications outside of the Box are not disturbed by communications inside, which contributes to the overall reliability of the wireless environment at the manufacturing plant.

Late customization and flexibility: All MCU chips can be supplied with the same bootloader using the same programming adapter. This can be done in an early stage of device manufacturing while the MCU chip does not even have to be assembled to the PCB yet. The bootloader must only be configured insofar that the wireless protocol, which is to be used to write the runtime firmware image, is set. At a later point in time when the MCU chip is mounted to the PCB and thus the actual IES has been manufactured, using the OTA bootloader firmware images can be written to the IES. This maximizes the time available for firmware development and permits the final customization of devices at a very late stage of production. For example, the same base hardware-platform could be used in multiple products by being customized with firmware images, which fulfill different tasks (e.g., read different sensors mounted at the same IES hardware platform).

Efficiency: Programming adapters only allow a small number of IES to be supplied with firmware simultaneously. In contrast, using the Box the number is only limited by the physical dimensions of the Box in relation to the the dimensions of the IES, and the maximum number of devices supported by the network protocol. Being able to supply the runtime firmware image to the IES over the wireless link using OTA updates can be considered the final acceptance test, as the successful transfer of the firmware

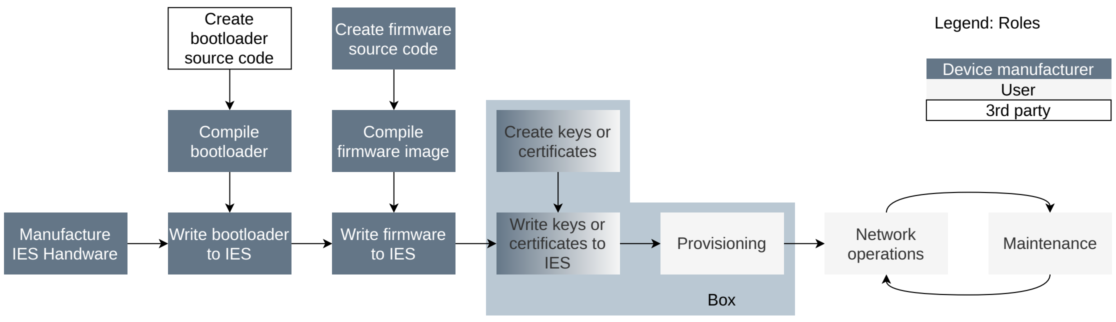


Figure 4.4: Using the Box in provisioning of IES: Tasks covered by the Box

image proves, that the wireless system of the IES is operational. Thus, no additional acceptance tests must be run, which saves time and costs.

Summary In this section I have shown how the Box can be used by the device manufacturer while producing IES and the benefits its provides both in terms of efficiency and security. In the next section, I am going to show how the Box also benefits the end user of the IES while provisioning their devices and establishing a network with those.

4.2.2 Using the Box in Provisioning of IES

Figure 4.4 shows how the Box can be used by the end user of the IES to perform provisioning of their devices. The following entities are involved: An arbitrary number of IES, one Box and one Backend. At the end user, the Backend might be a a server to which the IES transmit sensor readings to be processed and stored there. Figure 4.5 shows the operational sequence when using the Box for provisioning, divided into steps ① to ⑤.

The user interfaces the Box with the Backend over a secure channel, e.g., a cable, which can be considered a proximity-based LLC (step ①). The Box requests information on the network to which the IES shall be connected to from the Backend and stores them into its memory[5]. Now, the Box can be unplugged and is fully portable.

Next, the user powers the IES on, places them inside the Box and closes its lid (step ②). Upon detecting, that it is shut tightly, the Box creates a wireless network inside with a default address and no access control. Each IES can now utilize its wireless transceiver to join the network and request network information from the Box. The Box transfers network information to the IES and in turn requests the secret key from every IES which it stores mapped to the address of the IES (step ③). IES and the Box can exchange those sensitive information in clear text as the EM shielding of the Box secures this exchange[6]. Afterwards, the end user removes the IES from the Box and

[5]For example with 802.11 Wi-Fi (Wi-Fi) those information include the Service Set Identifier (SSID) and passphrase of the network of the end user the and Internet Protocol (IP) address of the Backend.
[6]Note, that after the initial setup of the IES the features to read out secret keys should be disabled or be cryptographically protected. However, this is subject to the firmware implementation.

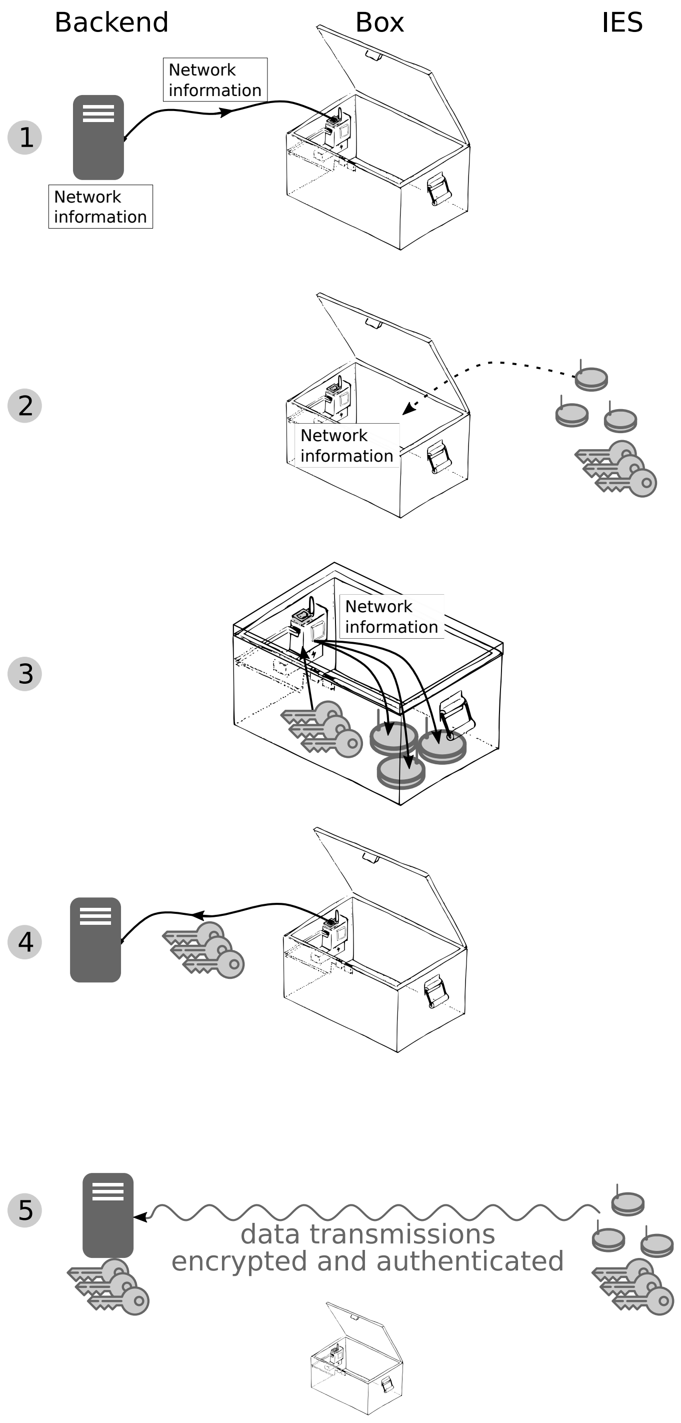


Figure 4.5: Using the Box in provisioning of IES: Workflow with the Box

deploys them. The end user connects the Box to the Backend and the Box uploads the secret keys from the IES to the Backend (step ④). Now, as the IES know the network information, they can join the network and communicate with the Backend. They can also utilize end to end encryption and authentication of messages, since every IES and the Backend share an individual secret key (step ⑤).

Discussion: Using the Box for Provisioning **Secure and user-friendly workflow:** My Box approach addresses the bootstrap problem faced by end users while provisioning their network of IES. The main idea is that the Box creates a LLC which provides a secure environment in which sensitive information such as network credentials and secret keys can be exchanged between IES and the Backend of the end user.

Many IES can be provisioned simultaneously by simply placing them in the Box, closing the lid of the Box, and removing the devices from the Box. The security-critical tasks of correctly exchanging sensitive information is automated by the Box. This improves speed and minimizes room for human error since the user only has to perform a few simple manual actions. Especially when provisioning a large number of IES this avoids repetitious sequences, such as, e.g., scanning Quick Response (QR) codes or Near-Field Communications (NFC) tags at every single device to be provisioned, which would quickly cause the user's attention to deteriorate, this leading to security errors [46].

Easy interfacing: Unlike other solutions discussed in Section 4.1.3, the Box interfaces the IES over their primary wireless interface. This means that no access to wired interfaces such as UART is required to write, e.g., network information or to read secret keys. Thus, unlike other LLCs proposed, using the Box does not require additional hardware or communication interfaces at the IES.

To simplify interfacing the Box with the Backend, the connection between those should be implemented by, e.g., mobile communications secured with end to end encryption and authentication. This means that the Box and the Backend share a permanent connection, automating the steps ① and ④ in Figure 4.5. This reduces the manual steps to be conducted while provisioning IES to opening and closing the lid of the Box and placing and removing the devices.

Creating own secret keys: The end user might not trust the secret keys provided by the device manufacturer and choose to create their own keys. In this case, the user creates keys at their Backend, which are then transferred to the Box. Together with the network information the Box writes those keys to the IES (step ③ in 4.5). Step ④, where the Box is connected to the Backend afterwards, is not needed, as the Backend already knows the secret keys. Hence, after having been removed from the Box, IES can directly establish secure communications with the Backend.

Using the Box for updating the firmware of IES: The Box has been designed for the initial provisioning of devices and using it during updating firmware does only provide little benefits. Firmware updates are often provided at the website of the device manufacturer, they are not considered confidential. However, we require firmware updates to be authentic and free of malicious functionality. The authenticity is typically ensured by the device manufacturer cryptographically signing the update files. The bootloader of the IES contains the public key of the device manufacturer and can thus verify the authenticity of the update file. As after having been provisioned using the Box, each IES and the Backend can communicate over a secure channel, we already have a layered security concept as demanded by industrial security standards [53]. Additionally, as the IES has a connection to the Backend, the end user can initiate the update process from the Backend. Thus I conclude that while the Box excels at simplifying the provision task, the end user does not need it to perform firmware updates of their devices.

Summary In this section I have discussed using the Box for device provisioning in terms of user-friendlyness and security. While the end-user benefits from using the Box, I find that the Box is even more useful to the device manufacturer. As to the best of my knowledge this use case of a FC-based LLC has not been explored before in literature, the architecture, implementation and user study presented in the following sections are centered around using the Box in device manufacturing.

4.2.3 Design of the Box

Having introduced the use of the Box, in this section, I detail the technical design of the Box, its subcomponents and its ecosystem. As stated before, the following descriptions focus on the use of the Box in device manufacturing. Hence, for the remainder of this chapter I assume the *user of the Box* to be a production staff at the device manufacturer with little to no technical background.

The Box comprises a Faraday Cage, a single board computer (SBC), wireless transceivers, battery, an open/close sensor and the user interface (UI). The UI consists of a speaker, two pushbuttons with built-in status LEDs (the *acquire* and the *deploy* button) and the power button, all placed inside the Box and being accessible only while the Box is open.

The Box is modeled as a five-state state machine, which is shown in Figure 4.6. States are based on the Box being opened or closed and whether its internal memory contains firmware images and keys. Transition between those states is initiated by the user performing manual actions, such as opening or closing the Box and pressing a button. Security-relevant tasks such as downloading firmware and keys from the Backend into the Box and the actual firmware distribution process are conducted automatically by the Box.

The speaker provides spoken information to the user about the current state and which action the user shall perform next. Using a speaker has several benefits over using e.g., LEDs: Firstly, mechanical integrity and EM shielding are not impaired, as no wires need to be led through the enclosure of the Box. Secondly, spoken audio feedback is more explicit than encoding information into LED on/off state and coloring.

To protect the exchange of firmware and secret keys from eavesdropping in addition to the EM shielding the Box utilizes both software and hardware attenuation mechanisms to transmit wireless messages at low transmission power. The transmission power is set such that IES inside the Box are just able to receive messages. I can utilize such

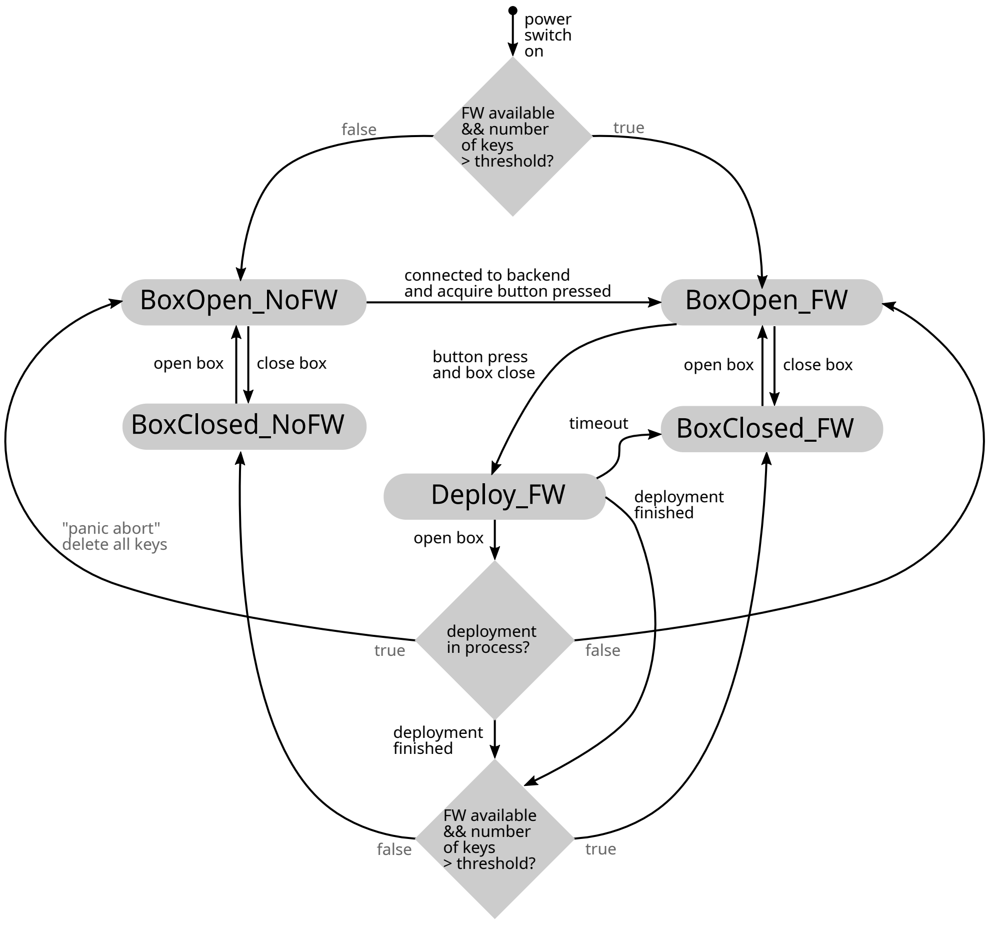


Figure 4.6: State machine of the Box

low transmission power and still maintain a good signal to noise ratio (SNR) inside the Box because the EM shielding protects communications inside the Box from external interference, e.g., found in industrial environments.

4.2.3.1 State Machine

As the pushbuttons and sensors, which cause state transitions, have been placed inside the Box, the mechanical design ensures, that no unintended states are reached

After power-up, the Box checks, if at least one firmware image and a number of keys greater than a preset threshold reside in its memory. If the actual number is below the threshold, it moves to the `BoxOpen_NoFW` state. The user connects the Box to the Backend and presses the *acquire* button. The Box automatically acquires a firmware image and secret keys and stores those[7]. Then, it moves to the `BoxOpen_FW` state.

[7]Note that this can also be done automatically, e.g., at the beginning of a production shift.

If the Box is closed, while the deploy pushbutton has not been pressed, the Box transitions into the `BoxClosed_FW` state. The `Deploy_FW` state is entered from the `BoxOpen_FW` state after the user has pressed the *deploy* pushbutton and closed the Box. Now the Box creates a wireless network using the transmission protocol and connection parameters expected by the IES. For example, if Wi-Fi is used, the name and passphrase of the access point as well as the address, at which the webserver serving the firmware images can be reached, match the defaults stored in the OTA bootloader of the IES.

If the user does not place IES inside the Box, the Box waits for requests for a certain time. If no requests were received, the Box informs the user about this. Then, it moves to the `BoxClosed_FW` state, where it resides until it is opened by the user.

Upon receiving an OTA firmware request from the IES, an individual secret key is patched into the firmware image. Binary patching is computationally less expensive then compiling each firmware with its individual key and thus much faster, speeding up the firmware distribution process. The Box sends those customized firmware images to the requesting IES. It counts individual device addresses, e.g., MAC addresses from which it received requests and stores the mapping of secret key and device address, which are sent to the Backend later. After deployment is finished, the Box informs the user about the number of IES provisioned and instructs the user to open the Box and remove the IES. Upon opening, if the remaining number of keys inside the database of the Box is higher than a threshold, the Box moves to the `BoxOpen_FW` state, else to the `BoxOpen_NoFW` state.

How does the Box know, if all IES have been supplied with firmware? My user study described in Section 4.4 shows, that users are aware of how many IES they placed inside the Box. Thus, users notice a mismatch between the number of provisioned IES the Box announced and the number of IES they placed inside the Box. Also, typically IES have an LED. With this, they can issue a blink pattern after having received the new firmware, indicating success.

While the Box is closed, wireless transmissions inside are secured against eavesdropping by the EM shielding in conjunction with the low transmission power. The most dangerous state would be `Deploy_FW` while the Box is opened. If the open/close-sensor of the Box detects, that it is opened while deployment is in process, it performs a panic abort, since an opened FC does not shield EM waves sufficiently. As the Box expects an attack, it erases all secret keys from its database and informs the user. Then, it moves to the `BoxOpen_NoFW` state.

The hardware-enforced state machine of the Box prevents reaching critical states while the Box by design reduces security-critical tasks to simple manual actions. Those manual actions hide the complexity of the OTA firmware and key distribution, while the mechanical feedback of operating the Box still allows the user to determine cause and effect of his actions. Hence, possibilities for handling errors are minimized by keeping the user out of the security loop while providing them a sense of control [15].

4.2.3.2 Shielding for Eavesdropping Security

I want to protect the exchange of firmware with an embedded secret key against a passive remote attacker. I assume the attacker knows how the wireless packets are structured, i.e. how the firmware is split into the payload blocks and where to find the key in a particular firmware chunk.

Obtaining information by eavesdropping is impossible if the received power P_{RX} is lower than the sensitivity of the attacker's receiver. Additionally, the received signal strength must be below the noise floor, i.e. the SNR is too low to be able to distinguish information from noise.

Using the radio link budget equation and taking into account the attenuation provided by the Box, the signal strength at the attacker can be calculated as [84]:

$$P_{RX} = P_{TX} + G_{TX} - L_{FSPL} + G_{RX} - L_{Box} \tag{4.1}$$

where P_{RX} is the power received by the attacker, P_{TX} the transmission power of the sender, G_{TX} gains at the sender, G_{RX} gains at the receiver, L_{FSPL} the free space path loss and L_{Box} the attenuation of the Box. I control P_{TX}, G_{TX} and L_{Box}, while the attacker can influence L_{FSPL} and G_{RX}.

The free space path loss is given by [84]:

$$L_{FSPL} = 20 \cdot log_{10}(d_{cm}) + 20 \cdot log_{10}(f_{MHz}) - 67.55 \tag{4.2}$$

where d_{cm} is the distance between sender and receiver in centimeters and f_{MHz} the frequency in megahertz. In Equation 4.1, I do not consider fading or losses caused by antenna polarization mismatch. Those would *increase* attenuation and contribute to the protection level. As I calculate a worst-case scenario here, I omit them.

First, I determine the minimal power $P_{RX,min}$, at which the IES used are still able to receive wireless messages. For the IES used in my example, this $P_{RX,min}$ is about $-90\,dBm$ [85]. Hence, I feed signals from the wireless transceiver of the Box through a chain of four $20\,dB$ hardware attenuators before they are being passed to the antenna. I reduce $P_{TX,Box}$ in software to cancel the antenna gain and set $P_{TX,Box}$ so low that IES within the Box are just able to receive the signal, in my example slightly above $-90\,dBm$. The EM shielding provided by the Box reduces interference from outside the Box, permitting reliable communications at this low transmission power. While it is desirable that only the Box sends sensitive information, IES already utilize very low transmission power to preserve energy, making it difficult for the attacker to overhear their messages in case they send sensitive information.

Example Calculations Transmitting at 2.4 GHz, assuming L_{Box} at $40\,dB$, $L_{FSPL} = 34\,dB$ in a distance of 50 cm and G_{RX} at $30\,dB$, we calculate $P_{RX,Attacker} = -134\,dBm$. This lies well below the sensitivity of typical receivers an attacker would use, which is around -96 dBm [86].

We can assume that the user of the Box would notice an attack setup this physically close to him. The attacker could use more powerful (and thus larger) antennas and wireless receivers to achieve a higher signal strength, however, his presence can be detected more easily then. To account for this, my hardware-store Box prototype, which attenuates by 40 dB, can be exchanged for commercial solutions offering shielding greater than $70\,dB$.

The SNR at the attacker's receiver is derived as follows: For exchanging firmware, in my tests, I use 802.11n Wi-Fi at $20\,MHz$ channel width and the maximal data rate of 150 Mbps supported by the IES used in my experiments. Assuming only thermal noise (a worst-case assumption security-wise) at bandwidth $B = 20\,MHz$ and room temperature, the noise floor of the attacker's receiver is $-101\,dBm$. The SNR is thus $-134\,dBm - (-101\,dBm) = -33\,dB$. I transfer each individual key only once, so the attacker can not reduce noise by combining multiple transmissions.

We can utilize the *Hartley-Shannon Law* to evaluate, if the attacker's receiver is able to receive data at a sufficient rate to not lose information under a given bandwidth and SNR [84]. The SNR needed to achieve *channel capacity* is calculated as follows:

$$\frac{S}{N} = 2^{C/B} - 1 \tag{4.3}$$

where C is the channel capacity in bit/s and B is the band width of the transmitted signal in 1/s. To achieve channel capacity for our Wi-Fi data rate of $C = 150\,Mbps$ at $B = 20\,MHz$ bandwidth, we require a SNR of $17.4\,dB$. Note that this is the best-case assumption for the attacker: It only holds true using optimal coding, which is not given in practice.

Those calculations show, that even close to the Box and assuming fairly low attenuation by the Box, the received power P_{RX} is lower than the minimal receive power required by a typical attacker. Further, the SNR at the attacker is so low, that channel capacity can not be achieved, hence received information will be erroneous. Thus, the attacker will not be able to learn the firmware image and the secret keys by eavesdropping. With this low SNR provided by the design of my Box, unlike previous solutions using a FC [76, 77], I do not require a jammer mounted externally to the Box.

To be able to maintain those security levels, the Box must be shut tight during firmware distribution, which is checked by the open/close sensor in the `Deploy_FW` state. If the Box is opened during key exchange, all keys are deleted from the memory of the Box and the user is warned.

4.2.3.3 Security Against Active Wireless Attacks

To hijack IES, an attacker might launch a rogue wireless network with the same identifier the Box would use to create its network. By doing so, IES are tricked into joining the attacker's network and the attacker could try to hijack IES by writing a malicious firmware to the devices. This attack is thwarted by the bootloader of the IES verifying the authenticity of firmware updates and rejecting unsigned firmware images.

The attacker could also launch jamming attacks to disturb the OTA firmware update procedure. The EM shielding of the Box counteracts this by reducing the signal strength of the jamming signal to lower levels. As an additional means of protection, the wireless spectrum at the manufacturing plant should be monitored continuously to detect jamming attacks.

How to detect a malicious IES inside the Box? The attacker could prepare a custom device with a malicious firmware, which eavesdrops wireless traffic. They would then try to smuggle the malicious device in the Box to eavesdrop the exchanged secret keys, bypassing the security provided by the EM shielding. Afterwards, the attacker could remove the malicious device from the Box later or exfiltrate the confidential data captured via a wireless channel. This data can then be analyzed to extract all keys distributed in this run of the Box.

For this attack, the attacker needs access to the manufacturing plant, which is assumed in our attacker model. In addition to that the attacker must prepare their eavesdropping device to make it appear like the benign IES to be supplied with firmware in this particular production run. Else, its visual appearance being different from the normal IES would be noted by the production staff.

To further prevent this kind of attack from inside the Box, *Secure Erasure* techniques proposed by Perito *et al.* can be applied [87]. The idea is that the Box requests a IES to erase its memory and the requester receives a proof if the device did obey or ignore this request. Secure erasure is used after the OTA bootloader was written to the devices over the wired interface and before the runtime firmware image is deployed. IES are forced to delete all memory not occupied by the bootloader, thus removing potentially malicious code. The malicious device placed inside the Box can act by either answering to the erasure request or ignore it. A device is not able to fake the verification of the erasure, therefore it is detectable if the memory was not cleared.

If all devices respond, the Box can verify that the whole memory is securely erased, and proceed with firmware and key exchange. If the malicious device does not communicate at all and only eavesdrops on wireless traffic passively, the Box can not notice this. However, as discussed earlier it is a reasonable assumption that the production staff knows how many devices were put in the Box. After the firmware update the Box announces via the speaker how many devices were securely erased and how many received new firmware. The productions staff intuitively notices the difference between the number of devices placed into the Box and the number of devices actually supplied with firmware. Thus, they can detect that not all devices were flashed. They can choose to write firmware to the batch again, e.g, individually or pairwise to reduce effort and find which devices were not flashed, or call for an investigation by a qualified colleague.

4.2.4 Backend

The Backend might be a server in the manufacturing plant and has the largest computational resources in the system. It can utilize a true randomness source and creates secret individual keys. Additionally, might compile or at least have access to firmware images for different types of IES which are manufactured. The Backend waits for the Box to be connected, this connection being secured by cryptographic protocols. Upon receiving a valid download request from the Box, the Backend transfers firmware images and a list with secret keys to the Box. After the Box has supplied secret keys to IES, the Box transfers the list including which secret key was assigned to which IES to the

Backend. This list can then be given to the end user who bought the IES allowing them to provision their devices.

4.2.5 Wirelessly Interconnected Embedded Systems

Initially, the device manufacturer writes an an OTA-capable bootloader to the IES using a programming adapter interfacing the debug interface. Then, OTA updates are used to program the IES, debug interfaces can be locked. This aids in protecting the bootloader from being manipulated and secret keys from being extracted.

The bootloader contains the public key corresponding to the private key used by the device manufacturer to sign their firmware images. After power-up, the bootloader tries to connect to a default address and upon being connected successfully requests a runtime firmware-image. The Box listens on this default address and upon receiving a request, it customizes a firmware image by injecting a secret key, signs it, and delivers this image to the requesting IES. The IES validates this firmware image using the public key and upon success writes it into its `Bootloadable` memory region and reboots, loading this image. The runtime firmware image contains an individual secret key which can be used by the end user for encrypting and authenticating messages. It also contains functionality to connect to the network of the end user, to read sensors, and to transmit sensor readings over the wireless link.

4.3 Implementation

4.3.1 Box

Figure 4.7 shows the first prototype of the Box. It is made up of a COTS lockable aluminum case spaced 320 mm · 230 mm · 150 mm. I covered the interior of the case with copper-sheets and -tape and metered the attenuation to be at least 40 dB in the 2.4 GHz frequency band. In prototype generations v2 and v3 (shown in Figures 4.8 and 4.9) I utilized a commercial FC which provides attenuation of 60 dB [88].

Inside the Box, a *Raspberry Pi Model 3* (Pi) ①, a battery pack ②, a power button ③, the *deploy* ④ and the *acquire* pushbutton ⑤ with built-in green and blue LEDs and a speaker ⑥ are attached. Using a tailored enclosure, only the buttons, the speaker and the Ethernet interface ⑦, which connects the Pi to the Backend, are exposed to the user. This prevents faulty operation by limiting the possible interactions. The user is instructed about the current state and how to proceed by spoken output over the speaker. Whenever the user is audibly instructed to press a particular button, the built-in LED of this button is flashed.

I monitor the state of the Box (e.g., whether its opened or closed) with a Hall Sensor ⑧ connected to a GPIO pin and hooked to an interrupt handler. The magnet ⑨ which activates the Hall sensor is attached to the lid. All invalid transitions (e.g., open lid during firmware distribution) result in a secure fail state that deletes all secret data in the memory of the Box, as we must assume, that the key exchange has been compromised due to the absence of EM shielding.

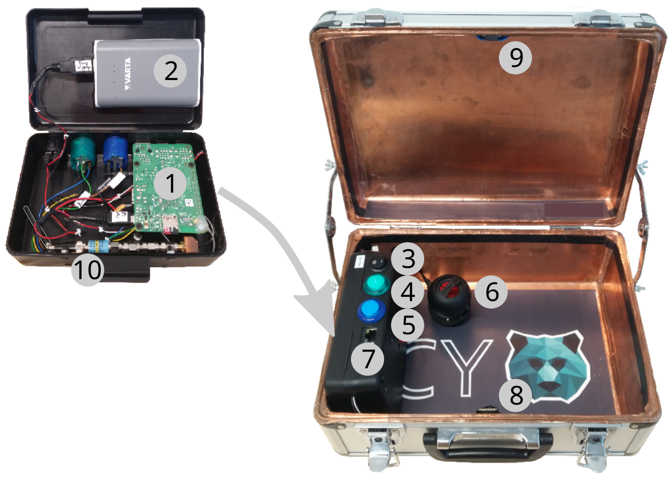


Figure 4.7: The first *Box* prototype.

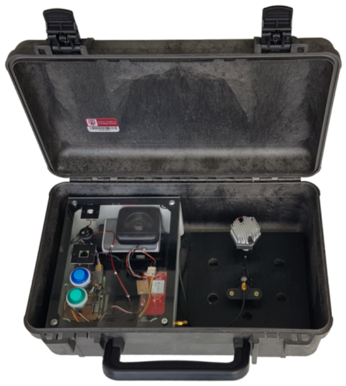


Figure 4.8: *Box* prototype v2 utilizing a COTS Faraday Cage.

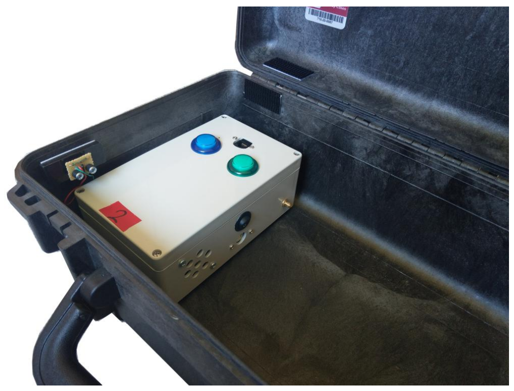


Figure 4.9: *Box* prototype v3 with miniaturized electronics.

The Pi acquires firmware images and keys from the Backend using Ethernet. The Pi can use its built-in Wi-Fi and Bluetooth transceivers to communicate with IES, however, to support other transmission protocols, additional transceivers can be attached via Universal Serial Bus (USB). In my tests I used the built-in Wi-Fi interface of the Pi. When deploying firmware (thus the Box being in closed state) the Raspberry Pi creates a Wi-Fi access point (AP). Firmware requests sent to the AP are handled by a web server which in return sends the firmware patched with individual keys to the IES. Multiple requests can be handled simultaneously by the web server, thus, several IES can be served in parallel.

The more powerful Backend builds the firmware images and transfers them to the Box rather than compiling them in the Box. The Box patches the firmware images by automatically opening the firmware binary image and replacing the key placeholder with an individual key. Subsequently, the Box signs firmware images. Hence, by using binary patching a firmware image can be customized much faster than writing keys into the source code and compiling each individual firmware image.

Security against eavesdropping is achieved by combining the EM shielding of the Box with hardware signal attenuation. For that, I replaced the Pi's built-in Wi-Fi antenna with an SubMiniature version A (SMA) connector. By chaining analog SMA-attenuators before the antenna ⑩, the transmission is reduced to the minimum power level at which the IES are still able to receive packets (about -90 dBm for our IES). This renders the external jammer needed by previous works obsolete, as the SNR at the attacker is already very low.

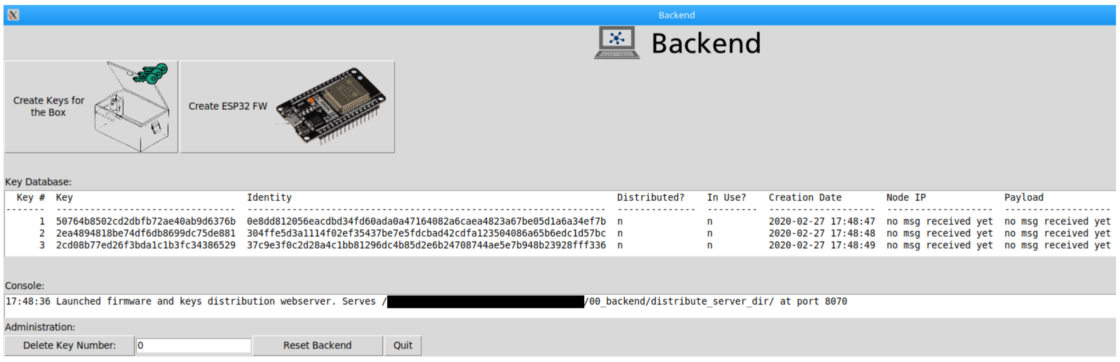
After the original research [52] I improved the prototype by utilizing a COTS FC which provides about of 60 dB of signal attenuation [88]. This prototype is shown in Figure 4.8. In prototype generation v3 I minimized the physical dimensions of the electronics so additional IES fit in the Box and thus can be provisioned simultaneously (Figure 4.9).

With those prototypes, I was able to further demonstrate the modular architecture and the applicability of the Box approach to practical IES deployment scenarios. To this, I added support for UHF-RFID transmissions so the Box could be used to provision the Intelligent Screw (IS) as described in Chapter 3.3.

4.3.2 Backend

The Backend, which would be the production server at the device manufacturer, is written in `Python3` and runs on an industrial computer (IPC). The GUI built with the `tkinter` package and has buttons for creating and deleting firmware images and secret keys. The process of creating keys can also be performed automatically, i.e., to ensure that a given number of keys is always available.

Communication between Backend and Box uses HTTP over an Ethernet cable, however HTTPS could be used if the link between Backend and Box is untrusted. To initiate the IES setup process, the user plugs the Box into the IPC. The Backend runs a Dynamic Host Configuration Protocol (DHCP) server, which supplies the Box with an IP



```
Key #  Key                               Identity                                                          Distributed?   In Use?   Creation Date         Node IP               Payload
-------  --------------------------------  ----------------------------------------------------------------  -------------  --------  --------------------  --------------------  --------------------
      1  50764b8502cd2dbfb72ae40ab9d6376b  0e8dd812056eacdbd34fd60ada0a47164082a6caea4823a67be05d1a6a34ef7b  n              n         2020-02-27 17:48:47   no msg received yet   no msg received yet
      2  2ea4894818be74df6db8699dc75de881  304ffe5d3a1114f02ef35437be7e5fdcbad42cdfa123504086a65b6edc1d57bc  n              n         2020-02-27 17:48:48   no msg received yet   no msg received yet
      3  2cd08b77ed26f3bda1c1b3fc34386529  37c9e3f0c2d28a4c1bb81296dc4b85d2e6b24708744ae5e7b948b23928fff336  n              n         2020-02-27 17:48:49   no msg received yet   no msg received yet
```

```
17:48:36 Launched firmware and keys distribution webserver. Serves /[redacted]/00_backend/distribute_server_dir/ at port 8070
```

Figure 4.10: The graphical user interface (GUI) of the Backend

address. It also runs a web server, from which the Box requests a firmware image and secret keys up to a pre-configured amount.

4.3.3 Wirelessly Interconnected Embedded Systems

I implemented the IES using *PYCOM LoPy4* development boards (based on the ESP32 MCU) and the *Pysense* sensor shield for ease of development. These devices support four wireless interfaces: WiFi, Bluetooth (BT), LoRAWAN and SIGFOX. Further, they are capable of performing OTA firmware updates over Wi-Fi. I used OTA over Wi-Fi with the default SSID, Wi-Fi passphrase and web server address set by the manufacturer of the LoPy4 devices. The Box uses those defaults to launch an access point and a web server, at which it serves firmware updates.
After having connected, the LoPy4 send a `GET` request to the web server of the Box, which prepares the firmware image with an individual key as described above and transmits the so customized firmware image to the requesting LoPy4.

4.4 Usability Study

Purpose and Scope: For this usability study, participants assumed the role of production staff at the device manufacturer and were asked to perform the manufacturing task of writing firmware to the IES. The aim of this usability study was to compare the speed and user-friendliness of supplying firmware and keys to IES using the Box and a programming adapter. In the following, those two approaches are called *Wired* and *Box*. The task for the participants was to supply firmware to IES using both approaches. I did not build a dedicated programming adapter for the study. Rather, for Wired, a single IES was interfaced via UART using a cable. After wiring the IES the following steps are automated using as script, which takes care of creating and injecting a unique key into the IES firmware image and writing the image to the IES.

I investigated the following hypotheses:

1. Using the Box to provision IES needs less time than using wired provisioning. The advantage in speed increases, the more devices are to be provisioned.

2. Using the Box is intuitive and does not permit the user to make security-critical errors.

Experimental Design and Setup: The two approaches *Box* and *Wired* are compared within-participants, i.e. all participants performed both approaches. I used LoPy4 IES and a laptop running `Ubuntu 18.04` representing the Backend. To interface the laptop with the LoPy4, I used `esptool.py`, a command-line tool, for which I created a clickable desktop icon for easier use.

Participants were given the task to program IES with firmware and device secrets. They were further instructed, that upon issuing a particular code word, the experiment supervisor would show up and assist them. The code word prevented accidental or inexplicit calls for help.

Sample: I recruited a total of 31 participants. Of those, 21 were male, 10 female. The age of the participants ranged between 23 and 53 years. Twenty participants were from computer science and electrical engineering study programs. For those I assumed a certain familiarity with networking, embedded systems, and security. Hereinafter, they are called *experts*. To test whether the Box approach is suitable for users with no technical background I recruited additional 11 participants. Five of those had an academic degree which I considered unrelated to the task, while six did not have an academic degree at all. Hereinafter they are called *novices*.

Procedure: Prior to execution, the supervisor explained the test scenario, described the tasks to be performed and explained his own role as technical support. Next, participants were given a printed instruction sheet in their native language. They were asked to read aloud the instruction sheet and ask questions on the instructions. There were no questions raised in the final study, as I had evaluated the instruction sheet in a pilot study with six participants and adjusted it to answer all questions raised there.

Participants were provided with a switched-on laptop, its desktop clearly showing the executable icon for initiating wired programming, and four LoPy4 IES. They were asked to supply firmware to those four devices using *Wired*. For that, they were instructed to connect the IES to the computer used for programming, hold a pushbutton to enter the bootloader mode, execute the script and finally unplug the IES.

To evaluate the Box approach, participants were given the Box prototype and four LoPY4 IES, all switched on prepared with an OTA over WiFi-capable bootloader. The Box already held a firmware image and a sufficient number of secret keys in its memory. This setup resembles a scenario in which the shift manager has already supplied keys to the Box at the beginning of the shift. Production staff manufacturing the IES would not be permitted to interact with the Backend. Participants were instructed to place all four IES together in the Box, press the deploy-button, and close the Box. Then they were supposed to wait while the Box programmed the IES, until the Box instructed them to open it and remove the IES.

Table 4.1: Study results. Times given for wired deployment are per IES, for the Box for one run with four IES.

		Wired				**Box**
Novices	Time needed [s]	#1	#2	#3	#4	total
		42.3	35.3	26.3	24.2	29.7
	Hesitations:	2 total				1
	Errors:	3 total				0
	Support calls:	0 total				0
Experts	Time needed [s]	#1	#2	#3	#4	total
		42.9	29.9	27.2	22.6	30.9
	Hesitations:	5 total				2
	Errors:	4 total				1
	Support calls:	0 total				2

Further, this accounts for opinions from one approach influencing the opinion on the other approach. After participants had completed the firmware distribution process I asked them for qualitative feedback on their experience with the Box using the widely-used System Usability Scale (SUS) [89].

Measurement: I measured the *time needed* to provision IES from the moment the subject consciously stated "start" until all devices were provisioned successfully. I counted the *number of hesitations*, *errors* and *calls for support* made. Hereby, *hesitation* denotes a time span of at least three seconds, in which the subject did not proceed. Hesitations are not necessarily related to an *error*, which denotes doing something different from the instruction sheet. Those metrics map to the usability attributes *learnability, attractiveness, user-friendlyness, low cost to operate* and *security interaction* according to ISO 9241-110:2006 [40].

4.4.1 Results

Due to the small sample size data were only analyzed descriptively. Results are summarized in Table 4.1.

Time Needed: Using *Wired* both experts and the novices took around 42 s to program the first IES. For the fourth IES it took them an average of 23 s. One run of the Box with four IES inside took the experts an average of 30.9 s and the novices 29.7 s.

Hesitations and errors: With *Wired* the novices hesitated twice and made three errors. All of them could be attributed to the subject struggling with the button to enter the bootloader mode. With the Box, one hesitation and zero errors were counted. The subject thought only a single IES was to be placed in the Box. After re-reading the instruction sheet, the subject placed the remaining IES in the Box and proceeded.

The experts hesitated five times and made four errors using wired distribution, while there were two hesitations and one error using the Box. All errors and hesitations during wired distribution were related to the bootloader mode button, e.g., the button

Table 4.2: SUS results. Results scale: 1 = strongly disagree, 5 = strongly agree

Question	Experts	Novices
I think that I would like to use this system frequently	4.9	4.8
I found the system unnecessarily complex	1.2	1.0
I thought the system was easy to use	5.0	5.0
I think that I would need the support of a technical person to be able to use this system	1.3	1.6
I found the various functions in this system were well integrated	4.8	4.8
I thought there was too much inconsistency in this system	1.1	1.2
I would imagine that most people would learn to use this system very quickly	4.9	5.0
I found the system very cumbersome to use	1.1	1.0
I felt very confident using the system	4.3	4.9
I needed to learn a lot of things before I could get going with this system	1.2	1.1

not being pressed, searching the button at the laptop screen rather at the IES or pressing and releasing the button immediately. The hesitations caused by the Box are related to the participants not having understood the audio feedback and hence they were not sure whether the programming process had finished. The error was caused by the subject turning off the open Box, despite the power switch being in the clearly marked "on" position and the green deploy button flashing.

Number of support calls: The novices requested the support zero times for both approaches, while the experts asked two times using the Box. Both occurrences were caused by the Box prototype not responding to the deploy button press.

SUS: Results from the SUS questionnaire are shown in Table 4.2. Participants gave overall good grades. Differences between experts and novices are negligible.

Qualitative feedback: Participants described the Box as very practical and intuitive to operate. They liked the gain in speed over the wired approach and that the Box replaces cable handling with more convenient manual actions of placing devices and operating the Box lid. The voice instructions were perceived helpful and could be understood easily. Participants felt reminded of automated external defibrillators, which also give step-by-step instructions and with which they are more familiar. As the Box informed the user that the firmware exchange was finished and the Box could be opened, participants were able to judge the success of deployment reliably.

4.4.2 Discussion

As expected the Box outperforms wired distribution in terms of speed and scalability. Two aspects contribute to the total time needed for one run: Firstly, time needed for devices to communicate, download the firmware and write the new firmware image to flash memory. Secondly, the time needed for device handling. Concerning the former, the gain in speed stems from OTA programming inside the Box taking place in parallel, while the wired approach is limited to programming one IES at a time. For device handling time, independently from previous experience, both groups became quicker while handling the IES in the wired approach. Assuming the fastest time needed for wired programming of one IES, which is about 23 s, I extrapolate that provisioning two IES takes about 46 s. Using the Box to set up *four* IES took both groups about 30 s. Thus I can conclude that the Box is faster than wired provisioning, as soon as more than one IES is to be programmed and if no dedicated programming adapter supporting more devices is available. Using the Box, device handling times varied, because some participants arranged IES neatly inside the Box, while others just took the bunch of four and dropped them randomly. While IES can be successfully provisioned at any position inside the Box, as stated in Section 4.2.3, the user should be aware, *how many* devices have been placed inside. Hence, it seems useful to structure the interior of the Box such that one IES resides at one position, e.g., by adding foam-cutouts.

The SUS and the qualitative feedback show that participants consider the Box intuitive, easy to use, and well integrated. Especially participants who had received industrial training liked, that operating the Box requires only simple manual actions, which they felt confident doing. They also liked that the Box hides the electrical and software aspects, which are exposed more in wired programming. Participants stated, that this reduced their worries of breaking something.

One error was made, as the subject turned the already switched Box off. Support was called twice, as the Box prototype did not respond to the button press. Two participants from the experts group wished to be able to understand better, what is happening internally. They asked for more fine-grained information, e.g., with which IES the Box is communicating at the moment. To account for that the Box should provide more precise feedback more often by playing repeated audio feedback without the user performing an action. Alternatively, a help button could be added which, upon being pressed, tells about the current state and how to proceed. Overall, all participants felt more confident and expressed higher satisfaction using the Box compared to wired programming.

4.5 Conclusion

In this chapter I introduced the *Box*, an intelligent Faraday Cage (FC), which can supply firmware images and secret keys to large numbers of low-power wirelessly interconnected embedded systems (IES) in a user-friendly manner using over-the-air (OTA) updates. I demonstrated two scenarios and workflows for utilizing the Box: Industrial manufacturing of IES and provisioning those at the end user. In the first case, I showcased how the Box can be used to efficiently deploy firmware and individual secret keys to IES leveraging OTA firmware distirbution. In the second case, I showed how the Box helps the end user to provision their IES, i.e., exchange the device-individual secret keys and network information between the devices and their network Backend.

In both scenarios my proposed workflow shifts security relevant tasks from the user to the Box, minimizing room for human error. Manual actions are reduced to opening and closing the box and placing and removing devices, which can be performed even by non-technical users. I confirmed the gain in speed and the high usability of the Box approach in a study which shows that both technically adept and non-technical users are able to use the Box intuitively and without security errors. Hence, I conclude that the Box approach satisfies CRANOR'S design principle $\boxed{1}$ by removing the user from the security loop.

5 Maintenance: Augmented Reality-based Network Monitoring System

After having shown how to perform user-friendly manufacturing and provisioning of networks of IES, in this chapter I present a novel approach for user-friendly *maintenance* of such networks[1]. In Chapter 3 I stated that the maintenance phase of the network requires user interaction. I found this interaction to be challenging, as IES lack input/output (I/O) at the device and thus are not able to provide information useful for troubleshooting to the user: After a centralized network monitoring system has indicated a failure in the network, a serviceman is dispatched to investigate the problem. The IES can not provide meaningful information to aid him in the maintenance process, such as helping to localize the malfunctioning devices among others which look alike or provide debugging information. For example, the Intelligent Screw (IS) introduced in Chapter 3 does not even have a status light-emitting diode (LED). This shows that users can not interact with IES intuitively and thus CRANOR'S design principle $\boxed{2}$ is violated.

My key idea detailed in this chapter is to equip the serviceman (which I consider the *user* for the remainder of this chapter) with a hand-held device which passively captures wireless network traffic and a camera image of the networked IES. The hand-held device superimposes the camera image with a visualization of the captured network traffic, creating an augmented reality (AR) representation of the networked devices and their otherwise imperceptible wireless communications. The field of view of the camera acts as a user-friendly filter to the information displayed, as my monitoring system only shows information on the devices captured by the camera. With this, users can inspect IES on-site, gather information from its wireless interface and thus obtain the information needed to perform troubleshooting. This permits intuitive interaction with networks of IES, satisfying design criterion $\boxed{2}$.

I consider a network of IES which has been configured and deployed and hence is now in the *operation phase* of the network life cycle: IES act as IES and collect sensor data and transmit those data to a server for processing and storage.

Similar to enterprise-class IT devices, such a network is usually monitored by capturing network traffic at the gateway. Then, at a central control terminal traffic logs, topology graphs and network statistics are presented to the network operator [91, 92].

[1] This chapter is based on the following publication [90]: *Martin Striegel, Carsten Rolfes, Johann Heyszl, Fabian Helfert, Maximilian Hornung, and Georg Sigl. EyeSec: A Retrofittable Augmented Reality Tool for Troubleshooting Wireless Sensor Networks in the Field. In* Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks (EWSN '19)*, 2019*

If there is a problem in the network and the *maintenance* phase is entered, the approaches for IES and more powerful networked devices differ: As discussed in Chapter 2, computationally more powerful devices such as Internet of Things (IoT) devices and enterprise-class IT devices (such as network hardware, e.g., switches) can be managed *actively* using e.g. Secure Shell (SSH) or Simple Network Management Protocol (SNMP). In contrast, IES lack those interfaces and can thus not be serviced remotely. Further, they typically lack I/O capabilities to display their status in a comprehensible way at the devices themselves. Hence, IES behavior can only be observed *passively* by inspecting network traffic and maintenance must be conducted on-site.

If IES failure is reported by the central monitoring system, a serviceman is dispatched to pinpoint and fix the problem on-site. To be able to provide the serviceman with information on the network at the location of deployment, [93, 94] described hand-held devices, which obtain information on the network from the gateway and display them. This is disadvantageous, as the hand-held devices require a permanent connection to the gateway and their operativeness depends on the availability of the gateway. As low-power wireless protocols such as *Bluetooth Mesh* or IPv6 over Low power Wireless Personal Area Network (6LoWPAN) use mesh networking, in which packets can be forwarded from source to sink node over multiple hops, information at the gateway might be incomplete. If packets from a certain source IES do not reach the gateway anymore, it might conclude that the source node is broken, although, the problem could be caused by the failure of an intermediate IES, which does not forward packets from source to sink anymore. Thus, to become independent of the gateway as network traffic source, separate and passive *capture (or "sniffer") devices* have been proposed [95, 96, 97]. Those mobile sniffer devices are deployed temporarily in the vicinity of the network to be observed and capture all network traffic.

Combining mobile sniffer devices to obtain network traffic and hand-held devices to display information permits the serviceman to work independently of existing infrastructure. However, by just shifting the visualization of network data from a central terminal to a hand-held device does not tackle specific problems encountered by the serviceman. While a complete view on the network such as traffic logs, topology graphs and network statistics can be displayed at a hand-held device, in practice the serviceman needs only limited information targeted at solving a specific problem. The serviceman needs to localize the physical device causing problems in the network among a large number of devices which all look the same. Hence, they need to map the *digital* device representation, i.e. network addresses obtained from captured network traffic, and the *visual* device representation of the physical IES in front of them. Vice versa, while manipulating a physical device, the serviceman needs to see the effects on the digital world, e.g., if a network connection was restored.

I found that there is no tool designed to provide the serviceman in the field with just the information needed to debug networks of IES and thus make this process user-friendly. To overcome this, I propose an augmented-reality based network monitoring system (ARMS), in which an *AR Device* equipped with a camera detects and identifies IES using Quick Response (QR) code markers. Portable *Sniffer Units* capture network traffic and extract digital information. Data from the visual and digital domain are merged

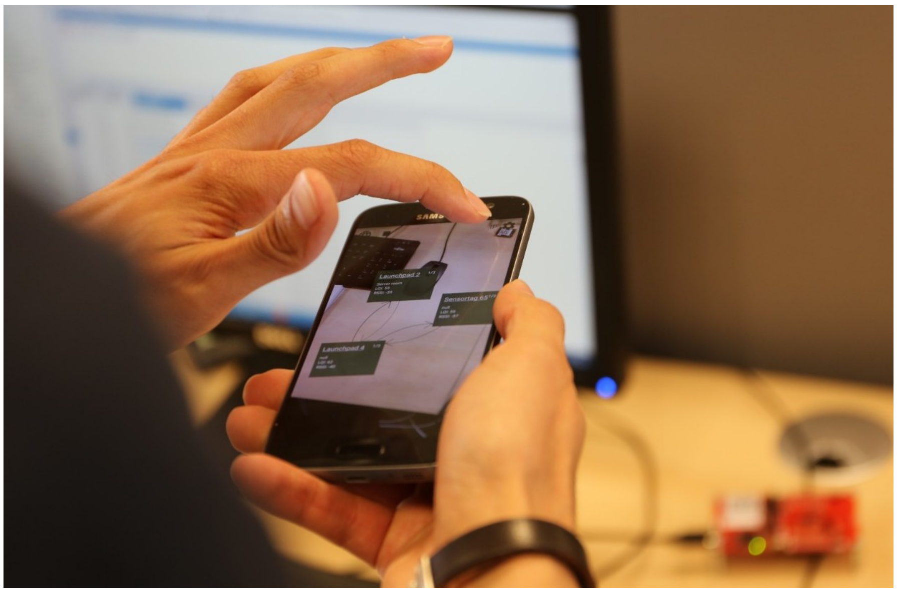


Figure 5.1: Hand-held AR device superimposes the camera image of the physical IES with device information and network traffic flows represented as arrows between devices

and stored at a *Backend* application. An AR device obtains consolidated information from the Backend and superimposes the camera image of the physical IES with this information, as shown in Figure 5.1. By doing so, data flows and connectivity between IES are visualized. Unlike centralized network visualization solutions, which display the full network, I exploit the serviceman's physical proximity to the IES and the field of view of the camera to limit displayed information to those of interest. ARMS is designed such that Sniffer Units and Backend can be installed ad-hoc at a network deployment site and removed after troubleshooting is finished, neither requiring changes to, e.g., the firmware of the networked devices being observed nor interfering with network operations at any time.

The contributions of the original publication were as follows:

- I presented the first completely mobile network monitoring system, which permits servicemen to work independently of any existing infrastructure.
- The system operates completely passively, i.e. no modifications to the firmware of observed devices is needed. This permits the monitoring system to be retrofitted to already deployed networks without introducing additional sources of error.
- The monitoring system utilizes off-the-shelf radio transceivers which are available widely and cheaply. The system has a modular and protocol-agnostic design so extra transmission protocols can be added easily.

I start by detailing requirements and design considerations for building a retrofittable and protocol-agnostic AR system in Section 5.1. Then, in Section 5.2, I show how this

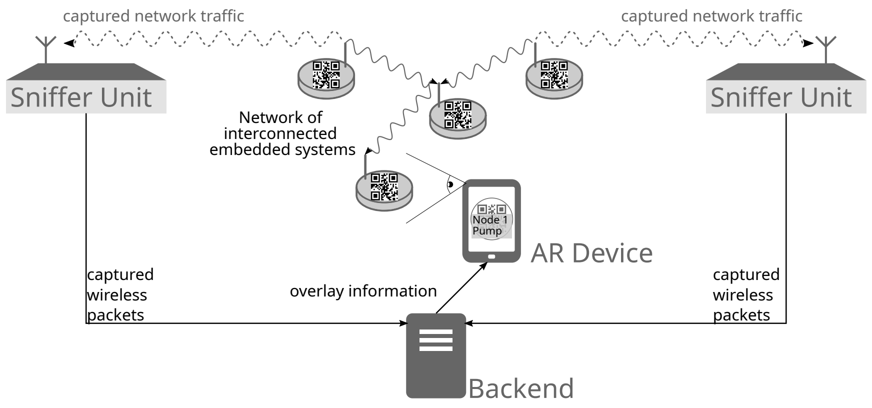


Figure 5.2: Using the augmented-reality based network monitoring system (ARMS).

design has been implemented. In Section 5.3, I evaluate the ARMS in a real network using 6LoWPAN/IEEE 802.15.4 Low-Rate Wireless Networks (IEEE 802.15.4) transmission protocol. In Section 5.4, I compare my system to previous approaches. Section 5.5 concludes this chapter.

5.1 Design

In this section, I present the design of the ARMS. I start by giving an overview of the hardware units used and how they communicate in Section 5.1.1. Then in Section 5.1.2, I outline how the ARMS is used and how usability is incorporated into the design. I describe data processing pipelines in detail in Sections 5.1.3 to 5.1.6. Finally, in Section 5.1.7 I discuss the security design.

5.1.1 System Design and Communications

The ARMS consists of three hardware units: *Sniffer Unit*, *Backend* and *AR Device*. They are designed modularly, such that hardware units can be merged or split depending on the size of the network to be monitored. Any combination of Sniffer Unit, Backend and AR Device can be utilized. Figure 5.2 shows the interaction of those units while monitoring a network of IES. The network is displayed in the center. Every IES has an optical marker, e.g., a QR code. Two dedicated Sniffer Units, shown to the top left and bottom right, have been deployed among IES and passively capture traffic from the sensor network. They extract data from traffic and transmit those data to a Backend, where they are stored. The AR Device is a hand-held device carried by the serviceman. It reads the optical marker of a IES, from which derives the identity of the IES. In addition, it has an integrated Sniffer Unit. The AR Device fetches data from the Backend and superimposes a IES with those data, e.g., the name and location of the node.

Hardware units of the ARMS communicate using Wi-Fi. The Backend creates a protected Wi-Fi network, which Sniffer Units and AR Devices join. I chose Wi-Fi to connect ARMS devices, as Wi-Fi data rates exceed those of low-power transmission protocols used in networks of IES. Thus it is ensured, that even in large-scale networks with high data transmission rates all collected information can be exchanged in time. This ensures that communications do not become a bottleneck. Additionally, since Wi-Fi can be operated in either 2.4 GHz or 5 GHz band, we can choose a band which does not interfere with the operations of the network which is monitored. This ensures, that the ARMS operates truly passively.

As all devices are part of the same network, they can be time-synchronized using the *Network Time Protocol (NTP)*. The Backend is configured as NTP server and can be equipped with a real-time clock module. Sniffer Units and AR Devices are NTP clients, which fetch time information from the server. Hence, we can utilize time stamps in data acquisition.

5.1.2 Usage and Usability Design

The modular design of the ARMS permits the serviceman to split and combine hardware units as needed to maximize usability. In the first exemplary use case, the serviceman is sent at the deployment site of a small network of IES to pinpoint and troubleshoot an error.

For this task, he can utilize an AR Device with an integrated Sniffer Unit. Even the Backend can be merged into the combined Sniffer Unit/AR Device, omitting the need to place any separate hardware unit and providing the serviceman with a single hand-held tool. While he might not be able to capture all network traffic with the single AR Device/Sniffer Tool, the serviceman is still able to inspect the IES of interest and its neighborhood. The single combined Sniffer Unit/AR Device is sufficient to capture traffic to and from the particular IES, which is currently inspected with the AR Device. Using this setup, the serviceman benefits from high mobility and zero setup time.

In another exemplary use case, a new network of IES has been deployed. The serviceman is sent on-site to confirm that all IES work as expected. As this inspection likely has to be repeated several times until it is verified that long-time stability of the network is given, the serviceman places multiple dedicated Sniffer Units such that they can capture all network traffic. Additionally, he installs a separate Backend. Depending on the estimated duration of monitoring those devices can be powered by battery or mains power. Utilizing the hand-held AR device, the serviceman can approach IES and check their status and connectivity. After having confirmed stable operations of the network over a sufficiently long observation period, he can remove the Sniffer Units and re-use them for observing another network. At any future point in time after having conducted modifications or upon encountering failure behavior in a network the ARMS can be re-applied for monitoring and troubleshooting. Again, the portable and modular design of the ARMS is beneficial as the serviceman can integrate it into his work flow conveniently rather than being forced into a certain work flow imposed by restrictions of the tool.

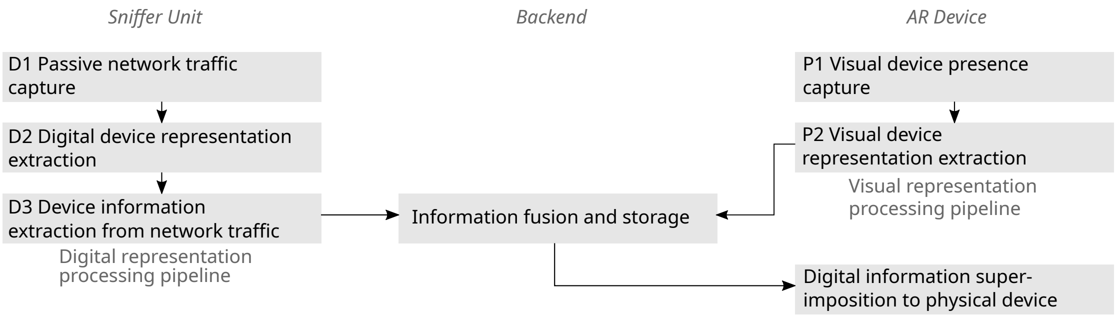

Sniffer Unit
Backend
AR Device
D1 Passive network traffic capture
D2 Digital device representation extraction
D3 Device information extraction from network traffic
Digital representation processing pipeline
Information fusion and storage
P1 Visual device presence capture
P2 Visual device representation extraction
Visual representation processing pipeline
Digital information super-imposition to physical device

Figure 5.3: ARMS processing pipelines mapped to Sniffer Unit, Backend and AR Device.

5.1.3 Digital Device Representation Pipeline

Each Sniffer Unit processes a *Digital representation pipeline*, shown to the left hand side in Figure 5.3. Its input stage captures network traffic. Next, it extracts communicating parties and associated information from captured traffic. Extracted data are then sent to the information storage at the Backend.

Block *D1 Passive network traffic capture* in Figure 5.3 needs to account for the diversity of transmission protocols, which can be encountered in networks of IES. Among those are e.g., *Bluetooth (Mesh)*, *802.15.4*-based protocols such as *6LoWPAN*, *Wi-Fi* or *LoRaWAN*.

Since OSI Layers 1 and 2 (Physical and Data Link Layer) differ among transmission protocols, we need hardware capable of passively capturing the transmission protocol of interest. Here, either Software Defined Radios (SDRs) or dedicated transceivers can be used. While the former offer great flexibility, the latter are more cost efficient, smaller and have lower energy consumption. Thus for our mobile application, they are preferable over SDRs. Most capture hardware outputs packets in *PCAP* format, which is the de-facto standard for packet processing [98]. This permits using established packet handling tools such as `Wireshark` [99].

In block *D2 Digital device representation extraction*, the *digital* representation of a device is obtained from captured network data. This representation consists of its identifier in the network and associated digital information. Again, we have to deal with various transmission protocols used in networks of IES which use heterogeneous primary addressing. For example, Bluetooth uses hardware addresses (Media-Access-Control addresses (MAC)) to deliver packets. LoRAWAN uses a Device-EUI (Extended Unique Identifier), often but not necessarily set to the MAC address. In contrary, 6LoWPAN utilizes IPv6 addressing. To homogenize this building block, we need a common digital device identifier, which must be derived from network traffic. This must be accomplished using only passive inspection of network traffic, since the ARMS shall neither require changes to IES firmware, nor interfere with network operations.

The hardware address (Medium Access Control (MAC) address) of a device is a suitable choice, as it is present in every networked device and typically static and unique within a network. Deriving the MAC addresses of network members from the primary addressing depends on the actual transmission protocol. Hence, processing block *D2* must be adapted to the network protocol of interest.

In block *D3 Device information extraction from network traffic*, the Sniffer Unit obtains information such as packets sent and received. In mesh networks, direct neighbors of a IES as well as the hops a packet takes are of interest. We treat hops as individual packets between neighboring nodes identified by MAC addresses, which we have derived from the primary network addressing scheme in block *D2*. In networks, which do not use mesh routing, the ARMS considers the hop-size to be one. Since the MAC address is known already from block *D2*, we can directly assign network information to a particular IES.

The ARMS has been designed to utilize multiple Sniffer Units to fully cover the network to be observed. However, their receive radii might overlap. Thus, a single packet might be captured by multiple Sniffer Units. If no actions were taken, the Backend, to which information extracted from packets are sent, would be cluttered with duplicate information, falsifying traffic statistics as well as information displayed to the serviceman. To prevent duplicate packets being stored in the database, I use the following procedure. Upon packet capture, the Sniffer Unit calculates a hash value of all data below the Network Layer using the Message-Digest Algorithm 5 (MD5). Those lower packet fields are static during the journey of a packet through the network. The Sniffer Unit transmits the hash value together with the packet data and the capture time stamp to the Backend. Using time stamps is possible since the Sniffer Units and the Backend are time-synchronized over the common wireless network, which they are all part of. Upon receiving a packet from a Sniffer Unit, the Backend can search its database using the MD5 hash value as an index, which permits efficient search. If a packet indexed with the received hash is found in the database, time stamps are compared. If the difference of time stamps is smaller than ε, the packets are considered duplicates and the freshly received packet is discarded. Typically, ε is chosen in the range of milliseconds. This accounts for varying packet travel times between message source and the Sniffer Units and especially for the precision limits of the time synchronization protocol NTP. Else, if the difference of time stamps is larger than ε, the received packet is added to the database.

I deliberately chose to let the Sniffer Unit create the MD5 hash and always send it together with the data to the Backend. One might argue, that those computations are wasted in case the Backend considers data as duplicate and discards it. While this is true, this design choice shifts computational burden from the Backend to the Sniffer Unit, preventing performance bottlenecks at the Backend. If packets were transmitted without the pre-computed MD5 hash, the Backend would be tasked with calculating hash values. With an increasing amount of Sniffer Units, the Backend would face high computational burden, which would ultimately require faster hardware, increase power consumption and reduce its portability. Thus, having the Sniffer Units calculate the hash value offers scalability, as this permits the serviceman to use as many Sniffer Units as needed to capture all traffic in the network.

The ARMS is protocol-agnostic in the sense that it is neither dependent on special message types nor information only available in a particular transmission protocol. With minor modifications to the digital device representation, extra transmission protocols can be added easily. Its homogenized output permits, that processing blocks, which are assigned to the Backend and the AR Device, need not be adjusted to a particular transmission protocol.

5.1.4 Visual Device Representation Pipeline

Every AR Device executes a *Visual representation processing pipeline*, which is shown on the right hand side in Figure 5.3. In step *P1 Visual device presence capture*, the actual presence of a device of interest needs to be detected. In the second step *P2 Visual device representation extraction*, the detected device needs to be identified. This distinction between presence detection and identification has an impact on the technologies used to perform those steps. Using for example *image recognition* with mature frameworks such as *OpenCV*, a device of interest can be detected reliably without needing optical markers [100]. However, identification of devices is difficult since networked devices share a similar visual appearance and are not distinguishable from another [101]. Additionally, image processing requires expensive computations, quickly draining the battery of the AR Device. Hence, *marker-based* approaches are favorable over markerless solutions. QR codes are optical markers, which permit both detection and identification of devices. They can be detected and read reliably by a camera. This offers great flexibility in choosing the capture hardware. Each IES needs to be supplied with a QR code. In the QR code, the MAC address of the device is embedded. Upon identifying a new IES by its QR code, the newly found device is announced to the Backend. There, we now have the MAC address as common device representation for both the digital and the visual world.

Adding QR codes has administrative overhead. However, most of the time, industrial devices are supplied with a printed sticker containing information on the device, so the QR code can be added easily. Additionally, unlike other approaches such as a IES blinking its MAC address in a Morse code way using an LED, as discussed in [102], we do not need modifications to the IES firmware. This brings high acceptance, as extending the firmware always comes at the risk of introducing new errors. Thus, using QR codes enables retrofitting of the ARMS to already deployed networks.

5.1.5 Information Storage at Backend

The Backend is responsible for *Information storage*, shown in the center of Figure 5.3. It receives extracted device information, such as message transfers from one or more Sniffer Units and visual device information from AR Devices.

The storage of the Backend is updated whenever new devices have been identified in steps *D2* and *P2*. The storage shall be the single central instance in the system, where information is merged and supplied. It must be ensured that no duplicate information is stored. Thus, the design of this storage is crucial for the overall system performance.

Visual and digital device representation are linked by the MAC address of a device. Thus, the MAC address is used as identifier for consolidating data from both domains. A suitable choice for storing information at the Backend is a database, which features fast write and read operations and is capable of handling simultaneous accesses, for example write operations of multiple Sniffer Units. Duplicate information can occur, if, e.g., the same QR code is scanned twice or if a single network packet is captured by more than one Sniffer Unit. In the case of scanned QR codes, the Backend searches its database whether the MAC address derived from the QR code is already present. To prevent duplicate packets being stored, Sniffer Units send their data to the storage combined with an MD5 hash identifier and a time stamp. The procedure to check, whether a packet received from a Sniffer Unit is a duplicate, has been described in section 5.1.3.

5.1.6 Output at AR Device

Digital information superimposition to physical device is the terminating processing block. It is shown on the bottom right in Figure 5.3. In this step, the AR Device obtains information from the Backend and annotates IES with those.

The ARMS shall support common hardware for AR Devices to provide flexibility in the implementation and remove the need of acquiring expensive dedicated hardware. An AR Device needs to be able to acquire combined visual-digital data from the information storage without requiring wired connections. Further, it must be able to detect and extract information from QR codes placed on IES. Simultaneous detection of multiple QR codes is needed, as this is a common situation encountered whenever multiple IES are within the camera image. Lastly, it needs to annotate a device identified by a QR code with the visual-digital data, using e.g., an overlay.

Due to their ubiquity and compliant hardware, smart phones are suitable AR Devices. However, one could also use a tablet computer or a notebook equipped with a web cam. I decided against using AR Headsets, as they are currently more expensive and less common than the solutions mentioned before. To create the annotation overlay on IES, the ARMS uses custom line draws. I could have also used established solutions such as *Vuforia*, *EasyAR* or *ARCore*. However, the the ARMS app only uses basic features of AR (recognition of QR codes and overlay view rendering). Thus, most features of the advanced AR SDKs are simply not necessary and would just clutter the app. Additionally, we are offered great flexibility and neither need expensive licensing nor cloud access, as it would be the case with the solutions mentioned beforehand.

5.1.7 Security Design

Besides 'natural' failure due to IES death by e.g., drained battery or bugs in the firmware, network operability can also be impaired by a cyber attack. For example, mesh networks can be subject to routing attacks, which cause network traffic to be misdirected or dropped [103, 104, 6]. Besides attacking routing protocols, network integrity can be assaulted by spoofing, i.e., copying a the identity of a IES to a malicious IES. This leads to network traffic being attributed or directed falsely by the routing protocol.

Handling MAC Address Spoofing While the MAC address is a convenient choice for IES network identity, it can be spoofed easily by an attacker, as it is publicly known. To prevent identity spoofing, *cryptographically secure identities* for IES need to be used. Such identities can be built using e.g., secret keys, public-private key pairs managed by a Public Key Infrastructure (PKI) or physically uncloneable functions (PUFs). If a secure identity is available, messages can be encrypted and then authenticated. The legitimate message sink node verifies that the sender really is the entity it claims to be. If this verification succeeds, the receiver decrypts the message contents. Else, it discards the message.

Working with Secured Networks The ARMS has been designed such, that it can be used with secured networks. To be able to verify the secure identity of a device, a Sniffer Unit needs to be supplied with this information. For example, each IES could have a public-private key pair. The private key is used to sign messages, while the public key is used to validate signatures by message recipients. Thus, the ARMS needs to know the public key of every IES, e.g., by integrating the ARMS in the PKI. Using the public key of a IES, the ARMS is able to validate message signatures of captured traffic and thus confirm the identity of a IES. If validating the authenticity of a IES fails, the ARMS concludes that the node uses a spoofed network address and issues a warning to the serviceman, revealing spoofing attacks.

Handling Attacks on Visual and Network Representations Targeting operations of the ARMS itself, an attacker can attempt to copy or forge the visual representation of IES, i.e., the QR codes, together with its network representation, the MAC address. The ARMS creates a single visual device representation when the serviceman scans the first QR code. Further, it sniffs traffic from both the legitimate and the malicious IES. As the ARMS has been supplied with public keys, it validates the signatures of all messages. Assuming the attacker has not stolen the secret key of the IES, whose identity has been copied, the malicious node is not able to forge message signatures. The ARMS tries to apply the public key of a legitimate IES to validate the signature but will fail, as the public key used does not match the secret key used by the malicious node to sign messages. As a result, the ARMS reports the failed signature validation to the serviceman.

In the case that the **malicious IES does not transmit messages**, the signature validation is of no use. Traffic originating from the IES, whose digital and visual representation has been copied, pass the signature validation as public and private key match. However, if no countermeasures were applied, the ARMS would visualize traffic originating from the legitimate IES at the copied node, too. To prevent this, the ARMS issues a warning upon detecting duplicate QR codes. In order to distinguish the legitimate from the malicious IES, signal strength measurements can be conducted. Approaching the legitimate IES, the signal strength of sniffed packets attributed to this IES increases. Moving from the legitimate to the malicious node, signal strength of those packets drops since the legitimate node either sends no packets at all or packets originating from it fail the signature validation and are thus discarded by the ARMS.

In the case that a **QR code is copied**, but the malicious IES uses a **unique network representation** (i.e., one that is different from any other encountered in the network), the situation is slightly different. The malicious node can now send messages using its own network representation. However, those messages fail the signature check, as the public key matching this network address is not known to the ARMS. Network traffic from the legitimate IES would still be visualized at the malicious node, but as in the example above, the ARMS detects duplicate QR codes and issues a warning.

If an attacker uses a **forged QR code, but has copied the network address of a legitimate IES**, the serviceman can add the malicious nodes visual representation to the Backend. Traffic originating from the malicious node fails the signature check and the ARMS issues a warning to the serviceman. As the visual representation of the forged QR code does not match the network address of the malicious node, no traffic is visualized to or from the malicious node, which is noticed by the serviceman. Again, by observing the change in signal strength of legitimate packets while moving between the optically indistinguishable IES aids in identifying the malicious node.

Lastly, **both visual and network representation** of a IES can be **forged**. Network traffic originating from the forged node fails the signature check, resulting in no traffic being visualized at it. As before, the ARMS reports the failed signature verification attempts. Pointing the AR Device at the malicious node, the ARMS hints that this is the IES whose signature checks failed repeatedly.

To sum it up, MAC addresses are the common representation for networked devices, which can be used with least effort and maximum flexibility. However, cryptographic means must be utilized, otherwise attacks such as IES spoofing are possible. The ARMS has been designed such, that it can be used with both secured and insecure networks. It can handle cryptographic IES identities and aid the serviceman in detecting attacks resulting from spoofed IES.

The ARMS utilizes cryptography to prevent remote attacks over the network. The Backend as the central point of communication creates a secured wireless access point. All RESTful Hypertext Transfer Protocol (HTTP) requests to the Backend require an Authentication Header. The authentication is done via user name and password. Every user name is assigned a role in the database at the Backend. For every API endpoint and every HTTP method, we specify, which roles are allowed to access this endpoint. If multiple Sniffer Units or AR Devices are used, each of them has individual login credentials. If a single device is compromised, we can revoke the compromised devices permissions. Further, we can provide fine-grained access to information at the Backend. A senior serviceman could be permitted to add new nodes to the Backend, while a trainee might be given read access only.

Currently, the ARMS is tailored towards network monitoring and troubleshooting. However, having incorporated protection against node spoofing attacks and by utilizing secure communications, the ARMS is fit for usage in a hostile environment.

5.2 Implementation

In this section I describe the proof of concept implementation of the ARMS. Since the original publication [90] the implementation has evolved. The most recent version is introduced in Chapter 6 where it is used in a usability study. Here, I am only going to discuss the most important aspects of the original implementation required to understand the evaluation in the subsequent chapter 5.3.

5.2.1 Sniffer Unit

Sniffer Units consist of a medium-performance machine running Linux (such as a Raspberry Pi). This machine can use its internal adapter to capture 802.11 Wi-Fi (Wi-Fi) traffic. To capture additional wireless protocols transceiver chips are attached to the Linux machine over Universal Serial Bus (USB). For example, to capture 6LoWPAN traffic a *CC2650* microcontroller (MCU) by *Texas Instruments* is used. Captured packets are transferred from the transceiver to the Linux machine, where they are passed into a packet analyzer such as *pyshark*, a Python wrapper for *Tshark* [105].

There the captured packets are being dissected into their layers and MAC addresses are extracted. The wireless protocol being captured by the Sniffer Unit might utilize mesh networking and thus packets might take multiple hops. In this case, the packet analyzer extracts the routes of the packet and treats each hop as separate packet.

5.2.2 Backend

The Backend is implemented on a powerful Linux machine. It creates a Wi-Fi access point and exposes its services, e.g. read and write access to the packet and device database and time synchronization via a a RESTful application programming interface (API). For information storage, a *PostgreSQL* database managed by the *SQLAlchemy* toolkit is utilized [106, 107]. Using the RESTful API the Sniffer Unit and the AR Device can interact with the Backend, for example add captured packets to the database or retrieve the list of known IES to visualize them.

5.2.3 AR Device

The original handheld AR Device was implemented on a *Samsung Galaxy S7* smart phone running *Android OS* 8.0.

The camera of the smart phone captures the environment and shows the live image at its screen. To detect and read QR codes, the *zxing* library is used [108]. Using the built-in camera of the smart phone, the 2 cm by 2 cm sized QR codes can be detected reliably at distances up to 1 m and angles up to 45°.

If an IES has been identified by the camera, but was not found in the database at the Backend, the user creates a new digital representation of that IES by using the touch screen of the AR Device. Similarly, information on a existing IES (e.g., location) can be altered.

For creating the overlay on its screen, the AR Device requests data from the Backend. Our custom visualization uses Android graphics canvas objects to draw a line between different node views. The position of start and end of the line only depend on the current position of the node views. Hence, only little computational effort is needed to draw these views. Together with efficient QR code detection, AR Device battery is preserved. In order to visualize *asymmetric* connections, we draw a bezier curve between the two IES. An arrow indicates the direction of packet flow. Arrow stroke width increases in a logarithmic fashion with traffic density.

Figure 5.4 is a capture of the screen of the AR Device, showing the overlay. There are

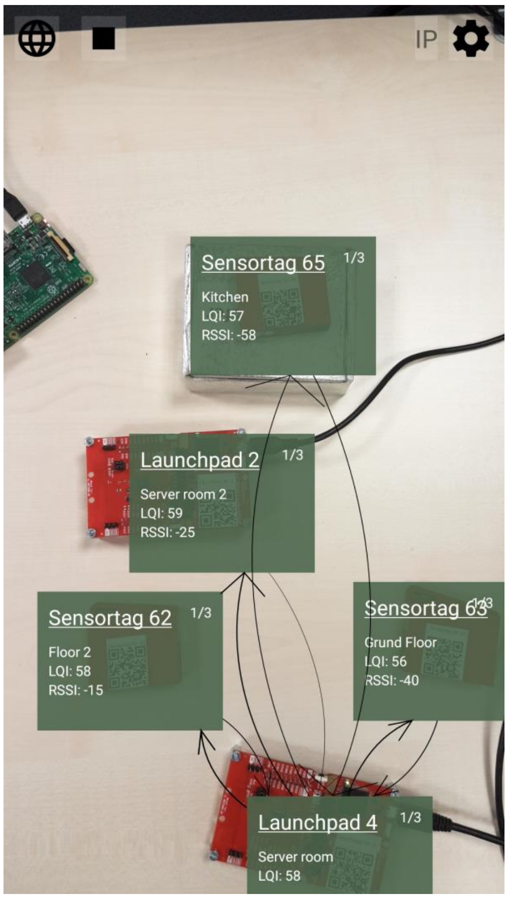


Figure 5.4: Arrows show 6LoWPAN traffic between IESs and network server (Launchpad 4).

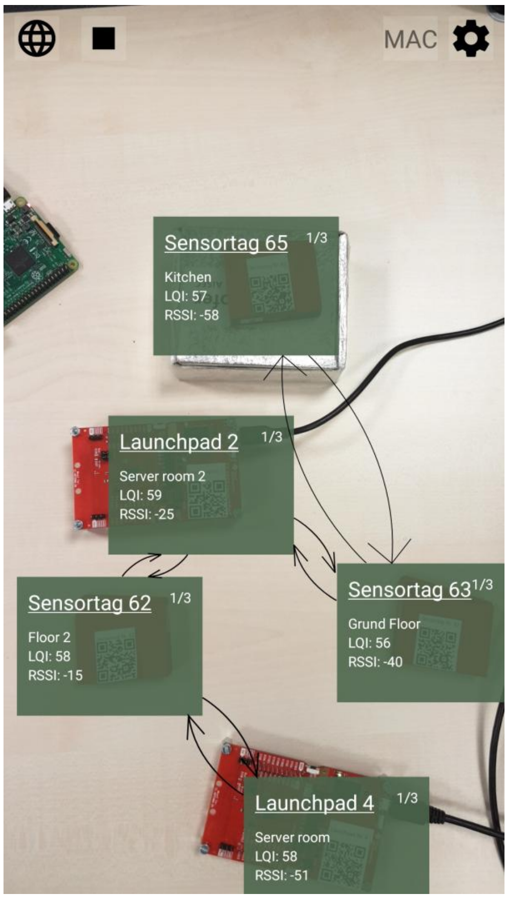


Figure 5.5: MAC-address based traffic view reveals hops.

four IES (Sensortag 65, Launchpad 2, Sensortag 62 and Sensortag 63) and the network server (Launchpad 4). Each device is identified by its QR code and an overlay is placed above the QR code, showing the name of the device and additional information. Network traffic between IES and the server is shown as arrows. It can be seen in Figure 5.4 that there is bi-directional traffic, i.e. the server responds to a received packet.

5.3 Evaluation

In this section, I describe the usage of the ARMS in a real network of IES. The test bed consists of six *CC2650 Sensortag* and *CC2650 Launchpad* IES. Devices transfer messages using the transmission protocol 6LoWPAN. Routing in the network is done by the *Routing Protocol for Low power and Lossy Networks (RPL)*, which creates a dynamic mesh network [109]. IES firmware utilizes the *rpl-udp* example, which is part of the *Contiki-NG* operating system [110]. The example contains both a *client* and a *server* implementation. One CC2650 device is flashed with the server firmware, all others with client firmware. Each client sends User Datagram Protocol (UDP) packets with an increasing number in fixed transmit intervals of 10 sec to the server. The server replies with the same number to the client. CC2650 devices have been placed in an area of $1\,m^2$ inside an office building. Each CC2650 is supplied with a printed QR code containing the wireless interface MAC address of the device. No further modifications to the sensor network were performed.

A single Sniffer Unit captures all 6LoWPAN network traffic and forwards captured packets to the Backend. The Backend is placed such that Wi-Fi connectivity between Sniffer Unit, AR Device, and Backend is given at all times.

5.3.1 Investigating Hop Behavior

While monitoring a 6LoWPAN network, the ARMS can show traffic either based on Internet Protocol (IP) addressing or MAC addressing. The former is shown in Figure 5.4 and can be used to confirm that every IES has a connection to the network server (Launchpad 4). To gain deeper insight into the network, the user can switch to MAC addressing to reveal the hops a packet takes. This is shown in Figure 5.5. It can be seen, that packets from device Sensortag 65 are forwarded by every other device, until they reach their destination at Launchpad 4. If no arrows originate from or terminate at an IES , this indicates that the IES has failed. Routing anomalies can be grasped intuitively, if e.g., some devices exchange traffic but have no route to the server, as this should not happen in normal network operations.

Besides helping the user to identify sources of failure, the MAC-address-based view can be used to optimize network topology. The user might identify a single device, which forwards traffic from many devices to the server. Such forwarder devices are a potential source of network failure, as forwarding dense traffic drains their battery quickly. This brings the risk of network parts becoming isolated. To prevent this, the user can, e.g., re-arrange device positioning or add a new IES to provide a second route to the server. Effects of those placement optimizations can be observed in real-time with the ARMS.

The user can choose, how long arrows between communicating sensor nodes are displayed. With a short duration *single message* transfers can be visualized. After the message transfer has finished, arrows disappear. By choosing a longer duration *traffic density* can be visualized. Message transfers on a single route are summed up, increasing arrow thickness as more traffic is aggregated over time. This can be exploited to identify

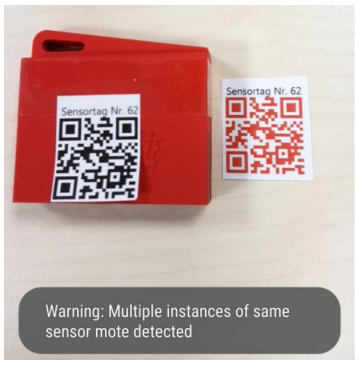
IES, whose traffic rates are unexpectedly high or low, which could indicate malfunction or an attack.

5.3.2 Duplicate QR Codes

Duplicate QR codes can occur due to errors during creation and placement of those. Further, it can not be ruled out that there are duplicate MAC addresses in networked devices. This will eventually lead to collisions while routing network traffic. Duplicate QR codes can also be caused by an attack, in which an attacker tries to spoof the visual representation of another sensor node by intentionally placing duplicate QR codes. This could trick the ARMS into attributing traffic falsely. Thus, on detecting duplicate QR codes, the ARMS issues a warning and prevents the user from adding information to either IES identified as duplicate, until the ambiguity has been resolved. This warning is shown in Figure 5.6. It must be noted that in order to verify which of two sensor nodes with the same MAC inscribed in the QR code is authentic, cryptographic solutions are needed. This has been discussed in Section 5.1.7.

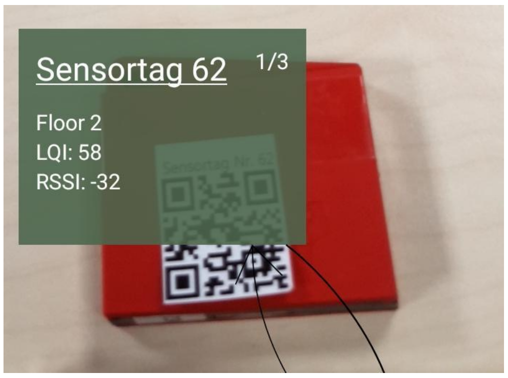
5.3.3 Handling Devices Out of Sight

Depending on device placement, it can occur that only an IES can be captured by the camera of the AR Device. To be still able to see traffic flows between the device in view and its neighbors, the ARMS tracks the rotations of the AR Devices and remembers, in which direction an adjacent device has been seen previously. Thus, traffic flow can still be drawn between the IES, which is captured by the camera right now, and the last known position of another IES not visible anymore. This behavior is shown in Figure 5.7, where the AR Device has been rotated such that the neighbor of Sensortag 62 moved outside the lower boundary of the display. Arrows indicating message flow can still be seen between Sensortag 62 and the last known location of its neighbor. This behavior enables the user to utilize the ARMS even in spatially extended networks, tracing hops in a 'bread-crumb' manner.



Warning: Multiple instances of same sensor mote detected

Figure 5.6: Duplicate QR codes in the camera field of view (FOV) raise a warning.



Sensortag 62 1/3

Floor 2
LQI: 58
RSSI: -32

Figure 5.7: Traffic flows to devices outside the camera FOV can still be visualized.

5.3.4 Investigating Network Behavior over Time

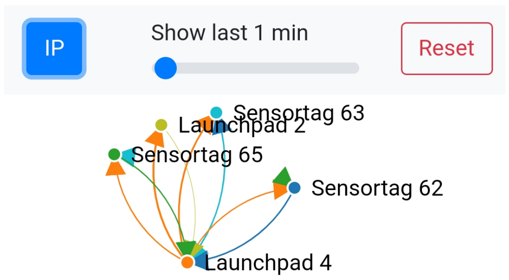


Figure 5.8: Full network graph permits viewing network behavior over time.

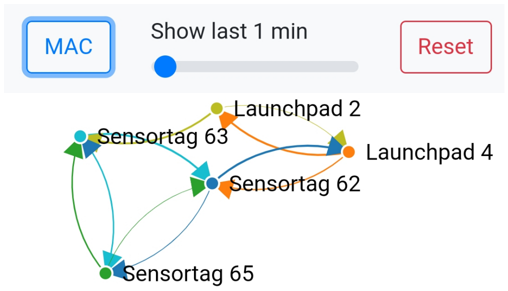


Figure 5.9: Full network graph also provides per-hop view.

As described in Section 5.1.1, captured packets are attributed with a time stamp. To be able to see how network topology has changed over time, the user can utilize the full network graph view shown in Figure 5.8. This view gives an overview of the full network, showing all traffic captured either based on IP addressing as shown in Figure 5.8, or using MAC-based addressing, again revealing hops (Figure 5.9). The user can step through time and, e.g., pinpoint, when a particular IES has failed and which effects the failure had on network topology. By using this view together with the MAC-based overlay shown in Figure 5.5, a network problem can be traced down both temporally and spatially, supporting the troubleshooting process.

5.4 Related Work

My ARMS incorporates technologies from many domains, such as network data acquisition and information extraction, physical device detection and visualization techniques. I focus this discussion on works, which are concerned with *passive network data acquisition* and such which focus on *Augmented Reality-based visualization* of networks.

[95, 111] discuss the design of a sniffer network for passive network observation. Unlike my ARMS, which lets the *user* interpret what they see, [95, 111] try to analyze the cause for node failure. Further, their approach displays network topology and additional information on a central screen, being more similar to classical network monitoring solutions. In contrast, my ARMS is tailored towards on-site network monitoring and troubleshooting utilizing a hand-held device.

[112] implement an AR interface for wireless networks on a hand-held device. Their system visualizes readings of humidity sensors placed inside walls, the location of sensors being indicated by a unique marker placed at the wall. A similar system has been described in [113]. Both approaches have in common that the hand-held device which visualizes sensor data has been *integrated* into the network to be observed. Thus, retrofitting is not possible without altering the already deployed network.

[102, 114, 115] present an AR-based approach which visualizes links and traffic between devices as well as information on devices being observed. They use a web cam for capturing the physical world and a touch screen for displaying the overlay. The touch screen permits interaction with the network. For example, the user is able to "cut" the connection between two devices by performing a cutting gesture through the virtual cable drawn in between. However, this interactivity comes at the cost of the AR hardware having to *actively interact* with the network. Thus, the system can not be retrofitted to existing networks without modifying the firmware of networked devices.

[116] uses an AR smart phone app to visualize antenna radiation patterns. Devices are identified by optical markers. Their experimental study shows that the marker-based device identification is a suitable approach and that visualization truly helps users to understand what is happening. However, as they are concerned with radiation patterns rather than network monitoring, they require a specialized setup using SDR wired to a processing server. Additionally, they need special software on the SDR to provide information on radiation patterns.

[117] place an overlay over interconnected IES, showing network topology and data flows. They use marker-based device identification. Usability of their approach is limited by the fact that they require a wired connection between every mobile device and the visualization device. Additionally, software at each mobile device must be modified to provide information on the packets being sent. Lastly, each device being monitored must be able to communicate with the visualization device at all times. This is a major drawback, since devices can fail for many reasons. This undermines the exact purpose of an AR visualization approach, which is to help users identify and fix failed networked devices.

[118, 119] use a Microsoft HoloLens to show network topology visualizations. As they have only released a demo abstract, a set of slides and some demonstration videos it is difficult to assess their system in detail. It seems, however, that their approach is tailored towards a given ZigBee network. Additionally, they require dedicated AR headsets while my ARMS runs on commodity hardware such as smart phones and single board computers (SBCs).

Compared to related works, my ARMS has been designed for usability and security. Unlike prior work, in which observed devices are queried actively, my contribution uses passive observation only. This approach is significantly more challenging, as the ARMS can only extract relevant information from observed network traffic. Thus, my system has neither internal knowledge of the network, nor the possibility to directly query an observed device for information. On the other hand, passive observation is beneficial, as it permits retrofitting my system. I consider this an important contribution for several reasons. Firstly, firmware of observed devices does not need to be extended. By keeping firmware sleek potential sources of error in the IES are reduced. This increases IES availability, which, as shown in Chapter 3, is a paramount safety and security goal of such networks. Secondly, by minimizing the communication interface capabilities in the observed IES, potential entry points for attacks are minimized. This goes hand in hand with my detailed security concept, which to the best of my knowledge makes my system the first ARMS designed with security in mind. Its modular low-cost hardware and software design utilizing AR supplies the user in the field with just the information needed to solve a specific problem.

5.5 Conclusion

In this chapter I presented an augmented-reality based network monitoring system (ARMS), a protocol-agnostic and portable AR tool to aid users in analyzing and troubleshooting networks of IES on-site. The ARMS utilizes one or more Sniffer Units, a Backend and one or more AR Devices. The mobile Sniffer Units are placed temporarily among IES to be observed. They passively capture network traffic, thus no modifications to a given network are needed. The low-cost and modular design of the Sniffer Units permits efficient coverage of spatially extended networks. Support for additional wireless transmission protocols can be added easily. Visual device representations are extracted from QR codes captured by the camera of AR Devices. The central Backend consolidates information extracted from network traffic and visual device information. Those are fed into the AR Device, which provides real time visualization of connections, traffic flows and multi-hop behavior.

I implemented a real sensor network utilizing 6LoWPAN/IEEE 802.15.4 transmission protocol. Using this network, I explored scenarios commonly encountered in network troubleshooting. The results show, how the ARMS aids the user in the field to pinpoint and fix those failures. The application shows what is actually happening without interpreting the results. The field of view of the camera acts as a user-friendly filter to the information displayed, as my monitoring system only shows information on the devices captured by the camera. This provides an unobstructed view on parts of the network which are of interest to the user without cluttering their view. Actually *seeing* the otherwise imperceptible wireless network traffic fosters understanding and permits intuitive interaction with the observed network, thus satisfying CRANOR'S design principle $\boxed{2}$.

6 Educating Users in Wireless Network Security Using Augmented Reality-Based Traffic Visualization

From the case studies in Chapter 3 we learned that in many application scenarios users (such as firefighters, servicemen or patients) are no security experts. Nevertheless, they are operating networks of low-power wirelessly interconnected embedded systems (IES). To fulfill CRANOR'S design principle $\boxed{3}$ we need to educate those users in the complex fields of wireless networking and security.

In Chapter 5 I have shown how network traffic can be visualized by using augmented reality (AR). Simple network failures such as malfunctioning devices could be noticed by the absence of traffic information in the AR overlay. My hypothesis was that the augmented-reality based network monitoring system (ARMS) is so intuitive that people with no or little background in wireless networks are able to *see attacks* on the wireless network. This could effectively educate users in wireless network security with a low barrier of entry and hence satisfy design principle $\boxed{3}$. To explore this I conduct a user study in which I investigate the benefits of using AR in wireless network security education[1] .

Wireless network security is typically taught in university lectures and accompanying laboratory exercises using tools with a steep learning curve, such as `wireshark` or `tcpdump` [7, 8]. Those network monitoring tools merely provide an abstract view of the network by providing text, topological graphs, or diagrams. However, crafted network packets can alter the network topology and thus affect *all* machines in the network. If the network is inspected from a *single* machine, the overall impact of the attack might be shadowed by the perceived limited angle of view provided by those monitoring tools.

In contrast to that, AR can provide a more immersive view of the network from a user-controlled birds-eye perspective. As stated in Chapter 5 AR permits to join information from the visual world and the radio frequency (RF) spectrum. In this case, a visualization of otherwise imperceptible network traffic can be superimposed on the camera image of the physical devices communicating with each other. The user can focus on selected parts of a wireless network by only capturing the devices of interest on camera, which provides a intuitive way of filtering information.

[1] This chapter is based on the following publication:
[120]: Martin Striegel, Jonas Erasmus, and Parag Jain. Evaluating Augmented Reality for Wireless Network Security Education. In *Proceedings of the 2021 IEEE Frontiers in Education Conference (FIE)*, Lincoln, Nebraska, 2021

Studies on the use of AR in the classroom in other areas of application have shown that this fosters both motivation and the student's understanding [121, 122, 123].

In this chapter I show that we can leverage those positive effects using AR in *wireless network security education.* I propose a laboratory setup and conduct a user study with 25 participants. They observe a Wi-Fi network through the AR application while the attacks *Evil Twin*, *Address-Resolution Protocol (ARP) Poisoning* and *Denial of Service* are launched on the network. Participants are tasked to detect and describe the attacks while they take place.

Remarkably, only by observing the changes in network topology, all participants were able to spot and describe the effect of all attacks. Hence, I conclude that AR is a valuable addition to traditional ways of teaching wireless network security. Furthermore, my results indicate, that AR-based network traffic visualization could permit access to the complex topic of wireless network security outside universities.

The contributions of the original publication were as follows:

- I designed and implemented an AR application tailored to wireless network security education, which helps users to spot attacks on a wireless network (Section 6.2).
- I experimentally evaluate this application with 25 participants (Section 6.3).
- Based on my results, to the best of my knowledge, the original work was the first to demonstrate that AR-based traffic inspection is beneficial in wireless network security education.

6.1 Related Work

My approach draws from previous work on AR-based visualization of networked devices, general engineering education, and cybersecurity education.

AR-Based Device and Network Observation [112] and [113] propose AR interfaces to visualize sensor readings acquired by interconnected embedded systems. [115, 102, 114, 117] additionally visualized network links between devices. Those solutions have in common that they acquire information on devices and network links *actively* by being actively engaged in the networks themselves. In contrast, my ARMS introduced in Chapter 5, acquires network data passively and thus my approach does not interfere with the attack being observed. For this chapter, I extended the architecture of the ARMS and added functionality to keep track of interrelated series of network packets. By doing so, the updated application is able to visualize *roles* of devices in the network and *attacks*, which consist of multiple packets.

Engineering Education with AR [122] found that AR improves the learning experience by providing a high level of interactivity and a shared educational experience while students interact with topics which can not be experienced in the real world.

[123] proposed an AR application to teach electromagnetism. The authors also found, that AR provides a positive shared interactive experience, as students exert control over the view and can manipulate objects.

[116] and [124] employed AR to visualize antenna radiation patterns in an overlay placed on the camera image of an antenna. This permits students to understand antenna characteristics and the effects of beamforming. The authors found that their approach increased the interest of students as well as reported learning gains.

Cybersecurity Education Cybersecurity laboratory design and the use of AR in education have been discussed in several studies. However, none of those works has investigated the beneficial effects of AR in wireless network security education.

[7] and [8] have shown that a game of attacker and defender is motivating to students. In their works, the defender is the system administrator who has to prevent attacks by securing the system *beforehand*, e.g., by configuring the firewall. AR is not employed. In my work, the defender is provided with the AR application and tries to detect and describe attacks *while they happen*. This gives the defender a more active role during the attack and provides a close interaction between attacker and defender.

In [125], an AR-based game for teaching smartphone application security and privacy to high-school students was developed. In the game, common threats to social media "attack" the student, who chooses and wields an appropriate wooden shield with an AR marker to fend off the threat. In their study, the authors find that this serious game increases the self-awareness of students regarding cybersecurity and develops their critical thinking about technology.

[126] proposed a laboratory course which leverages Software Defined Radios (SDRs) to teach concepts related to wireless security such as eavesdropping and jamming. In a final competition students tested their implementations by trying to eavesdrop on a 802.11 Wi-Fi (Wi-Fi) wireless link with different levels of protection. The authors found that this is highly motivating for students. As their laboratory course is about SDR, they focused on attacks close to the physical layer instead of network *packets*, neither did they utilize AR.

While the visualization benefits of AR in engineering education have been demonstrated in previous works, there is a research gap in the application of AR in wireless network security education with its particular need for *situational awareness*. Users must be trained to detect an attack, how the attack is conducted and the implications of the attack [127]. Due to the complexity of wireless network protocols those goals are difficult to achieve by merely presenting, e.g., networking sequence diagrams in class. Thus I intended to utilize the motivating effects of AR in education [123, 116, 124] as well as the competitive aspects of attacker and defender games [126, 7, 8, 125] to provide a inciting way to wireless network security education, which, to the best of my knowledge, has not been studied before.

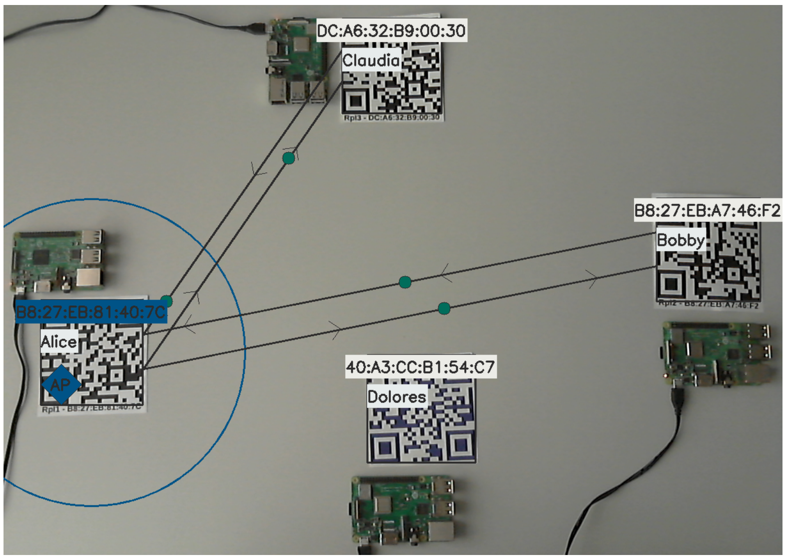


Figure 6.1: Monitoring a network of four devices. All communicating devices are captured in the camera image. Claudia and Bobby exchange messages, which are routed by Alice. Alice transmits a beacon frame.

6.2 Augmented Reality Network Traffic and Attack Visualization

6.2.1 Overview

Figure 6.1 shows a screenshot of my AR application, the camera being pointed at a Wi-Fi network with four devices. Every device is identified by a Quick Response (QR) code which contains the Medium Access Control (MAC) address of the device. If the AR application captures a wireless packet, whose source and/or destination address correspond to a device visible in the camera image, the application superimposes traffic information on the camera image[2]. Here, the topmost device and the device to the right communicate with the access point (AP) on the left side of the figure. Every colored dot between devices corresponds to one Wi-Fi network packet using a unicast address, arrows indicating the direction of data flows. The circle around the AP device represents a packet send to the broadcast address.

A device of interest might not be captured by the camera image. In this case the user can select the device from a list of known devices, which is composed of all known MAC addresses, and drag it into the camera image to an arbitrary location. It is then treated as a virtual device, as shown in Figure 6.2. Thus, the user is still able to inspect traffic flows from and to this device.

[2]The application assumes, that MAC addresses remain static, as randomization would cause a mismatch between identifiers derived from QR codes and those found in network packets. I consider this acceptable for a laboratory setup.

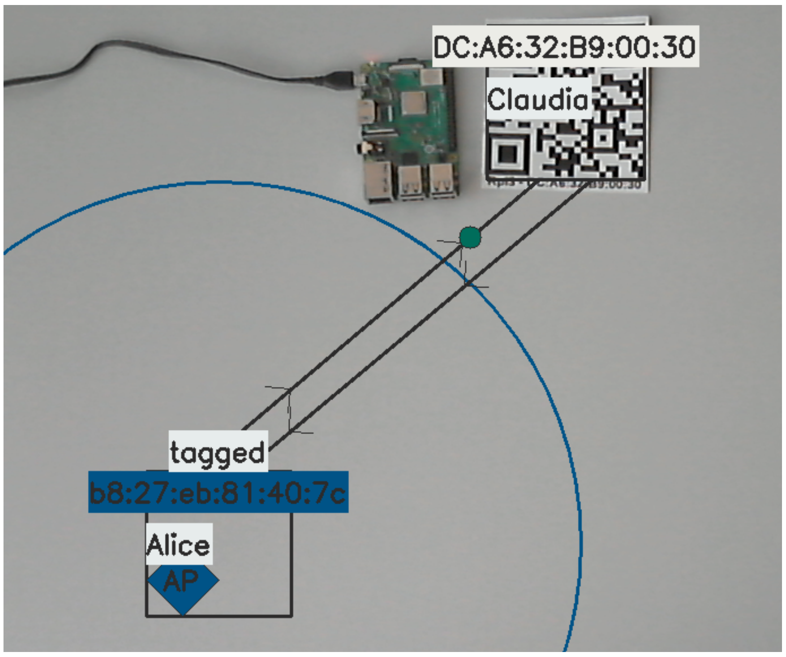


Figure 6.2: Monitoring a single device (Claudia), whose communication partner (Alice) is not captured by the camera image and thus represented by a virtual device.

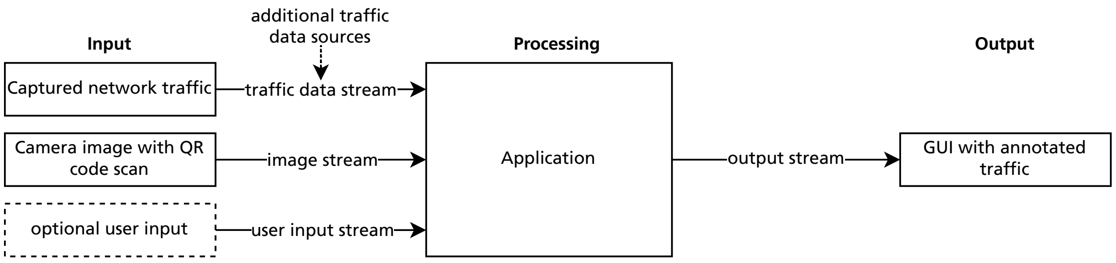


Figure 6.3: Architecture: Data flows in the AR application

6.2.2 System Design and Data Flows

In the user study I use an updated implementation of the ARMS introduced in Chapter 5. Figure 6.3 shows data flows in the AR application, which are discussed in detail in the upcoming section. The most recent implementation of the application is written in *Python3.7*, using *OpenCV*[3] for image acquisition and processing, *pyzbar*[4] for QR code detection and *scapy*[5] for capturing and manipulating network packets. It can hence be run on all devices, which support Python3 and which are capable of capturing a video stream and network traffic, either using the internal adapter or an external capturing device, e.g., connected via Universal Serial Bus (USB) or Ethernet.

[3]`https://pypi.org/project/opencv-python/`, last accessed 2021-12-30
[4]`https://pypi.org/project/pyzbar/`, last accessed 2021-12-30
[5]`https://pypi.org/project/scapy/`, last accessed 2021-12-30

Data Input There are three sources of input: network traffic, camera images, and user supplied information. Network traffic is captured passively to not interfere with the operation of the network, which is being observed by the application. Traffic is transformed in a data stream and forwarded to the application for processing. This modular architecture permits capturing different wireless protocols as well as wired protocols by adding a receiver for that protocol. Additional packet sources, such as external traffic monitoring systems, can inject packets into this data stream to provide additional information.

Camera images are captured and fed into the application via an image stream. If desired, the user can input additional information such as friendly names and the roles of particular devices such as the APs using a configuration file. Furthermore, the user can also control the application by performing mouse clicks in the displayed video stream to open and close annotations. Those actions are written in the user input stream and fed into the application.

Processing Device identifiers are extracted from the QR codes found in a camera image. Identifiers are also extracted from the source and destination fields of captured network packets. Thus, in general, all types of packets which include a source and a destination field can be processed by the application. A wireless protocol might use specific addresses, e.g., multicast or broadcast addresses in Wi-Fi. These protocol-specifics are stored in a protocol configuration file which tells the AR application how to handle the addresses. In case of, e.g., a broadcast packet, the application interprets the destination as a virtual broadcast-device and draws the overlay.

If a new device identifier is encountered, either in the network traffic or in an optical marker, the application adds this identifier to the list of known devices. If a known device identifier is found in the source or in the destination field of a captured network packet, traffic is drawn in the overlay. If both source and destination devices are visible in the camera image, a direct connection is drawn between them. If only either source or destination is seen, the invisible device is represented as a virtual device, so traffic flows can still be visualized.

By analyzing network traffic, the application can derive special roles of devices and keep track of events, which consist of multiple packets. For this, the application uses packet filtering and stateful decision trees, which encode network behavior. Packet filtering differentiates between control and data packets. For example, to find the AP in a Wi-Fi network, the application searches for Wi-Fi beacon frames. A device which sends a number of those frames is identified as an AP by the application. If multiple devices advertise the same network, this is fed into a decision tree, as it could indicate an *evil twin* attack. If subsequently a de-authentication packet is captured by the application, it is likely that an evil twin attack is taking place.

This straightforward classification is suitable for a wireless laboratory setting, where textbook attacks are being taught and experimented with. To be useful in a real-world setting, the modular architecture permits machine-learning based mechanisms for sophisticated network traffic analysis to be added in the processing routines.

Output The application outputs a video frame, which consists of the original camera image with superimposed information on network traffic and attacks. The goal is to provide a clear, yet informative view on the network. Initially, the application displayed additional information about the packet content on every packet. As many devices might be captured by the camera simultaneously, the amount of information displayed needed to be balanced to not lose important information but also not to clutter the view. To find this balance I conducted a pilot study with ten participants to understand which visualization of traffic is perceived most helpful. In result, every unicast packet is shown as one dot without additional information, as participants commented that more information clutters the view too much. Participants also liked that Wi-Fi packets, which are sent to the *broadcast address*, are shown as circles. They found the graphical distinction between unicast and broadcast traffic on a logical layer helpful and precise, despite every wireless packet being of broadcast nature on the physical layer.

The application can provide various levels of visual guidance. By default, only network packets exchanged between devices are visualized as one white dot following a directed line to the destination device. The user can choose to activate filtering and the decision trees. Then, the application colors packets and annotates special roles of devices. For example, data packets are display in green, while Wi-Fi de-authentication packets are colored orange. Devices which the application found to have special roles, e.g., the Wi-Fi AP or a device having been subject to an attack, are marked by a colored diamond.

If the user has supplied additional information on devices, such as friendly names of devices or device roles, those are also shown. This set of customization options permit the AR application to be tailored to the laboratory course and the varying skill levels of the students, e.g., by providing hints.

6.2.3 Laboratory Testbed

The testbed consists of four Raspberry Pi 3 named Alice, Bobby, Claudia and Dolores for easier reference. Every device has a QR code optical marker that shows its MAC address. Alice acts as a Wi-Fi AP, while Bobby and Claudia are connected to the AP as clients. We use Wi-Fi in the testbed, as it is the prevalent wireless protocol and thus typically taught in university courses.

Bobby and Claudia exchange messages over User Datagram Protocol (UDP) once per second and those messages are routed by Alice. Every message is acknowledged by the peer. Dolores is the attacker and is also a client of the wireless network. The test supervisor, who takes the role of the attacker, can access her via Secure Shell (SSH) and launch attacks on the network. Using *scapy*, I implemented a set of attacks typically used in Wi-Fi network security laboratories [8].

- *Evil-Twin* attack: De-authenticating a device from a Wi-Fi network, advertising a malicious AP with high signal strength and tricking the de-authenticated victim into joining the malicious AP (Figure 6.4).
- ARP poisoning: Creating a mismatch between IP and MAC addresses. This results in sending of packets to the attacker instead of the legitimate receiver (Figure 6.5).

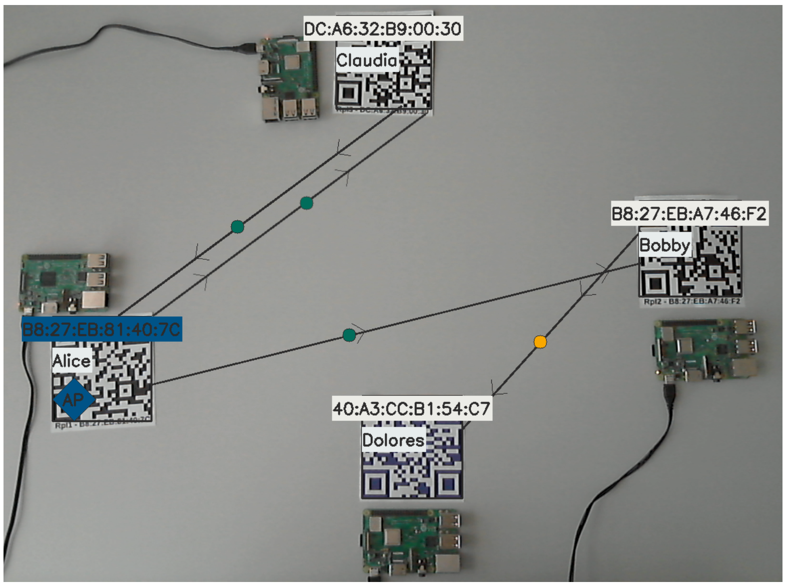


Figure 6.4: Dolores performs an *Evil Twin* attack by de-authenticating Bobby from Alice's AP, then tricking Bobby into joining Dolores' malicious AP. Alice still considers Bobby to be part of its network. Bobby tries to send packets to Claudia, but Dolores does not forward packets. Due to having emitted a de-authentication request followed by Wi-Fi beacons, our application considers Dolores to be an attacker and thus colors the packets to and from Dolores orange.

- *DoS* attack: Overloading the victim device by sending packets at a very high rate. This might also overload the AP or prevent other devices in the network from sending packets, as the wireless channel is overloaded (Figure 6.6).

6.3 User Study: Is AR Beneficial in Wireless Security education?

Purpose and Scope Students were given the ARMS application to observe a Wi-Fi network. They were tasked to spot attacks on the wireless network launched by the supervisor and to describe cause and effect of the attacks. The research questions investigated in this study are:

1. Does AR-visualization provide interesting information on a wireless network in a comprehensible manner?
2. Does visualization of wireless networks and attacks permit students to spot attacks?
3. Does the visualization enable them to explain the underlying mechanism of an attack?

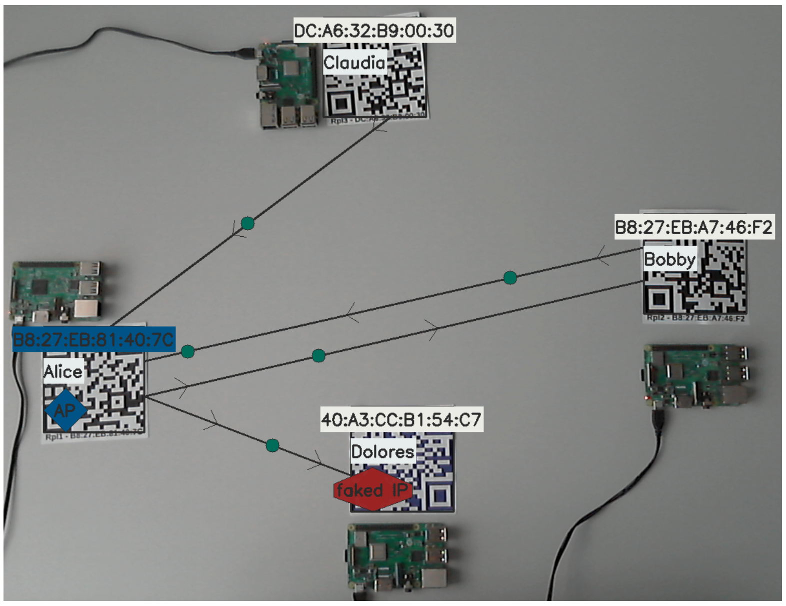


Figure 6.5: Dolores performs *ARP* poisoning and creates a mismatch between Internet Protocol (IP) and MAC addresses. As a result, Bobby takes Dolores' MAC address for Claudia's and sends packets destined for Claudia to Dolores instead. The ARMS flagged Dolores with a red symbol, because it saw Dolores send a large number of ARP "responses" which were not preceded by an ARP request.

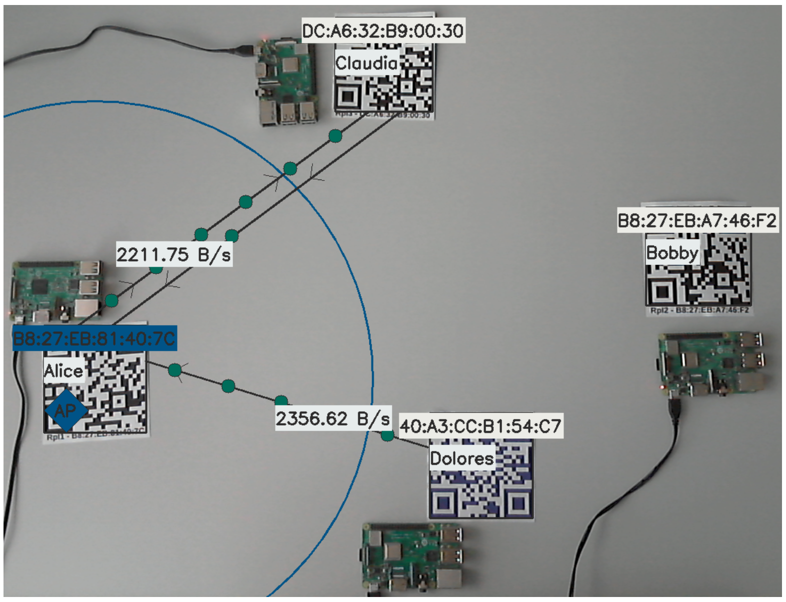


Figure 6.6: Dolores performs a *Denial of Service (DoS)* attack against Claudia by sending packets at a high data rate. Alice is busy forwarding packets from Dolores to Claudia. As a side effect, Bobby does not receive packets and assumes, its connection to the AP is lost.

Sample I recruited a total of $N = 25$ participants. 19 participants were from computer science and electrical engineering study programs (9 CS, 10 EE), who already had taken a course related to wireless network security. To test if my application is useful to a person with no background in those topics, I recruited additional six participants.

Procedure Prior to execution, the supervisor explained the test scenario, described the tasks to be performed and explained his own role as technical support. Next, participants were given a printed instruction sheet in English language. They were asked to read aloud the instruction sheet and ask any questions they may have.

Participants were provided with a switched-on laptop to which a webcam was attached. On the laptop the AR application was launched already. Initially, participants were presented the view on the wireless network with no attacks being present. They were free to move the camera to explore the network and familiarize with the AR application. They were given time to ask questions about the general operation of the application. These were noted down by the supervisor and used as additional feedback.

Participants confirmed that they were ready to begin. Next, they were asked how many devices in the network were actively sending data, which device was the AP and at which interval it transmitted advertisement frames. Additionally, participants were asked to find out the name of the advertised Wi-Fi network and the IP address of network member Claudia. Answers to those closed questions were noted by the test supervisor.

After the participant had finished accessing the network in its original state, the supervisor, being in control of Dolores, launched an attack. Participants neither knew which attack they would encounter, nor which attacks were available in the test pool. After the participant confirmed that something in the network visualization had changed, the test supervisor asked the following open questions. Participants answered the questions orally. The test supervisor noted the answers word for word and also added hints given to them.

1. Did you notice the attack? How?
2. What effects does the attack have on the network?
3. Can you explain the attack in detail?

By using this three-stage questionnaire, we were able to differentiate between proficiency of the participants and test different parameters: Question one is intended to verify that the attack has been visualized by the application in a comprehensible way. Question two investigates whether the visual representation of the attack helps in explaining its *effect* on the network. Question three explores whether the visualization of the attack helps participants in explaining the *cause* of an attack. After participants had completed all three attacks, the supervisor asked them for qualitative feedback on their experience with the application.

Each of the three authors of the original paper [120] individually performed cluster analysis on the transcribed answers. We defined three clusters: (1): the question had not been answered or a wrong answer had been given; (2): some key concepts or a basic

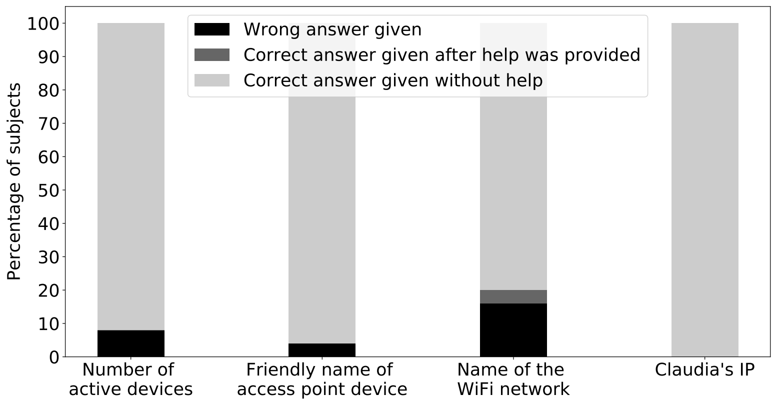


Figure 6.7: Percentage of participants who were able to gather information indicated on the x-axis by inspecting the wireless network with the AR application.

explanation were given, potentially, after the supervisor gave a hint; (3): all key concepts were mentioned or thorough answer was given without requiring a hint.

Limitations This research is subject to limitations. As this study has been planned, executed and analyzed by the same group of people they were not "blind" and therefore a bias of the study results due to the expectations of these persons cannot be excluded. Due to the small sample size data were analyzed descriptively. As future work, the number of participants should be expanded significantly to be able to draw robust statistical conclusions.

6.3.1 Experiment: Basic Use and Information Presentation

Results Figure 6.7 shows how participants dealt with the general presentation of information in the AR application. 92 % of participants were able to determine the number of active devices in the network and 96 % were able to obtain the friendly name of the AP device (Alice). 80 % of the participants were able to spot the name of the Wi-Fi network by clicking on Alice and reading the expanded information. 16 % needed a hint to click on the AP and found the requested information there, while 4 % were not able to find the name in the expanded view. All participants were able to obtain Claudia's IP address by clicking on this device.

Discussion In average 92 % of participants were able to find requested information without needing help from the supervisor. This is especially remarkable, since 24 % of our participants had no background in computer science and electrical engineering and thus could not be expected to be familiar with Wi-Fi core concepts such as addressing. Information such as the number of active devices and the friendly name of the device hosting the AP could be seen at a single glance. Additional information requested in the experiment, such as the name of the Wi-Fi network and the IP address of device

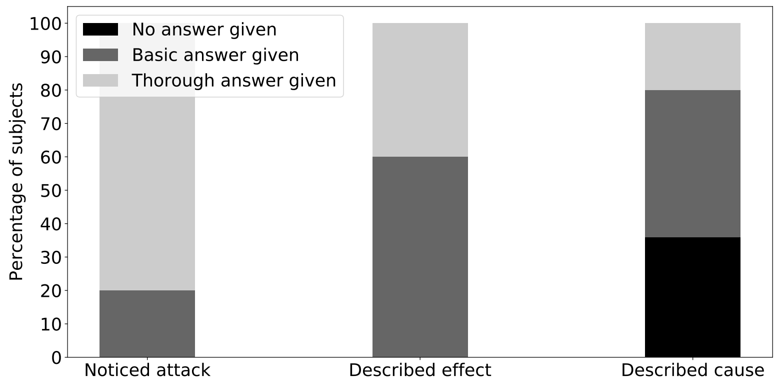


Figure 6.8: Percentage of participants who noticed the Evil Twin attack; who could explain effect and cause.

Claudia, could not be seen directly in the AR view. In those cases, participants clicked on a particular device, expecting that this would reveal additional information. Thus, I conclude that the combination of the AR view plus clickable devices is intuitive. All relevant information must be provided in those views, otherwise, information might be missed by the user.

Besides the AR view, the application also offers a network graph perspective on the network as well as a text-based packet log, similar to `wireshark`. In this study, I did not tell participants about their existence, as I wanted to evaluate how to present information in an optimal way in the AR view. Four curious participants discovered the existence of the packet log and used it to further investigate attacks. However, as they noted the similarity to `wireshark` and as my AR application is not intended to replace such text-based network inspection tools, I am going to remove the packet log in future versions of the application. I rather aim to improve the amount of information shown to the user and strive for a balance to not overwhelm the user, but nevertheless display relevant information. For example, more information could be displayed by default at the collapsed devices, e.g., their IP address or, in case of the AP, the name of the Wi-Fi network.

6.3.2 Experiment: Evil Twin

Results Figure 6.8 shows that all participants noticed the Evil Twin attack and were able to give a basic or thorough answer. Basic notice means that the de-authentication packet was noticed and that Bobby now communicates with Dolores instead of Alice. 80 % of participants noted additionally that Dolores had sent broadcast frames before.

All participants were able to explain the effect of the attack. Among those, 60 % gave a basic explanation and described that Bobby now sends data directly to Dolores, while it previously had sent packets to Claudia via Alice. 40 % gave a thorough explanation and stated additionally that Bobby had left Alice's network and joined Dolores' network instead.

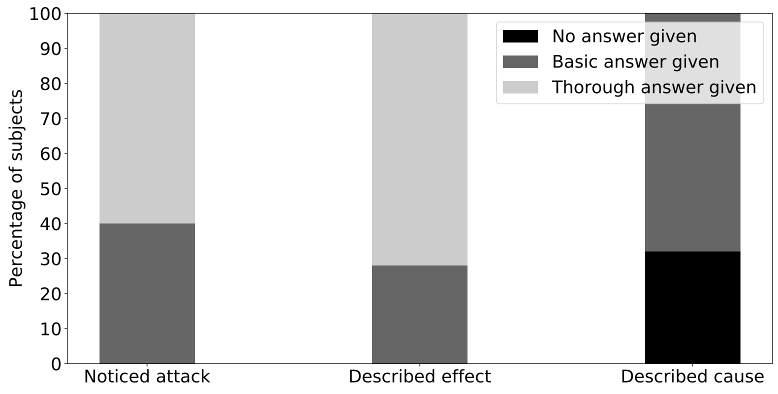


Figure 6.9: Percentage of participants who noticed the ARP poisoning attack; who could explain effect and cause.

36 % of participants could not state the cause. 44 % could explain the cause of the attack in a basic way, describing that Bobby changed its communication behavior based on a broadcast frame emitted by Dolores. The thorough explanation given by 20 % of the participants additionally included that Bobby had left Alice's network and had joined Dolores' network.

Discussion Concerning the Evil Twin attack, the AR-based visualization of the wireless network permitted all participants, even those without a professional background, to *spot* that an attack had taken place. All participants were able to describe the *effects* an attack had on the network. Hence, I conclude that the birds-eye view provided by AR is an intuitive way to visualize attacks.

Only 20 % gave a thorough description of the Evil Twin attack. To fully comprehend this attack, deep understanding of Wi-Fi networks is needed, especially the distinction between Evil Twin, where only beacons are forged, and *Impersonation* attacks, where the attacking device tries to mimic the legitimate network in all aspects. Participants who gave the correct explanation stated that they were able to do so based on their formal education or previous experience. Hence I noticed that currently our application does not provide visual clues to foster this understanding. As a consequence, additional information should be added to the AR view, for example the frequency of occurrence and the signal strength of beacons as well as the announced Service Set Identifier (SSID) of the wireless network. Such visual clues may indicate to the user that with high probability an Evil Twin attack is taking place.

6.3.3 Experiment: ARP Poisoning

Results All participants noticed the attack. The basic explanation was that packets changed their route to Dolores. For a thorough answer the testees noticed that the data stream originally going from Bobby to Claudia is now routed through Dolores.

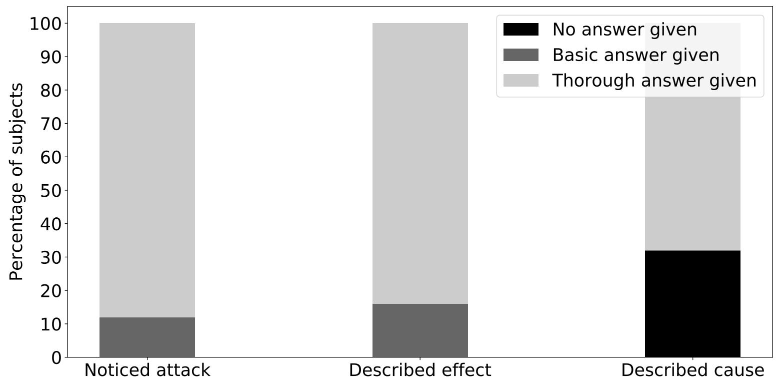


Figure 6.10: Percentage of participants who noticed the DoS attack; who could explain effect and cause.

The effect could also be explained by all participants: 28 % stated that there is a new traffic stream to Dolores, which had not been there before. 72 % additionally pointed out that the missing data to Claudia and that it seemed as if the data stream from Bobby to Claudia had been rerouted.

32 % of participants could not give an explanation for the cause of the attack. 68 % gave the basic explanation that Dolores somehow managed to fool Bobby into sending data to her instead of Claudia. No participant gave the thorough explanation that traffic was redirected by a manipulation of Bobby's ARP table.

Discussion All participants were able to notice and describe the effect of the attack, as the different traffic streams could be seen using the AR view. Explaining this attack, however, required participants to understand, how addressing in Wi-Fi-based wireless networks works and how the translation between MAC and IP addresses is done by ARP. To aid the user in understanding ARP, a view filter should be added to the application, letting users switch between IP-based and MAC-based addressing. By alternating between those views, participants could confirm that on the IP layer everything looks fine, while there has been a change in MAC-based addressing.

6.3.4 Experiment: Denial of Service

Results All participants noticed the DoS attack. 10 % only noticed that packets neither travel to, nor emanate from Bobby. 90 % additionally noticed that, unlike before the attack, a high number of packets go from Dolores to Claudia via Alice.

Concerning the effect of the attack, 16 % concluded that Bobby does not communicate anymore and thus seems to have lost its connection. 84 % additionally stated that the traffic rate between Dolores, Alice and Claudia is much higher than before.

No explanation for the cause of the attack was given by 32 % of the participants. 68 % described that the attack caused Bobby to be disconnected due to an overload of the AP.

Discussion The DoS attack was noticed and its effect could be described by all participants, as the high amount of packets and the connection loss of network participant Bobby was easily visible in the AR application. Especially in the case of DoS, the visual representation of the attack is able to precisely show its effect. While 30 % of the participants were not able to link Bobby's disconnected state to Alice being overwhelmed with packets, 70 % of the participants gave this correct explanation.

In this attack, Bobby seems to sit idle, which is not the case. Bobby is issuing *Request To Send (RTS)* packets, asking Alice for clearance to send messages, i.e., re-join requests. As Alice does not answer Bobby's RTS, Bobby assumes he is not permitted to send packets and thus does not send the request to re-join Alice's network. The application currently does not show those low-level control frames, making it seem that Bobby is not trying to reconnect to Alice' AP. This was a design choice, as in a managed Wi-Fi network every regular frame is preceded by a RTS frame and visualizing those frames would clutter the AR view too much. Adding this, however, could be beneficial in a university course on Wi-Fi internals and an option to enable or disable the visualization of those frames should be added to the application.

6.3.5 Qualitative Feedback

All participants expressed that the AR-based visualization provided them with an intuitive visualization of Wi-Fi networks and attacks on those. They stated that visualizing each packet as one moving dot is clear. However, they wished that interrelated packets are visualized as such to make cause and effect of particular packets more clear. The use of QR codes was accepted for the university course setting. For a real-world application, they wished to be able to work without those.

Using the application in the laboratory setup sparked their interest and made them eager to investigate real-world setups. Participants with no background in wireless network security stated they would like to use the application to troubleshoot their Wi-Fi network at home. They pictured the scenario in which they were able to connect their smart phone to the home router, but not their tablet computer. By seeing traffic being exchanged between the smart phone and the router problems with the router could be ruled out and, with very high confidence, they caould be narrowed down to the tablet computer.

Another proposed use case comprises investigating whether *seeing* a smart phone sending location and telemetry data has an influence on the user's notice of privacy. As packet sniffing is rarely used on smart phones, the user could further observe whether his disabling of telemetry data yields the desired effect or whether there is a discrepancy between conceived privacy settings and real device behavior.

Besides, it was suggested to make the effects of encryption and authentication visible, e.g., by letting participants observe the effects of attacks on a secure and an insecure connection.

My results show that AR is a suitable approach for wireless network security education. It provides an easy to understand and clear visualization of complex attacks, permitting participants to spot the attacks and describe their effect. Even participants with no background in engineering or computer science were able to spot attacks and give a description of its effect. The suggestion of new use cases shows, that observing a wireless network through the AR lens sparked the interest of participants.

The application is designed to hide the complexity of wireless network security from the user, permitting access to the topic with a flat learning curve. Hence, I see the AR-based approach to be used in an introductory course or alongside text-based monitoring programs. This rationale is in line with 14 participants who expressed, that the AR application is useful for quick initial analysis. For an in-depth analysis they would use tools such as `wireshark`, which displays all information, although in a less accessible manner. I conclude that the AR application is a valuable supplementary tool, providing easy access to the complex matter of wireless network security, making hard-to-grasp concepts comprehensible and sparking student's interest.

While I suggest that every student has their own handheld device in a laboratory, my approach could also be used within a wireless network security lecture. For this the lecturer could place three to four laptops or low-cost single-board computers at their desk, capture those devices with a camera attached to their presentation laptop and display the network traffic visualization using a projector. This low-effort approach provides a shared and visually appealing experience for students, which fosters their motivation to explore the AR tool in the accompanying laboratory.

6.4 Conclusion

In this chapter I investigated whether an augmented-reality based network monitoring system (ARMS) which visualizes network traffic flows and common attacks in wireless networks is beneficial in wireless network security education. To this I conducted a user study with 25 participants. 19 of those participants had received education in the fields of computer science or electrical engineering, 6 participants did not. Participants were asked to observe an 802.11 Wi-Fi (Wi-Fi) network using my augmented reality (AR) application while the test instructor launched common attacks on the network. Using the application, they were tasked to spot and explain the effects and causes of the attacks. All participants were able to spot all attacks and explain their effects on the network, while 66 % were able to explain the underlying mechanisms of the attack. Remarkably, even students with neither a background in computer science nor in electrical engineering, were able to spot the attacks and at least describe the effects on network topology. This shows that the AR-based approach is a suitable tool to be used in wireless communications security education, as it fosters the understanding of the otherwise hard-to-grasp attacks. With this I have demonstrated how to satisfy Cranor's design principle $\boxed{3}$ as it can be used to educate users. At the current state I consider ARMS to be a valuable addition to wireless network security laboratories. However, due to its easy accessibility it could also be used outside university courses.

7 Conclusion

In this thesis I demonstrated user-friendly approaches which empower non-technical users, i.e. users who have no relevant background in Computer Science (CS) or Electrical and Electronic Engineering (EEI), to set up and maintain networks of low-power wirelessly interconnected embedded systems (IES).

I reviewed important works from the field of usable security which investigate the role of the user in securing a system. The main challenge lies in the complexity of computer systems, abstract sets of rules and users not being equipped to understand those. Despite their complexity, those systems asks users to make security-critical decisions while not providing sufficient feedback to guide users in their decisions. Hence, users can be considered the weakest link in securing a system.

To account for that Cranor proposed three design goals to deal with the role of users in securing a system: $\boxed{1}$ try to completely remove the user from the security loop; $\boxed{2}$ if this is not possible, permit intuitive user interaction and $\boxed{3}$ educate users to empower them to perate complex systems securely.

With this in mind, the main conclusions from this thesis are as follows.

1. Applying Cranor's principles to networks of IES is challenging I characterized the properties of IES and highlighted the differences between IES and "Internet of Things (IoT) devices". This comparison showed that IES suffer from a lack of input/output (I/O) interfaces, both for remote and local control, and that this makes intuitive user interaction challenging.

I analyzed human factors in networks of IES by discussing user interactions in three networks I have worked on. I found that user interaction mainly occurs in the manufacturing, setup and maintenance phases of the network life cycle and that security-critical tasks could often not be automated or moved to a skilled person. Hence, removing the user from the security loop as suggested by design principle $\boxed{1}$ was hard to achieve previously. Additionally I found that due to the environment in which the IES are used and limited I/O capabilities intuitive interaction (design principle $\boxed{2}$) is challenging.

2. Auxiliary devices and mechanical workflows are beneficial In the main part of this thesis I introduced approaches to support non-technical users in setting up networks of IES. I proposed an appliance and workflow utilizing a intelligent Faraday Cage (FC) and over-the-air (OTA) firmware updates to supply a large number of IES with firmware and individual device secrets simultaneously. This approach automates security-relevant tasks and reduces user interaction to simple manual actions such as placing devices and opening and closing a box. This removes the user from the security loop, hence satisfying design principle $\boxed{1}$. My user study showed that especially non-technical users like the mechanical approach as it reminds them of manual actions they are familiar with. The appliance provided clear spoken instructions which manual action to perform, making users feel confident while interacting with the FC. Hence this shows that for non-technical users device interaction must be designed as simple as possible. Complexity must be hidden and remaining actions to be performed should be simple and match the expectations of the user of the system.

On the other hand the study showed that users with a technical background wished for *more* feedback about the internal state of the system and what is happening. Hence systems should be designed such that the amount of information and feedback provided to the user can be adjusted to suit their needs.

3. Visualization of networks is beneficial Following conclusion two, auxiliary devices can also be used to provide intuitive interaction during the maintenance phase of the network. I introduced an augmented-reality based network monitoring system (ARMS) which visualizes otherwise imperceptible wireless network traffic in an accessible way by superimposing it on the camera image of networked devices. To achieve this, the ARMS taps into the only interface available at every IES: the primary wireless interface used for communications. By passively observing network traffic relevant information about the network such as network topology, device activity status and message flows could be derived and visualized. Using the camera field of view (FOV) as intuitive filter mechanism, users can focus on parts of the network of interest to them and see only relevant information. This permits intuitive interaction and hence satisfies design principle $\boxed{2}$.

Evaluating the ARMS in a user study I found that using augmented reality (AR) permits even non-technical users to spot attacks on wireless networks. With this study, I was able to show that the learning gains found in other areas of engineering education also apply to cybersecurity education. I conclude that the user-friendly visualization provided by AR permits even non-technical users to access the complex subject of wireless network security. This satisfies design principle $\boxed{3}$. I hence recommend to use the ARMS in university courses to supplement traditional approaches of teaching wireless network security.

Bibliography

[1] Lorrie Faith Cranor and Simson Garfinkel. *Security and Usability: Designing Secure Systems that People Can Use*. O'Reilly and Associates, 1 edition, 2005.

[2] Bruce Schneier. *Click Here to Kill Everybody: Security and Survival in a Hyper-Connected World*. W. W. Norton and Company, 1st edition, 2018.

[3] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 129–142, Oakland, CA, USA, May 2008.

[4] Chunxiao Li, Anand Raghunathan, and Niraj K. Jha. Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*, pages 150–156, Columbia, MO, USA, June 2011.

[5] Eduard Marin, Dave Singelée, Flavio D. Garcia, Tom Chothia, Rik Willems, and Bart Preneel. On the (in)security of the latest generation implantable cardiac defibrillators and how to secure them. In *Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC '16*, pages 226–236, Los Angeles, California, 2016.

[6] Alexandre DHondt, Hussein Bahmad, and Jeremy Vanhee. RPL Attacks Framework. Technical report, 2015. https://rpl-attacks.readthedocs.io/en/latest/report.pdf, last viewed 2021-09-04.

[7] Paul J Wagner and Jason M Wudi. Designing and Implementing a Cyberwar Laboratory Exercise for a Computer Security Course. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, March 3-7*, pages 402–406, Norfolk, Virginia, USA, 2004.

[8] Mina Malekzadeh, Abdul Azim Abdul Ghani, and Shamala Subramaniam. Design of Cyberwar Laboratory Exercises to Implement Common Security Attacks against IEEE 802.11 Wireless Networks. In *Journal of Computer Systems, Networks, and Communications*, volume 2010, pages 1–15, 2010.

[9] Bluetooth SIG. Bluetooth 5.0 Core Specification.pdf, 2016. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043, last viewed 2021-09-04.

[10] Iulia Ion, Marc Langheinrich, Ponnurangam Kumaraguru, and Srdjan Capkun. Influence of user perception, security needs, and social factors on device pairing method choices. In *Proceedings of the Sixth Symposium on Usable Privacy and Security - SOUPS '10*, page 1, Redmond, Washington, 2010.

[11] Bruce Schneier. *Secrets and lies - digital security in a networked world: with new information about post-9/11 security*. Wiley, 2004.

[12] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. In *Proceedings of the IEEE*, volume 63, pages 1278–1308, 1975. https://www.cs.virginia.edu/~evans/cs551/saltzer/, last viewed 2021-09-04.

[13] Anne Adams and Martina Angela Sasse. Users are not the enemy. In *COMMUNICATIONS OF THE ACM*, volume 42, 1999.

[14] Eugene H Spafford, Richard A DeMillo, Andrew Bernat, Steve Crocker, David Farber, Virgil Gligor, Sy Goodman, Anita Jones, Susan Landau, Peter Neumann, David Patterson, Fred Schneider, Douglas Tygar, and William Wulf. Four Grand Challenges in Trustworthy Computing: Second in a Series of Conferences on Grand Research Challenges in Computer Science and Engineering, 2003. https://archive.cra.org/reports/trustworthy.computing.pdf, last viewed 2022-01-03.

[15] Lorrie Faith Cranor. A Framework for Reasoning About the Human in the Loop. In *Usability, Psychology, and Security Proceedings, UPSEC'08, April 14*, San Francisco, CA, USA, 2008.

[16] Zephyr Project. Zephyr Project, 2021. https://www.zephyrproject.org/, last viewed 2021-09-04.

[17] Amazon Web Services Inc. FreeRTOS Real-time operating system for microcontrollers, 2021. https://www.freertos.org/, last viewed 2021-09-04.

[18] Kim Bumsik. FreeRTOS-Shell, 2016. https://github.com/kbumsik/FreeRTOS-Shell, last viewed 2021-09-04.

[19] Zephyr Project. Zephyr 3.0.0. API Reference - Shell, 2022. https://docs.zephyrproject.org/3.0.0/reference/shell/index.html, last viewed 2022-05-01.

[20] Marco Dietz, Martin Striegel, Robert Weigel, and Amelie Hagelauer. A new heat-warning-system based on a wireless body area network for protecting firefighters in indoor operations. In *2018 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNet)*, pages 34–37, Anaheim, CA, January 2018.

[21] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An Analysis of a Large Scale Habitat Monitoring Application. In

Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04, pages 214–226, New York, NY, USA, 2004.

[22] Stephen Dawson-Haggerty, Steven Lanzisera, Jay Taneja, Richard Brown, and David E. Culler. @ scale: insights from a large, long-lived appliance energy WSN. In *The 11th International Conference on Information Processing in Sensor Networks (co-located with CPS Week 2012), IPSN 2012, Beijing, China, April 16-19, 2012*, pages 37–48, 2012.

[23] Ken Munro. Hacking / ruining the Samsung Smart Fridge, 2018. https://www.pentestpartners.com/security-blog/hacking-ruining-the-samsung-smart-fridge/, last viewd 2022-04-25.

[24] Brian Dorey. Amazon Echo Show 5 2nd Gen Smart Display Teardown, 2021. https://www.briandorey.com/post/echo-show-5-2nd-gen-smart-display-teardown, last viewed 2022-04-25.

[25] ifm. Operating instructions io-key AIK050, 2021. https://www.ifm.com/mounting/80294077UK.pdf, last viewed 2022-05-01.

[26] WEPTECH elektronik GmbH. MIOTY robust LPWAN IoT technology, 2022. https://www.weptech.de/en/technology/mioty.html, last viewed 2022-05-01.

[27] m2m Germany GmbH. LoRa Sensors, 2022. https://www.m2mgermany.de/shop/produkte/lora-sensors/, last viewed 2022-05-01.

[28] Roche Diagnostics GmbH Diabetes Care. Insight Insulin Pump, 2011. https://fccid.io/VWI1239, last viewed 2022-05-01.

[29] Dexcom Inc. Spread Spectrum Device (Dexcom G5 Continuous Glucose Monitor), 2016. https://fccid.io/PH29715, last viewed 2022-05-01.

[30] Georg Bramm, Matthias Hiller, Christian Hofmann, Stefan Hristozov, Maximilian Oppelt, Norman Pfeiffer, Martin Striegel, Matthias Struck, and Dominik Weber. CardioTEXTIL: Wearable for Monitoring and End-to-End Secure Distribution of ECGs. In *IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 2021.

[31] Mahmudur Rahman, Bogdan Carbunar, and Madhusudan Banik. Fit and Vulnerable: Attacks and Defenses for a Health Monitoring Device. In *IEEE Transactions on Mobile Computing*, volume 15, pages 447–459, February 2016.

[32] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. In *IEEE Trans. Inf. Theory*, volume 29, pages 198–207, 1983.

[33] E. Ronen, A. Shamir, A. Weingarten, and C. O'Flynn. IoT Goes Nuclear: Creating a Zigbee Chain Reaction. In *2017 IEEE Symposium on Security and Privacy, (SP) , San Jose, CA, USA, May 22-26*, pages 195–212, 2017.

[34] Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, and Yuval Elovici. dr0wned - Cyber-Physical Attack with Additive Manufacturing. In *11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, August 14-15, 2017*, page 15, 2017.

[35] Matthias Niedermaier, Jan-Ole Malchow, Florian Fischer, Daniel Marzin, Dominik Merli, and Volker Roth. You Snooze, You Lose: Measuring PLC Cycle Times under Attacks. In *12th USENIX Workshop on Offensive Technologies, WOOT 2018, Baltimore, MD, USA, August 13-14, 2018*, page 17, 2018.

[36] Florian Fischer, Matthias Niedermaier, Thomas Hanka, Peter Knauer, and Dominik Merli. Analysis of Industrial Device Architectures for Real-Time Operations Under Denial of Service Attacks. In *Information and Communications Security*, Lecture Notes in Computer Science, pages 462–478, Cham, 2020.

[37] Alma Whitten and J D Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. 1999.

[38] J Callas, L Donnerhacke, H Finney, D. Shaw, and R. Thayer. OpenPGP Message Format, 2007. https://www.ietf.org/rfc/rfc4880.txt, last viewed 2022-05-02.

[39] Deutsches Institut für Normung. *DIN EN ISO 9241-110, Ergonomie der Mensch-System-Interaktion. Teil 110, Interaktionsprinzipien.* Beuth Verlag GmbH, Frankfurt am Main, 2006.

[40] International Organization for Standardization. *ISO/IEC 2382:2015(en), Information technology — Vocabulary.* ISO/IEC, 2015. https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en, last viewed 2021-09-04.

[41] Department of Homeland Security. A Roadmap for Cybersecurity Research. Technical report, 2009. https://www.dhs.gov/sites/default/files/publications/CSD-DHS-Cybersecurity-Roadmap_0.pdf, last viewed 2021-09-04.

[42] Adam Beautement, M Angela Sasse, and Mike Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 Workshop on New Security Paradigms, Lake Tahoe, CA, USA, September 22-25, 2008*, pages 47–58, 2008.

[43] Bryan D. Payne and W. Keith Edwards. A Brief Introduction to Usable Security. In *IEEE Internet Computing*, volume 12, pages 13–21, 2008.

[44] Deanna D. Caputo, Shari Lawrence Pfleeger, M. Angela Sasse, Paul Ammann, Jeff Offutt, and Lin Deng. Barriers to Usable Security? Three Organizational Case Studies. In *IEEE Security & Privacy*, volume 14, pages 22–32, September 2016.

[45] M. Angela Sasse, Matthew Smith, Cormac Herley, Heather Lipford, and Kami Vaniea. Debunking Security-Usability Tradeoff Myths. *IEEE Security & Privacy*, 14(5):33–39, 2016.

[46] Donald A. Norman. Design rules based on analyses of human error. *Communications of the ACM*, 26(4):254–258, April 1983.

[47] Bruce Schneier. A Plea for Simplicity. You can't secure what you don't understand., 1999. https://www.schneier.com/essays/archives/1999/11/a_plea_for_simplicit.html, last viewed 2022-06-08.

[48] Amit Sahai and Brent Waters. Fuzzy Identity Based Encryption. *IACR Cryptol. ePrint Arch.*, 2004. http://eprint.iacr.org/2004/086, last viewed 2022-01-03.

[49] European Telecommunications Standards Institute. ETSI TS 103 357 Short Range Devices; Low Throughput Networks (LTN); Protocols for radio interface A V 1.1.1, 2018.

[50] NXP Semiconductor. SL3S4011_4021 UCODE I2C Product data sheet Rev. 3.5, 2018. https://datasheetspdf.com/pdf-file/1444052/NXP/SL3S4021FHK/1, last viewed 2021-09-04.

[51] Frank Stajano and Ross J. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols, 7th International Workshop, Cambridge, UK, April 19-21, 1999, Proceedings*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–194. Springer, 1999.

[52] Martin Striegel, Johann Heyszl, Florian Jakobsmeier, Yacov Matveev, and Georg Sigl. Secure and user-friendly over-the-air firmware distribution in a portable faraday cage. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '20, July 2020.

[53] International Electrotechnical Commission. *IEC 62443-1-1 Industrial communication networks - Network and system security - Part-1-1 - Terminology concepts and models.pdf*. Number 62443-1-1 in International standard / IEC. Geneva, edition 1.0 edition, 2009.

[54] European Commission. Directive 2014/53/EU of the European Parliament and of the Council of 16 April 2014 on the harmonisation of the laws of the Member States relating to the making available on the market of radio equipment and repealing Directive 1999/5/EC, April 2014. http://data.europa.eu/eli/dir/2014/53/oj/eng, last viewed 2022-06-10.

[55] European Commission. COMMISSION DELEGATED REGULATION (EU) of 29.10.2021 supplementing Directive 2014/53/EU of the European Parliament and of the Council with regard to the application of the essential requirements

referred to in Article 3(3), points (d), (e) and (f), of that Directive, 2021. https://ec.europa.eu/growth/system/files/2021-10/C_2021_7672_F1_COMMISSION_DELEGATED_REGULATION_EN_V10_P1_1428769.PDF, last viewed 2022-06-10.

[56] International Electrotechnical Commission. *Security for industrial automation and control systems – Part 4-2: Technical security requirements for IACS components*. Number 62442-4-2 in International standard / IEC. IEC Central Office, Geneva, edition 1.0 edition, 2019.

[57] Dirk Balfanz, D K Smetters, Paul Stewart, and H Chi Wong. Talking To Strangers: Authentication in Ad-Hoc Wireless Networks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2002, San Diego, California, USA*, 2002.

[58] Colin Swindells, Kori M Inkpen, John C Dill, and Melanie Tory. That one there! Pointing to establish device identity. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, pages 151–160, Paris, France, October 2002.

[59] Jaap-Henk Hoepman. The Ephemeral Pairing Problem. In *Financial Cryptography, 8th International Conference, FC 2004, Key West, FL, USA, February 9-12, 2004. Revised Papers*, volume 3110 of *Lecture Notes in Computer Science*, pages 212–226. Springer, 2004.

[60] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Secure Device Pairing based on a Visual Channel (Short Paper). In *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May*, pages 306–313, Berkeley, California, USA, 2006. IEEE Computer Society.

[61] Nitesh Saxena and Md. Borhan Uddin. Blink 'Em All: Scalable, User-Friendly and Secure Initialization of Wireless Sensor Nodes. In *Cryptology and Network Security*, volume 5888, pages 154–173. Springer Berlin Heidelberg, 2009. Series Title: Lecture Notes in Computer Science.

[62] Matthias Gauger, Olga Saukh, and Pedro Jose Marron. Enlighten me! secure key assignment in wireless sensor networks. In *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pages 246–255, Macau, China, October 2009.

[63] Dirk Balfanz, Glenn Durfee, Rebecca E Grinter, D K Smetters, and Paul Stewart. Network-in-a-Box: How to Set Up a Secure Wireless Network in Under a Minute. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, 2004.

[64] C.V. Lopes and P.M.Q. Aguiar. Aerial acoustic communications. In *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, pages 219–222, New Platz, NY, USA, 2001.

[65] M.T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 10–10, Lisboa, Portugal, 2006.

[66] Claudio Soriente, Gene Tsudik, and Ersin Uzun. HAPADEP: Human-Assisted Pure Audio Device Pairing. In *Information Security*, Lecture Notes in Computer Science, pages 385–400. Springer Berlin Heidelberg, 2008.

[67] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Ubicomp 2001: Ubiquitous Computing*, volume 2201, pages 116–122. Springer Berlin Heidelberg, 2001. Series Title: Lecture Notes in Computer Science.

[68] Jonathan Lester, Blake Hannaford, and Gaetano Borriello. "Are You with Me?" – Using Accelerometers to Determine If Two Devices Are Carried by the Same Person. In *Pervasive Computing*, volume 3001, pages 33–50, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

[69] Claude Castelluccia and Pars Mutaf. Shake them up!: a movement-based pairing protocol for CPU-constrained devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05*, page 51, Seattle, Washington, 2005.

[70] Rene Mayrhofer and Hans Gellersen. Shake Well Before Use: Authentication Based on Accelerometer Data. In *Pervasive Computing*, volume 4480, pages 144–161. Springer Berlin Heidelberg, 2007. Series Title: Lecture Notes in Computer Science.

[71] M. Cagalj, S. Capkun, and J.-P. Hubaux. Key Agreement in Peer-to-Peer Wireless Networks. In *Proceedings of the IEEE*, volume 94, pages 467–478, February 2006.

[72] Ramnath Prasad and Nitesh Saxena. Efficient Device Pairing Using "Human-Comparable" Synchronized Audiovisual Patterns. In *Applied Cryptography and Network Security*, volume 5037, pages 328–345. Springer Berlin Heidelberg, 2008. Series Title: Lecture Notes in Computer Science.

[73] Nitesh Saxena and Md. Borhan Uddin. Automated Device Pairing for Asymmetric Pairing Scenarios. In *Information and Communications Security*, volume 5308, pages 311–327. Springer Berlin Heidelberg, 2008. Series Title: Lecture Notes in Computer Science.

[74] Ming Li, Shucheng Yu, Wenjing Lou, and Kui Ren. Group Device Pairing based Secure Sensor Association and Key Management for Body Area Networks. In

2010 Proceedings IEEE INFOCOM, pages 1–9, San Diego, CA, USA, March 2010. IEEE.

[75] T. Perkovic, M. Cagalj, T. Mastelic, N. Saxena, and D. Begusic. Secure Initialization of Multiple Constrained Wireless Devices for an Unaided User. In *IEEE Transactions on Mobile Computing*, volume 11, pages 337–351, February 2012.

[76] Cynthia Kuo, Mark Luk, Rohit Negi, and Adrian Perrig. Message-in-a-bottle: user-friendly and secure key deployment for sensor nodes. In *Proceedings of the 5th international conference on Embedded networked sensor systems - SenSys '07*, page 233, Sydney, Australia, 2007. ACM Press.

[77] Yee Wei Law, Giorgi Moniava, Zheng Gong, Pieter Hartel, and Marimuthu Palaniswami. KALwEN: a new practical and interoperable key management scheme for body sensor networks: KALwEN:. In *Security and Communication Networks*, volume 4, pages 1309–1329, November 2011.

[78] Linaro Limited. MCU Boot - Secure boot for 32-bit Microcontrollers, 2022. https://www.mcuboot.com/, last viewed 2022-06-04.

[79] MBED OS. Firmware Over the Air (FOTA) Updates, 2019. https://os.mbed.com/teams/Bluetooth-Low-Energy/wiki/Firmware-Over-the-Air-FOTA-Updates, last viewed 2021-09-04.

[80] Espressif Systems Inc. Over The Air Updates (OTA), 2021. https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/ota.html, last viewed 2021-09-04.

[81] Johan Stokking. Firmware Updates over Low-Power Wide Area Networks, 2019. https://www.thethingsnetwork.org/article/firmware-updates-over-low-power-wide-area-networks, last viewed 2022-01-03.

[82] NXP. ZigBee Cluster Library (for ZigBee 3.0) User Guide, 2018. https://www.nxp.com/docs/en/user-guide/JN-UG-3115.pdf, last viewed 2021-09-04.

[83] Mark Solters. OTA for Contiki (CC2650 SoC), 2016. http://marksolters.com/programming/2016/06/07/contiki-ota.html, last viewed 2021-09-04.

[84] A. Bruce Carlson and Paul B. Crilly. *Communication systems: an introduction to signals and noise in electrical communication.* McGraw-Hill Higher Education, Boston, 5th ed edition, 2010.

[85] Espressif Systems Inc. ESP32 Datasheet Version 2.1, 2018.

[86] Atheros Communications Inc. AR9271 Single-Chip Data Sheet, 2011. https://datasheetspdf.com/pdf-file/788267/Atheros/AR9271/1, last viewed 2021-09-04.

[87] Daniele Perito and Gene Tsudik. Secure Code Update for Embedded Devices via Proofs of Secure Erasure. In *Computer Security – ESORICS 2010*, volume 6345, pages 643–662, 2010.

[88] Faraday Cases. Product Sheet Zeus Injection Molded Cases, 2021. https://faradaycases.com/wp-content/uploads/2021/10/Faraday_InjectionMolded.pdf, last accessed 2021-12-28.

[89] John Brooke. SUS - A quick and dirty usability scale. In Patrick W. Jordan, B Thomas, Ian Lyall McClelland, and Bernard Weerdmeester, editors, *Usability Evaluation In Industry*. CRC Press, 1 edition, 1986.

[90] Martin Striegel, Carsten Rolfes, Johann Heyszl, Fabian Helfert, Maximilian Hornung, and Georg Sigl. EyeSec: A Retrofittable Augmented Reality Tool for Troubleshooting Wireless Sensor Networks in the Field. In *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks (EWSN '19)*, 2019.

[91] Carsten Buschmann, Dennis Pfisterer, Stefan Fischer, Sándor P. Fekete, and Alexander Kröller. SpyGlass: A Wireless Sensor Network Visualizer. In *SIGBED Rev.*, volume 2, pages 1–6, January 2005.

[92] L. Shu, C. Wu, Y. Zhang, J. Chen, L. Wang, and M. Hauswirth. NetTopo: Beyond Simulator and Visualizer for Wireless Sensor Networks. In *2008 Second International Conference on Future Generation Communication and Networking*, volume 1, pages 17–20, December 2008.

[93] M. Turon. MOTE-VIEW: a sensor network monitoring and management tool. In *The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II.*, pages 11–17, Sydney, Queensland, Australia, 2005.

[94] N. D. Bokde, P. D. Peshwe, A. Gupta, and K. D. Kulat. RemoteWSN: A novel technique for remotely visualizing connectivity in WSN working on a weight based routing algorithm. In *2015 National Conference on Recent Advances in Electronics Computer Engineering (RAECE)*, pages 92–95, February 2015.

[95] M. Ringwald and A. Vitaletti. SNIF: Sensor Network Inspection Framework. In *Technical Report No. 535, Department of Computer Science*, pages 179–192, 2006.

[96] M. Ringwald and K. Romer. Deployment of Sensor Networks: Problems and Passive Inspection. In *2007 Fifth Workshop on Intelligent Solutions in Embedded Systems*, pages 179–192, June 2007.

[97] Y. Yang, P. Xia, L. Huang, Q. Zhou, Y. Xu, and X. Li. SNAMP: A Multi-sniffer and Multi-view Visualization Platform for Wireless Sensor Networks. In *2006 1ST IEEE Conference on Industrial Electronics and Applications*, pages 1–4, May 2006.

[98] The Tcpdump Group. LIBPCAP, 2021. https://github.com/the-tcpdump-group/libpcap, last viewed 2021-12-28.

[99] Wireshark Foundation. Wireshark Network Protocol Analyzer, 2021. https://www.wireshark.org/, last viewed 2021-09-04.

[100] OpenCV team. Open Source Computer Vision Library (OpenCV), 2021. https://opencv.org/, last viewed 2021-09-04.

[101] T. Haramaki and H. Nishino. A Device Identification Method for AR-based Network Topology Visualization. In *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, pages 255–262, November 2015.

[102] Kenya Sato, Naoya Sakamoto, Shinya Mihara, and Hideki Shimada. CyPhy-UI: Cyber-Physical User Interaction Paradigm to Control Networked Appliances with Augmented Reality. *The Sixth International Conference on Advances in Computer-Human Interactions (ACHI)*, 2013.

[103] Anthea Mayzaud, Remi Badonnel, and Isabelle Chrisment. A Taxonomy of Attacks in RPL-based Internet of Things. In *International Journal of Network Security*, volume 18, pages 459–473, 2016.

[104] Linus Wallgren, Shahid Raza, and Thiemo Voigt. Routing Attacks and Countermeasures in the RPL-Based Internet of Things. In *International Journal of Distributed Sensor Networks, 2013*, volume 9, 2013.

[105] KimiNewt. pyshark, 2021. https://github.com/KimiNewt/pyshark, last viewed 2021-09-04.

[106] PostgreSQL Global Development Group. PostgreSQL, 2021. https://www.postgresql.org/, last viewed 2021-09-04.

[107] SQLAlchemy. The Python SQL Toolkit and Object Relational Mapper., 2021. https://www.sqlalchemy.org/, last viewed 2021-09-04.

[108] ZXing Project. ZXing ("Zebra Crossing") barcode scanning library for Java, Android, 2021. https://github.com/zxing/zxing, last viewed 2021-09-04.

[109] IETF Network Working Group. *RFC6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks.* 2012.

[110] Contiki-NG: The OS for Next Generation IoT Devices, 2021. https://github.com/contiki-ng/contiki-ng, last viewed 2021-09-04.

[111] Matthias Ringwald, Kay Römer, and Andrea Vitaletti. Passive Inspection of Sensor Networks. In *Distributed Computing in Sensor Systems*, pages 205–222, 2007.

[112] Malinda Rauhala, Ann-Sofie Gunnarsson, and Anders Henrysson. A Novel Interface to Sensor Networks Using Handheld Augmented Reality. In *Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '06, pages 145–148, 2006.

[113] D. Goldsmith, F. Liarokapis, G. Malone, and J. Kemp. Augmented Reality Environmental Monitoring Using Wireless Sensor Networks. In *2008 12th International Conference Information Visualisation*, pages 539–544, July 2008.

[114] Kenya Sato, Naoya Sakamoto, and Hideki Shimada. Visualization and Management Platform with Augmented Reality for Wireless Sensor Networks. *Wireless Sensor Network*, 2015.

[115] Naoya Sakamoto, Hideki Shimada, and Kenya Sato. Design and Implementation of Sensor Network Device Control System with AR Technology. In *Mechatronics and Information Technology*, volume 2 of *Advanced Engineering Forum*, pages 131–134. Trans Tech Publications, 2011.

[116] Cem Sahin, Danh Nguyen, Simon Begashaw, Brandon Katz, James Chacko, Logan Henderson, Jennifer Stanford, and Kapil R. Dandekar. Wireless communications engineering education via Augmented Reality. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–7, 2016.

[117] T. Ohta, R. Ito, and Y. Kakuda. Design of a Node Status Visualizing Software Utilizing the AR Technology for Multihop Wireless Networks. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 270–271, July 2017.

[118] G. Koutitas, J. Jabez, C. Grohman, C. Radhakrishna, V. Siddaraju, and S. Jadon. Demo/poster abstract: XReality research lab - Augmented reality meets Internet of Things. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2, April 2018.

[119] George Koutitas. Presentation: Augmented Reality meets Internet of Things, 2018. https://www.iaria.org/conferences2018/filesMMEDIA18/GeorgeKoutitas_IARIA_Tutorial_AR_IoT_share.pdf, last viewed 2021-09-04.

[120] Martin Striegel, Jonas Erasmus, and Parag Jain. Evaluating Augmented Reality for Wireless Network Security Education. In *Proceedings of the 2021 IEEE Frontiers in Education Conference (FIE)*, Lincoln, Nebraska, 2021.

[121] M.B. McGrath and J.R. Brown. Visual Learning for Science and Engineering. In *IEEE Computer Graphics and Applications*, volume 25, pages 56–63, September 2005.

[122] Mark Billinghurst and Andreas Duenser. Augmented Reality in the Classroom. In *IEEE Computer*, volume 45, pages 56–63, 2012.

[123] María Blanca Ibánez, Ángela Di Serio, Diego Villarán, and Carlos Delgado Kloos. Experimenting with electromagnetism using augmented reality: Impact on flow student experience and educational effectiveness. In *Computers & Education*, volume 71, pages 1–13, February 2014.

[124] Danh H. Nguyen, Logan Henderson, James Chacko, Cem Sahin, Anton Paatelma, Harri Saarnisaari, Nagarajan Kandasamy, and Kapil R. Dandekar. BeamViewer: Visualization of dynamic antenna radiation patterns using Augmented Reality. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 794–795, April 2016.

[125] Mikel Salazar, José Gaviria, Carlos Laorden, and Pablo G. Bringas. Enhancing cybersecurity learning through an augmented reality-based serious game. In *2013 IEEE Global Engineering Education Conference (EDUCON)*, pages 602–607, 2013.

[126] Cem Sahin, Danh Nguyen, James Chacko, and Kapil R. Dandekar. Wireless cybersecurity education via a software defined radio laboratory. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, October 2015.

[127] George P. Tadda and John S. Salerno. Overview of Cyber Situation Awareness. In *Cyber Situational Awareness*, volume 46, pages 15–35. 2010.

[128] LoRA Alliance. LoRaWAN Specification v1.1, 2021. https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/, last viewed 2021-09-04.

[129] The Things Industries. Adding Devices | The Things Stack for LoRaWAN, 2022. https://www.thethingsindustries.com/docs/devices/adding-devices/, last viewed 2022-06-08.

[130] WiFi Alliance. Wi-Fi Easy Connect Specification Version 2.0, 2020. https://www.wi-fi.org/downloads-public/Wi-Fi_Easy_Connect_Specification_v2.0.pdf/35330, last viewed 2022-08-06.

[131] Martin Striegel, Bodo Selmke, and Katja Miller. Verfahren und Vorrichtung zum Bereitstellen zumindest eines kryptografischen Schlüssels für zumindest ein Mobilgerät, October 2019. Europäisches Patent EP3557897A1.

A Appendix

A.1 Examples of Provisioning in Common Wireless Protocols

In this section, I briefly introduce the provision process of the common wireless transmission protocols LoRaWAN, Bluetooth Low Energy (BLE) and 802.11 Wi-Fi (Wi-Fi). It can be seen, that all of those protocols require the manual distribution of information, which must be confidential and/or authentic, between devices. This means, that for those protocols, the user is part of the security loop.

LoRaWAN In a LoRaWAN-based communication system there are four entities: End-devices (which are low-power wirelessly interconnected embedded systems (IES)), gateways, a network server and an application server. The user owns the end-devices and operates an application at which data are received, stored and displayed and which is hosted on the application server. LoRaWAN uses symmetric encryption and message authentication based on Advanced Encryption Standard (AES)-128 to provide end to end security from the end-node to the application [128].

To assign their end-devices to their application, a 128 bit *AppKey* and a 128 bit *NwkKey* must be created and made known to the LoRaWAN-enabled IES and the LoRaWAN application. Additionally, every LoRaWAN-enabled device has a unique 64 bit Device Extended Unique Identifier (EUI) (DevEUI). To assign the device to the application, the user must make this DevEUI known to the application server.

The LoRaWAN specification does not define the creation, provisioning, or management of keys during life time. In some applications, the AppKey and NwkKey were created by the device manufacturer and are stored as a variable in the source code of the IES and thus compiled into the firmware image [129]. In this case, the user must read the keys as well as the DevEUI from the end-device and insert them into his application. Many LoRaWAN-enabled devices utilize an app to retrieve those information from anNear-Field Communications (NFC) tag or Quick Response (QR)-code attached to the device. The app might be connected to the application of the user and forward those information to the application.

If no keys were written onto the end-device by the manufacturer, the user can create the keys in their application. The user must then transfer those to the end-device via an unspecified mechanism. In practice, most LoRaWAN end-devices use NFC and a smart phone app for doing this.

BLE BLE with the *Secure Connections* suite of cryptographic algorithms uses symmetric AES-based encryption and keyed-Hash Message Authentication Code (HMAC)-

Secure Hash Algorithm (SHA)-256 message authentication on the link layer[1] [9]. The keys used in those algorithms are generated using P-256 Elliptic-Curve Diffie-Hellman (ECDH) in the provisioning phase, which Bluetooth calls *pairing*. Pairing occurs between a Bluetooth *peripheral* device, e.g., a IES, and a Bluetooth *central*, e.g., a smart phone or a single board computer (SBC). One central device can be connected to multiple peripheral devices. For each connection, an individual symmetric key is derived from the ECDH key exchange. Depending on the input/output (I/O) capabilities of peripheral and central, this key exchange occurs unauthenticated (*Just Works* association model), or authenticated: In *Passkey Entry* association model the user enters the same 6-digit number in both the peripheral and the central. With the *out of band (OOB)* association model, the public key of the peripheral is transferred to the central over an authentic location-limited channel (LLC). For this, in practice typically NFC is used, as it is present in most modern smart phones. Lastly, with *Numeric Comparison* association model both central and peripheral device calculate confirmation values based on the results of the ECDH key exchange and present those to the user, who confirms equality.

Wi-Fi Wi-Fi networks are typically protected with *Wi-Fi Protected Access version 3 (WPA3)* in *Personal* or *Enterprise* mode [130]. In the former, to provision a device and add it to a network, the passphrase of the network must be entered in the device, from which a symmetric 256,bit encryption key is derived. In the latter, a *Remote Authentication Dial-In User Service (RADIUS)* authentication server is used which verifies, if a user is permitted to join the network. As IES are typically not tied to enterprise user accounts, RADIUS authentication is not encountered in networks of IES.

WPA3 added a feature named *Easy Connect*, with which devices without a display (such as IES) can be provisioned. A smart phone reads a QR code or an NFC tag attached to the IES, or listens to BLE packets sent at low transmission power by the IES. All of those channels can be considered authentic LLC. In all cases, by doing one of the above, the smart phone learns the public key of the IES. The smart phone is connected to the wireless network, to which the IES shall connect to and thus there is a secure channel between the smart phone and the network. For every device scanned via QR code or NFC, the smart phone forwards the public key to the network.

[1]Secure Connections is the only suite of cryptographic algorithms which is recommended in practice, as the other suites such as *LE Legacy Pairing* contain algorithms considered outdated.

A.2 List of Publications

1. Marco Dietz, Martin Striegel, Robert Weigel, and Amelie Hagelauer. A new heat-warning-system based on a wireless body area network for protecting firefighters in indoor operations. In *2018 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNet)*, pages 34–37, Anaheim, CA, January 2018

2. Martin Striegel, Carsten Rolfes, Johann Heyszl, Fabian Helfert, Maximilian Hornung, and Georg Sigl. EyeSec: A Retrofittable Augmented Reality Tool for Troubleshooting Wireless Sensor Networks in the Field. In *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks (EWSN '19)*, 2019

3. Martin Striegel, Johann Heyszl, Florian Jakobsmeier, Yacov Matveev, and Georg Sigl. Secure and user-friendly over-the-air firmware distribution in a portable faraday cage. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '20, July 2020

4. Martin Striegel, Jonas Erasmus, and Parag Jain. Evaluating Augmented Reality for Wireless Network Security Education. In *Proceedings of the 2021 IEEE Frontiers in Education Conference (FIE)*, Lincoln, Nebraska, 2021

5. Georg Bramm, Matthias Hiller, Christian Hofmann, Stefan Hristozov, Maximilian Oppelt, Norman Pfeiffer, Martin Striegel, Matthias Struck, and Dominik Weber. CardioTEXTIL: Wearable for Monitoring and End-to-End Secure Distribution of ECGs. In *IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 2021

6. Martin Striegel, Bodo Selmke, and Katja Miller. Verfahren und Vorrichtung zum Bereitstellen zumindest eines kryptografischen Schlüssels für zumindest ein Mobilgerät, October 2019. Europäisches Patent EP3557897A1