Dipl.-Ing. Dipl.-Ing. Thomas Ulz, BSc.

Towards Trustworthy Smart Sensors

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Inform. Dr.sc.ETH Kay Römer

Institute of Technical Informatics

Advisor
Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Graz, May 2019

AFFIDAVIT

Acknowledgements

During my PhD studies I was fortunate to meet many great people with whom I could discuss ideas and collaborate on various topics. I am very thankful to all of these people. Although I think it is not possible to adequately thank all of them in this place, I still want to express my deepest appreciation to some persons that supported me during this my PhD studies.

This thesis has been carried out at the Institute of Technical Informatics at Graz University of Technology in close cooperation with our industrial partner Infineon Technologies Austria AG. Therefore, I want to first and foremost extend my deepest gratitude to my supervisor Prof. Kay Römer for guiding me through my PhD studies via discussions, helpful feedback, and in organizational matters. I am also extremely grateful to my mentor Dr. Christian Steger who supported me in everyday topics such as publications, conferences, and technical discussions. Additionally I want to sincerely thank Prof. Thorsten Strufe for agreeing to serve as a second adviser for this thesis. I also want to thank our industrial partner as well as the involved funding agencies for facilitating my PhD studies[1].

Furthermore, I want to extend my sincere thanks to colleges that sometimes had to steer me into the right direction during technical discussions. Thank you Thomas Pieber, Holger Bock, Sarah Haas, Rainer Matischek, Andrea Höller, and Andreas Wallner. I also want to thank all students who contributed to this thesis for their great work. Thank you Markus Feldbacher, Martin Zöhrer, Benjamin Bara, Benjamin Mößlang, Fikret Basic, and Niko Mittendrein.

I also would like to extend special thanks to people at the Institute of Technical Informatics that cheered me up and constantly challenged and motivated me during my PhD studies. This very special group of people includes Michael Spörk, Rainer Hoffmann, Martin Erb, and Michael Krisper. Thank you for many evenings that cleared my mind from technical issues that otherwise would have given me sleepless nights.

Finally, I want to thank the most important people without whom my studies would not haven been possible and who supported me during my entire PhD studies, my friends and family. These people constantly motivated and encouraged me while also tolerating the limited time I could spend with them during time consuming phases of my PhD studies. My heartfelt thanks go to Jürgen, Philipp, Jakob, Vanessa, Marco, Simon, Rene, Manuel, Verena, and Christian as well as to my family Renate, Johann, Franziska, Josefa, Madlen, and Jade.

Thank you to all of you for such a great and memorable yet challenging time!

Graz, May 2019
Thomas Ulz

[1] This work has been performed within the IoSense (http://iosense.eu) project. This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia. IoSense is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2016 and April 2019. More information: https://iktderzukunft.at/en/.

Abstract

Sensors are embedded in a wide range of devices and systems nowadays, be it smartphones, Cyber-Physical Systems, or devices in the Internet of Things. Additionally, the number of so-called smart sensors that are equipped with a computational device and a networking interface is rapidly rising. Although these devices might not be as powerful in terms of computational power, the large number of devices and the consistent availability of these devices makes them an attractive target for numerous types of attacks. Many recent incidents have demonstrated that a large number of resource-constrained devices can be used in attacks that target larger systems. Default configurations or standard authentication credentials are often the main weakness that allows such attacks. Also, other weaknesses such as unprotected sensor interfaces may lead to issues where a system is compromised by attacks targeting an embedded sensor.

Thus, the first part of this thesis investigates the question whether there are yet unknown types of attacks that target sensor interfaces. The first interface, the sensing interface itself, is often unprotected since sensor measurements are often not considered as confidential information. However, in systems where a process is controlled based on these sensor measurements, the confidentiality of sensor measurements is important. We demonstrate that only protecting the confidentiality of sensor data leads to easy attacks that may cause dangerous system behavior. In this thesis, we also investigate the impact of unprotected sensor interfaces to bypass barriers such as process isolation. We successfully demonstrate exploiting sensor configuration interfaces as well as the sensing interface itself for building sensor-based covert channels. We present three different approaches that differ in the achievable covert-channel data rate as well as in the likelihood of such a covert channel being detected. We believe that the behavior of the stealthiest of these covert channels is not distinguishable from normal system operation where multiple processes access a single sensor. All presented covert channels are applicable for any system, but in particular, we were able to demonstrate them on the well-known operating systems Linux and Android.

Due to the demonstrated security issues caused by unprotected configuration interfaces, we demonstrate a secured configuration approach for smart sensors based on Near-Field Communication (NFC) in the second part of this thesis. On the one hand, this secured NFC-based configuration interface is capable of improving the security of smart sensors by mitigating most attacks that could target the interface. On the other hand, we think that using this well-known technology will also improve the perceived usability when changing configuration data. However, the proposed configuration approach is not only applicable for end users but can also be used for applying initial configurations during the manufacturing of smart sensors. The presented approach includes a tamper-resistant hardware extension for smart sensors as well as software components that provide secured transfer of configuration data, attestation of applied configuration data, and the password-free authentication of devices when applying configuration data. Furthermore, this password-free approach automatically derives authentication credentials from configuration data and thus helps mitigating issues caused by users not changing default credentials. Finally, we present an NFC transport-layer protocol that provides secured data transfer for NFC applications, thus mitigating the need for application-specific security mechanisms and instead providing a standardized method for secured NFC-based data transfer. This so-called QSNFC protocol provides security as well as efficiency by fulfilling the zero-round-trip-time requirement for recurring connections.

Kurzfassung

Eingebettete Sensoren finden sich heute in einer Vielzahl verschiedener Geräte und Systeme, ob in Mobiltelefonen, Cyber-physischen Systemen oder Geräten im Internet der Dinge. Zusätzlich dazu steigt auch die Anzahl sogenannter smarter Sensoren, welche mit einer Recheneinheit und Netzwerkfähigkeit ausgestattet sind. Obwohl all diese Geräte leistungsschwach sind und keine starke Rechenleistung besitzen, sind sie durch ihre große Anzahl und die ständige Verfügbarkeit interessante Ziele für viele Arten von Angriffen. Wie jüngste Angriffe gezeigt haben, können diese leistungsschwachen Geräte benutzt werden, um Angriffe auf größere Systeme auszuführen. Sehr oft ermöglichen unveränderte Standardzugangsdaten oder –konfigurationen solche Angriffe. Aber auch Sicherheitslücken wie zum Beispiel ungesicherte Sensorschnittstellen können Probleme verursachen, welche dazu führen, dass Systeme kompromitiert werden können.

Der erste Teil dieser Dissertation behandelt daher die Frage, ob es bisher unbekannte Arten von Attacken auf Sensorschnittstellen gibt. Die Messschnittstelle des Sensors ist meist nicht geschützt, da Sensormesswerte vielfach nicht als vertrauliche Informationen betrachtet werden. Allerdings sollten diese Informationen in Systemen, in denen Prozesse aufgrund dieser Sensormesswerte geregelt werden, als vertrauchlich behandelt werden. Diese Arbeit zeigt, dass selbst Systeme, welche die Vertraulichkeit von Sensormesswerten sicherstellen, einfach attackiert werden können. Außerdem behandelt diese Dissertation die Auswirkung von ungesicherten Sensorschnittstellen, welche ausgenutzt werden, um Prozessisolationen zu umgehen. In dieser Arbeit werden mehrere sensorbasierte Datenkanäle gezeigt, welche aufbauend auf unterschiedlichen Sensorschnittstellen realisiert wurden. Die gezeigten Varianten unterscheiden sich in der erzielbaren Datenrate, aber auch in der Wahrscheinlichkeit, dass solche Datenkanäle entdeckt werden. Die unsichtbarste Variante dieser Datenkanäle ist nicht vom Verhalten eines Systems mit einem Sensor und mehreren Prozessen unterscheidbar. Alle in dieser Arbeit gezeigten Datenkanäle können auf beliebige Systeme angewendet werden. In dieser Dissertation werden sie an den sehr bekannten Betriebssystemen Linux und Android demonstriert.

Basierend auf den gezeigten Schwachstellen, die auf ungesicherten Konfigurationsschnittstellen basieren, wird im zweiten Teil dieser Dissertation eine NFC-basierte Methode zur gesicherten Konfiguration von smarten Sensoren präsentiert. Zum einen kann so eine Methode die Sicherheit von Systemen verbessern, zum anderen wird in dieser Arbeit die Meinung vertreten, dass die Benutzerfreundlichkeit eines Konfigurationsprozesses damit gesteigert werden kann. Der gezeigte Ansatz ist nicht nur geeignet, um Konfigurationen beim Endbenutzer durchzuführen, sondern ist bereits während der Produktion des Gerätes, in der Standardkonfigurationen übertragen werden müssen, einsetzbar. Der in dieser Arbeit gezeigte Ansatz verwendet gegen Manipulation gesicherte Hardware in Kombination mit Softwarekomponenten, um eine gesicherte Übertragung und Validierung von Konfigurationsdaten zu ermöglichen, was in weiterer Folge auch ohne die Verwendung von Passwörtern ermöglicht wird. Anstelle von Passwörtern werden Zugangsdaten von der aktuellen Konfiguration abgeleitet, was impliziert, dass jede Konfigurationsänderung automatisch auch die Zugangsdaten ändert und somit die Sicherheit des Systems weiter verbessert werden kann. Abschließend wird ein gesichertes NFC-Transportschicht-Protokoll präsentiert, welches die Notwendigkeit von anwendungsspezifischen Sicherheitslösungen überflüssig macht. Zusätzlich ermöglicht dieses QSNFC genannte Protokoll auch einen sehr effizienten Schlüsselaustausch bei wiederkehrenden Verbindungen, was es besonders geeignet für NFC macht.

Contents

List of Figures

List of Tables

List of Acronyms

0-RTT	Zero Round Trip Time
AB	Abort
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AE	Authenticated Encryption
APDU	Application Protocol Data Unit
ASK	Amplitude-Shift Keying
BER	Bit Error Rate
BT	Bluetooth
BYOE	Bring Your Own Encryption
BYOK	Bring Your Own Key
CA	Certificate Authority
CC	Common Criteria
CH	Client Hello
CoC	Coil on Chip
CoM	Coil on Module
CPU	Central Processing Unit
CPS	Cyber-Physical System
DDoS	Distributed DoS
DES	Data Encryption Standard
DH	Diffie-Hellman
DoS	Denial-of-Service
DTLS	Datagram TLS
DSS	Data Security Standard
ECC	Elliptic-Curve Cryptography
ECDH	Elliptic-Curve Diffie-Hellman
EDC	Error Detecting Code
EKE	Encrypted KE
EM	ElectroMagnetic
EMV	Europay, Mastercard, and Visa
EPC	Electronic Product Code
FEC	Forward Error Correction
FSK	Frequency Shift Keying

GRoT	Global RoT
HF	High Frequency
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure
I2C	Inter-Integrated Circuit
I/O	Input/Output
IoT	Internet of Things
IIoT	Industrial Internet of Things
IT	Information Technology
IP	Internet Protocol
JEEC	Joint Encryption and Error Correction
KDF	Key Derivation Function
KE	Key-Exchange
LLC	Last Level Cache
LRoT	Local RoT
LSB	Least Significant Bit
M2M	Machine-to-Machine
MAC	Message Authentication Code
MITM	Man-in-the-Middle
MSB	Most Significant Bit
NCS	Networked Control System
NDEF	NFC Data Exchange Format
NFC	Near-Field Communication
OS	Operating System
OTS	Off-the-Shelf
PC	Personal Computer
PCI	Payment Card Industry
PD	Proportional-Derivative
PER	Packet Error Rate
PKC	Public-Key Cryptography
PKI	Public-Key Infrastructure
QR	Quick-Response
QSNFC	Quick and Secured NFC
REQ	Request
RES	Response

RF	Radio Frequency
RFID	Radio Frequency Identification
RJ	Reject
RoT	Root of Trust
RSA	Rivest-Shamir-Adleman
RSSI	Received Signal Strength Indicator
SC	Security Controller
SCADA	Supervisory Control and Data Acquisition
SD	Standard Data
SH	Server Hello
SK	Session Key
SNR	Signal-to-Noise Ratio
SPAKE	Simple PAssword-based KE
SPI	Serial Peripheral Interface
SQN	Sequence Number
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module
TTL	Time to Live
TTP	Trusted Third Party
UHF	Ultra-High Frequency
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
WSN	Wireless Sensor Network
ZB	ZigBee

1 Introduction

"The S in IoT stands for security."
– Tim Kadlec

In this chapter, we are going to motivate why security for resource-constrained devices such as smart sensors and devices in the Internet of Things (IoT) is important. We also discuss known security issues of smart sensors, define the problem statement of this thesis, list the contributions that are made in this thesis, and finally give a brief outline for the remaining chapters of this thesis.

1.1 Motivation

The number of sensor-equipped devices is rapidly increasing due to sensors being embedded in a wide range of devices. These sensor-equipped devices aim at making processes easier, safer, or more efficient by using sensor data to improve the monitored processes in various domains such as smart homes, smart factories, smart cities, or smart healthcare [1–4]. Such sensor-equipped devices can often be categorized as so-called standalone *smart sensors*.

Huijsing et al. [5, 6] first defined a smart sensor as the combination of a sensor, analog interface circuit, Analog to Digital Converter (ADC), and a bus interface. Kirianaki et al. [7] further formalized this initial definition by Huijsing et al.:

> *[...] a smart sensor is one chip, without external components, including the sensing, interfacing, signal processing and intelligence (self-testing, self-identification or self-adaptation) functions.*

Yurish [8] defined the presence of a microcontroller as a necessary but not a sufficient condition for such sensors to be intelligent. The author furthermore introduced the term intelligent sensor for such devices. Morris and Langari [9] relativized the difference between the definitions by stating that there is no hard distinction between the function of intelligent and smart sensors.

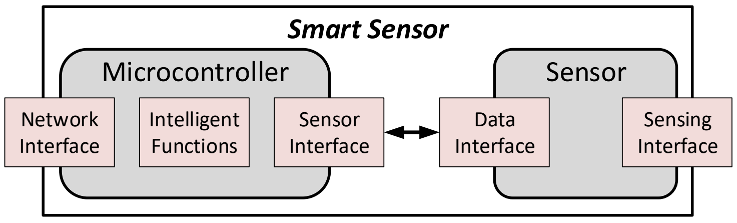


Figure 1.1: Block diagram of smart sensor based on the definition that is used in this thesis.

Thus, in this thesis the term smart sensor is used for systems that include – among others – a sensor and microcontroller and offer some intelligent function such as data aggregation, self-identification, or self-adaption. In addition, it is assumed that in order to be of any use to external systems, the smart sensor includes a networking interface of any form, such as I2C, SPI, Bluetooth, or WiFi. A simple block diagram of such a smart sensor is shown in Figure 1.1.

Due to this definition, a large number of devices including most IoT devices and Cyber-Physical Systems (CPSs) need to be considered as smart sensor as well. In 2016, *Business Insider* [10] predicted a number of 35 billion sensor-equipped devices by 2020. Similar to that, *Statista* [11] predicts the number of IoT devices to grow by 10 % from 27 billion in 2019 to 30 billion in 2020, up to 75 billion by 2025. The corresponding trend from 2015 to 2025 can be seen in Figure 1.2. An even more bold statement from *Cisco and General Electrics* was reported in 2014 by *embedded.com*. According to these estimations, 1 *trillion* sensors would be connected to the Internet by 2020, with a market value of 15 trillion U.S. dollars. Compared to the actual and predicted number of IoT devices, this number seems very high. However, if other domains such as (connected) cars that can contain up to 200 sensors per car (multiplied by roughly 110 million cars per year) [12] are considered, the number becomes reasonable.

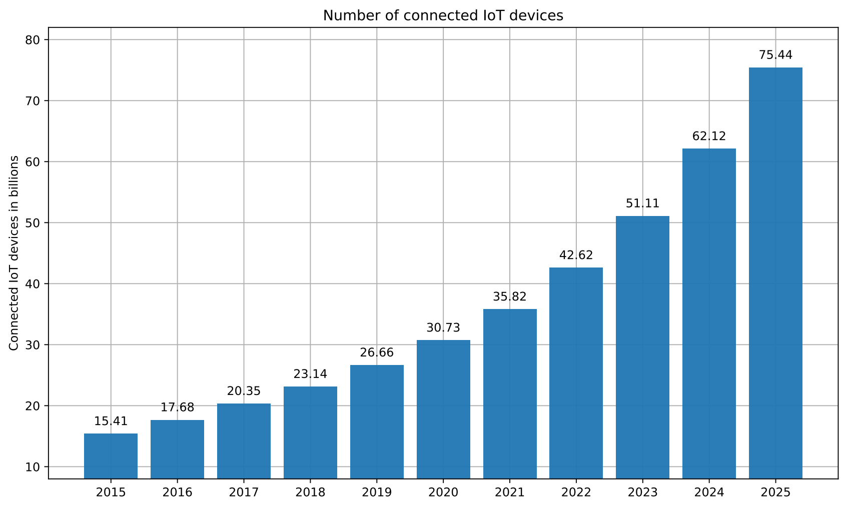


Figure 1.2: Number of IoT devices from 2015 to 2025 according to *Statista* [11].

1.2 Smart Sensor Security

The security of smart sensors is often neglected since these sensors often monitor non-confidential information, such as temperature or humidity. Since this information can typically be measured by anybody at any time, neither the confidentiality, integrity, nor the authenticity of sensor data is protected. Even if sensors are deployed in industrial settings where confidential data might be monitored, security is often reduced to intrusion prevention and the mitigation of eavesdropping on sensor data [13]. However, no fully fledged security is considered for sensors in most cases, which can make systems vulnerable to attacks due to sensor-based security weaknesses. Thus, security needs to be considered for any component of a system, as noted by Perrig et al. [14].

Security is sometimes viewed as a standalone component of a system's architecture, where a separate module provides security. This separation is, however, usually a flawed approach [...]. To achieve a secure system, security must be integrated into every component, since components designed without security can become a point of attack. Consequently, security must pervade every aspect of system design.

Considering this quote, insufficiently protected sensors can be used as an initial entry point for attacks that subsequently target other parts of a system. However, also smart sensors might be an attractive target for attackers as the following three facts pointed out by Kolias et al. [15] highlight.

1. **Number of devices:** Although smart sensors and IoT devices lack computational power compared to traditional Information Technology (IT) systems such as Personal Computers (PCs) or servers, their large number compensates for this fact. Recent attacks such as the Mirai botnet [16] demonstrated that a large number of resource-constrained devices can perform powerful Distributed DoS (DDoS) attacks. In 2016, the Internet service provider *Dyn* was targeted by the Mirai botnet, disabling many popular websites including *Twitter*, *Netflix*, *Reddit*, and *Github*.
2. **Always online:** Due to their field of application in monitoring, most smart sensors and IoT devices are continuously connected to the Internet. This high availability was also demonstrated in the Mirai botnet attack, where 400,000 out of roughly 600,000 infected devices were available to the botnet at any time. The data rate recorded in an attack performed by the Mirai botnet peaked at $1.1\,\mathrm{Tbps}$.
3. **Insufficient security configurations:** Most smart sensors and IoT devices are deployed by their users and never reconfigured. This means that many devices are operated using default configuration parameters such as factory-set username and password combinations. Studies demonstrated that between $10\,\%$ and $40\,\%$ of devices are vulnerable due to using such default configurations [17–19].

Due to these facts, smart sensors and IoT devices are targets of attacks with various different potential ramifications for the device owner. These ramifications range from loss of non-confidential data up to industrial espionage or physical damage to systems that may cause a large negative financial impact or even threaten human lives [20]. The following list gives an overview of potential attacks, categorizing their ramifications based on the *CIA Triad* properties (see Subsection 2.1.2). Many of these attacks can be performed on any embedded system, regardless of whether a sensor is present or not. However, the presence of a sensor usually adds additional attacking points that can be exploited by an adversary. Such sensor-based attacking points include the sensing interface itself or a sensor's configuration interface.

Device capture. If a device is captured, the ramifications are manifold. The adversary is able to access data on the device, install malicious software, and run this software [21]. That is, the adversary is capable of using the device for malicious activities such as espionage, in botnet attacks, or as an entry point for further attacks in a system. This can lead to data loss, noticeable or unnoticeable misbehavior of the system, or even physical damage. Thus, data confidentiality, integrity, and availability may be compromised.

Information stealing. Data that is gathered and processed by a smart sensor can be stolen in various ways. As mentioned previously, if a device is captured, data loss is one of the potential consequences. Since smart sensors are connected to other entities over a wired or even wireless communication channel, adversaries might also be capable of stealing information that is transferred between the smart sensor and other entities. Losing information leads to privacy issues that can be as severe as huge financial losses due to industrial espionage [22]. The property that is impacted by information stealing is data confidentiality.

Side-channel attacks. In addition to loss of confidentiality, stolen sensor measurements can also be exploited as potential side-channel information to reveal information that is not directly monitored by the respective sensors. For instance, attacks have been presented where a smartphone's ambient light sensor was used to reveal information the users were typing on their smartphone [23]. Also, sensors might be exploited to transfer data between otherwise isolated processes or even devices. For instance, temperature sensors were used to transfer information between computers that were even physically separated (not touching) from each other [24].

Deception attacks. Unlike the previous two attacks where sensor data is stolen, the adversary injects malicious sensor data into a system in deception attacks. Similar to information stealing, there are various ways of achieving this goal. On a captured device, sensor data can directly be modified before this data is aggregated or sent to other entities. In addition, sensor data can be manipulated while it is being transferred over the wired or wireless communication channel over which sensor data is transmitted to other entities [25]. However, not only data manipulation but also data injection or replay attacks can be seen as a form of deception attack. These attacks target the integrity and availability of the system and may lead to physical damage if malicious sensor data is used to control a physical process.

Sensor data spoofing. Similar to deception attacks, in sensor data spoofing attacks the adversary tries to manipulate sensor data. However, in this type of attack, actually measured sensor data is not manipulated, but the sensor is tricked by artificially fabricated malicious physical properties into measuring wrong values [26]. Since the adversary needs to be capable of introducing artificial physical stimuli such that they are measured by the sensor, these attacks require closer proximity of the adversary, as opposed to deception attacks that are performed in software. That is, the possible distance of sensor data spoofing stimuli is usually limited by the range of the emitting device such as lasers. These types of attacks target a system's integrity and availability and may also lead to physical damage.

Physical attacks. If an adversary has direct physical access to the smart sensor, so-called physical attacks or active side-channel attacks can be performed. In this type of attack, the adversary actively tampers with the device to reveal confidential information such as cryptographic keys. To do so, semi-invasive or even invasive attacks that destroy the attacked device can be performed by the adversary. Such attacks may include probing existing interfaces, removing chip coatings for probing and tampering with the underlying hardware, or trying to induce faults by injecting voltage spikes [27]. Depending on the leaked information, the confidentiality, integrity, and availability of a system may be compromised.

Denial of Service. If an adversary is neither interested in stealing a system's data nor in influencing the system's behavior but just in disabling the system, so-called Denial-of-Service (DoS) attacks are performed [28]. Similar to other attack types, the adversary also has many possibilities to perform a successful DoS attack. If a sensor is captured, it is easy to bring it down and thus disable its own functionality as well as the correct functionality of the system that relies on the sensor's data. Another method of DoS attacks is, for instance, the jamming of the communication channel that is used to transmit sensor data from the smart sensor to other entities in the system. The sensor might also be bombarded by a huge amount of requests, that even if denied, lead to the sensor being incapable of performing its intended tasks. DoS attacks generally target the attacked system's availability.

Although attack taxonomies were defined early and some research regarding smart sensor security was done, the topic is not to be considered well researched. However, recent incidents increased public awareness regarding IoT security and the topic thus is becoming more prominent. A study by *KPMG* [29] revealed that 84 % of IoT adopters have experienced some sort of security breach. A recent study by *Gemalto* [30] concluded that only 48 % of businesses are capable of detecting whether any of their IoT devices was part of a security breach. The same study stated that the belief in IoT security as an ethical responsibility tripled from 2018 to 2019.

Although there seems to be a rising agreement about the importance of security measures for smart sensors and IoT devices, there are several factors that may limit the adoption of security mechanisms for these devices. Four of this factors are discussed in the following list.

1. **Security awareness:** As mentioned earlier, the awareness regarding the security of smart sensors and IoT devices is very low, although recent incidents have led to the awareness tripling from 4 % of study participants in 2018 to 14 % in 2019; however, compared to other domains, this value is still very low. Without an inherent security awareness, device owners will not adopt increased security measures [31].
2. **Resource-constrained devices:** Smart sensors and IoT devices are usually resource-constrained. That means that classical and well-established security methods cannot be applied since these methods might be too complex or resource intensive in terms of computational power and energy consumption. Thus, security mechanisms tailored for these resource-constrained devices need to be proposed such that a high level of security can be provided without straining the device's processor or power source too much [32].
3. **Device cost:** Introducing security to devices that are usually aimed at being low cost will increase the costs of such devices considerably. However, if device costs are increased too much, potential buyers might opt for alternatives without these cost-intensive security measures. Usually a trade-off between the provided level of security and costs can be made [33], and this factor thus needs to be considered when introducing security measures to low-cost devices such as smart sensors or IoT devices.
4. **Usability:** If security measures that negatively influence the usability of devices are introduced, the device owners will tend to dislike these security measures [34]. Often, the security measures are then disabled, thus making them even less useful than less secure security measures with better usability. Therefore, a usability-security trade-off has to be considered.

1.3 Problem Statement

Due to the high amount of potential attack scenarios and security vulnerabilities in systems that contain smart sensors, the security of such systems needs to be considered as a domain for further research. However, the limiting factors mentioned in Section 1.2 need to be considered in order to propose security measures that are well-suited for the domain of smart sensors. Thus, in this thesis, potential new attack scenarios and suitable countermeasures that target smart sensors are identified. In addition, possibilities for secured, easy-to-use, and efficient configuration of such smart sensors are investigated since default configurations and unprotected interfaces are a major weakness of today's smart sensors. Based on these observations, the following *Research Questions* were formulated at the beginning of this thesis.

RQ1: *Are there yet unknown types of sensor-based attacks that target unprotected interfaces of smart sensors?*

RQ2: *How can the trustworthiness of smart sensors be improved by facilitating a secured, yet efficient and easy-to-use configuration update process?*

The following two subsections discuss both research questions in more detail, with Subsection 1.3.1 addressing RQ1 and Subsection 1.3.2 addressing RQ2.

1.3.1 Sensor-Based Security Issues

Resource-constrained devices such as smart sensors are plagued by different potential attacks that target the smart sensor's correct functionality as already discussed in Section 1.2. So why would it be beneficial to devote the first research question of this thesis to finding new sensor-based security issues? The reason for this is that efficient countermeasures that mitigate security issues can only be investigated if a security weakness is already known. That is, any security measure that was presented addressed existing security issues [35]. For instance, the research on tamper-resistant algorithms and hardware began only after side-channel attacks were used to reveal confidential information. That is, revealing new types of vulnerabilities is very beneficial to security research.

As a first part in answering RQ1, countermeasures for existing sensor-based security issues are briefly analyzed regarding weaknesses. One of the most common sensor-related attack types are deception attacks. In this type of attack, sensor data is either manipulated, injected, or replayed. Such an attack is capable of tricking a controller that relies on the provided sensor data into unsafe operation, and thus, may cause physical damage to the system, the physical process itself, or even to human beings. In this first part, we demonstrate that existing cryptographic countermeasures indeed are capable of mitigating deception attacks due to targeted data manipulation; however, these countermeasures actually make the system more prone to attacks that try to force the system into an unstable state by rendering any sensor data useless for the controller.

As a second part in answering RQ1, we identify a sensor interface that is usually unprotected. Exploiting this sensor interface to sabotage the sensor's desired behavior on its own is already a serious security issue. However, such an attack would be trivial to perform on unprotected interfaces and malfunctioning sensors might be easy to detect for any monitoring system. Instead, we exploit unprotected configuration interfaces by using them as a side channel while building so-called covert channels to transport data between otherwise isolated processes. Contrary to making a sensor useless by exploiting the unprotected configuration interface and applying malicious configuration data, we aim at actually using the sensor and its configuration interface as intended. That is, normal sensor functionality should not be impacted, and thus, systems that monitor the sensor's correct functionality would detect no anomaly in its operation. Such undetected malicious operation and covert channels may compromise a system's confidentiality, integrity, and availability by presenting a serious security weakness that can be exploited for stealing confidential information, controlling isolated processes, or by allowing malicious code to be injected into a smart sensor. Of course, we are not only going to present and demonstrate such security issues, but also discuss how easy it would be to detect the respective malicious behavior as well as potential countermeasures for mitigating the presented attacks.

Therefore, in answering research question RQ1, we actually demonstrate attacks targeting the two following interfaces of a smart sensor. Figure 1.3 depicts sensor interfaces that we exploited in this thesis and other interfaces for which no attacks were performed.

Data interface. This interface is used to transfer measured data from the sensor to other entities in the system that are interested in the sensor's measurements, such as a controller in a CPS. This interface presents an actual networking interface, and thus, information is transferred either via a wired or wireless communication channel in which the presence of an adversary needs to be considered.

Configuration interface. This interface allows changing non-confidential configuration data such as sampling frequencies or confidential data such as Wi-Fi credentials that are used for connecting a smart sensor to a network. A sensor's configuration interface might be accessible via a wired or wireless communication channel such that any entity having network access is allowed to change configuration parameters or it might be restricted to processes that are running on the actual smart sensor itself.

Attacks targeting the sensing interface of a sensor are not covered in this thesis. Contrary to the data interface and configuration interface which are connected to a communication interface and can thus be exploited remotely via software attacks, the sensing interface requires physical presence of the adversary and attacks need to be performed on the physical layer.

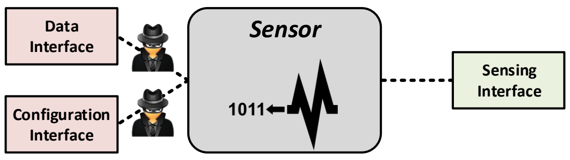


Figure 1.3: Interfaces for which attacks are presented in this thesis. Both data interface and configuration interface are targeted, while attacks targeting the sensing interface are not covered in this thesis.

1.3.2 Secured Sensor Configuration

Most smart sensor-related attacks that were discussed in Section 1.2 can be caused by insufficiently protected interfaces or due to default authentication credentials such as username and password combinations that are used on the attacked devices. Therefore, the research question RQ2 of this thesis tries to address the secured configuration of smart sensors. On the one hand, such a configuration process might encourage smart sensor owners to change default configurations and update security-relevant parameters more frequently. On the other hand, the configuration interface itself might be subjected to attacks and thus be an additional attack vector that further weakens smart sensor security. Therefore, such a configuration approach must also be sufficiently secured to not introduce additional security weaknesses. In the following list, requirements for such a smart sensor configuration approach are given.

Req1 Smart sensor configuration should be possible in a secured manner such that the confidentiality, integrity, and authenticity of configuration data is protected.

Req2 Secured smart sensor configuration should provide high usability such that high user acceptance is achieved due to the ease-of-use of the proposed solution.

Req3 The secured configuration approach should be efficient such that it is applicable to mostly resource-constrained devices such as smart sensors.

Req4 The proposed smart sensor configuration approach should be suitable for both home and industrial usage alike. That is, the approach should scale to large numbers of devices.

Req5 Instead of distinguishing between certain lifecycle phases, the proposed smart sensor configuration approach should rely on the same mechanisms for applying configurations during the smart sensor's entire lifecycle. Configuration updates should be possible in any phase.

Req6 Off-the-Shelf (OTS) hardware should be supported such that no additional hardware is required for using the proposed smart-sensor configuration approach. Thus, the cost overhead due to requiring additional hardware is kept at a minimum.

As can be concluded from this list, the three requirements *security*, *efficiency*, and *usability* are demanded for a smart sensor configuration approach. However, these three entities usually influence each other negatively [36], as depicted in Figure 1.4. As can be seen there, all three requirements and their impact depend on each other. If the attribute of one requirement is improved, usually the other two attributes experience a negative impact. For instance, improving the efficiency of a smart sensor in terms of required computational power and power consumption could be achieved by using weaker but more efficient cryptographic algorithms which weakens security, or by having less powerful interfaces which negatively influences the smart sensor's usability. However, since the main goal of this thesis is to improve the trustworthiness of smart sensors, any presented measure is aimed at increasing security. Thus, to account for the presented security-efficiency-usability trade-off, the goal of this thesis is to keep the negative impact on efficiency and usability as low as possible or not even influence these parameters at all while improving security. Therefore, the evaluation of the presented smart sensor configuration approach will focus on all three of these properties and demonstrate improvements as well as the severity of potential negative impacts on other dependent properties.

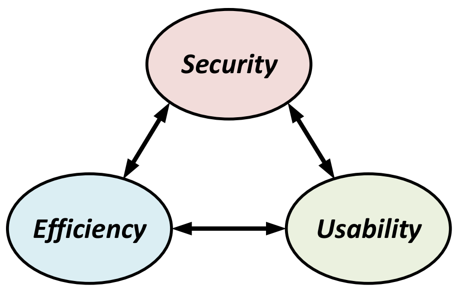


Figure 1.4: Tradeoffs between security, efficiency, and usability in smart sensor configuration approaches.

Another major requirement regarding smart sensor configuration is the possibility to use the same configuration approach during the smart sensor's entire lifecycle. This requirement is specifically addressed in the *IoSense* [2] research project to which parts of this thesis contributed. IoSense is a European research project, with the aim of improving the flexibility of manufactured sensors.

During the smart sensor's lifecylce, various phases starting with manufacturing of the device until the decommissioning of the smart sensor are passed through. During any of these phases, configurations may need to be applied to the smart sensor. In this thesis, the following four phases are defined, three of which require updating the device's configuration. The resulting smart-sensor configuration lifecycle is shown in Figure 1.5.

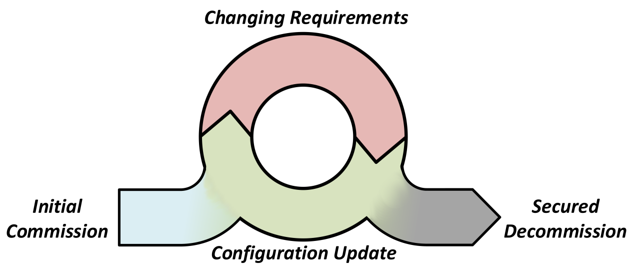


Figure 1.5: Configuration steps during the entire lifecycle of a smart sensor.

1. **Initial Commission:** During manufacturing of smart sensors, initial commission is performed where the default configuration of the respective sensor needs to be transferred to the device. Such default configurations may contain general parameters such as a default sampling frequency, but also confidential information such as initial keys.
2. **Changing Requirements:** Usually upon deploying the smart sensor in its intended environment, the requirements regarding the smart sensor's operational parameters change. However, during normal operation requirements such as a sampling frequency may also change. In addition, security-relevant operations such as key changes may need to be performed.
3. **Configuration Update:** To account for the changing requirements, an updated configuration needs to be applied to the smart sensor. In industrial settings there might be different authorization levels for configuration updates. Crucial configuration updates such as calibration data may only be updated by selected (external) service personnel, while other configuration data such as sampling frequencies may be updated by any authorized person.
4. **Secured Decommission:** If a smart sensor is discarded from its normal operation, a secured decommission process includes the elimination of confidential configuration data such as production-relevant information or cryptographic keys. This can usually be done by overwriting current configuration data either with the initial default configuration such that the sensor can be re-sold, or with dummy data that may render the smart sensor useless.

1.4 Contributions

The contributions of this thesis are manifold in their scope, and thus, are split into several parts. Figure 1.6 gives an overview of how each particular contribution adds to answering the research questions that were defined in Section 1.3.

[2] http://www.iosense.eu/

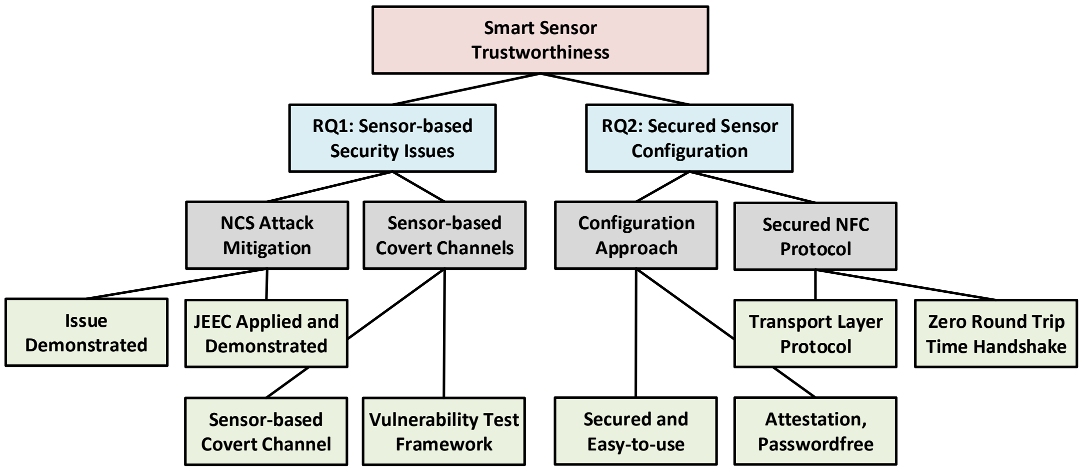

Smart Sensor Trustworthiness
RQ1: Sensor-based Security Issues
RQ2: Secured Sensor Configuration
NCS Attack Mitigation
Sensor-based Covert Channels
Configuration Approach
Secured NFC Protocol
Issue Demonstrated
JEEC Applied and Demonstrated
Transport Layer Protocol
Zero Round Trip Time Handshake
Sensor-based Covert Channel
Vulnerability Test Framework
Secured and Easy-to-use
Attestation, Passwordfree

Figure 1.6: Overview of contributions of this thesis with mapping to the respective research sub-questions.

1.4.1 NCS Attack Mitigation

In this thesis, we argue that current state-of-the-art crytographic countermeasures against deception attacks introduce a new potential attack vector for systems where sensor and controller are distributed over a network. If sensor data in such systems is protected by encryption or even Authenticated Encryption (AE) to provide data confidentiality, integrity, and authenticity, a potential adversary is able to render any packet that contains sensor data useless if a single bit of transferred data can be flipped. Therefore, in this thesis, we demonstrate this issue based on simulations and propose to use Joint Encryption and Error Correction (JEEC), a method commonly used in satellite communication, as a potential countermeasure.

Novel concepts: We demonstrate that deception attack mitigations that are currently state of the art for systems controlled over a network may lead to a new attack vector. We defined this new type of attacks, as so-called *bit-flip attacks* for which we propose to use JEEC as a countermeasure which was, to our best knowledge, never applied in such a context.

1.4.2 Sensor-based Covert Channels

In literature, many sensor-based covert channels have been presented that exploit sensors for transferring data over a covert channel to bypass isolation mechanisms. However, none of the presented approaches exploits the sensor's unprotected configuration interface. Thus, in this thesis, we present novel sensor-based covert channels that exploit these unprotected interfaces. Our work highlights the importance of protecting a smart sensor's configuration interface. To facilitate easier evaluation of sensor-based security weaknesses, a framework is presented that contributes to finding security issues that can lead to such sensor-based covert channels.

Novel concepts: We demonstrate covert channels that are based on exploiting a sensor's configuration and sensing interface. These covert channels can be used to bypass process isolation which we demonstrate for various popular systems. We also include measures to provide reliable data transfer in our covert channel. To our best knowledge, no covert channel that is exploiting the same weaknesses as our proposed approach was presented yet.

1.4.3 Configuration Approach

Since the secured configuration of smart sensors is of utmost importance to prevent leakage of confidential configuration data as well as to prevent security issues such as sensor-based covert channels that exploit unprotected configuration interfaces, a secured, efficient, and easy-to-use configuration approach for smart sensors is presented in this thesis. This approach comprises the end-to-end secured transport of configuration data from a back-end to the smart sensor via Near-Field Communication (NFC), the attestation of successfully applied configuration data, and the password-free authentication to improve the approach's usability.

Novel concepts: We present a smart sensor configuration approach that removes the configuration interface of a sensor from the network as well as from local processes. To facilitate a configuration process, an NFC-based configuration interface is proposed. We also demonstrate two-layered attestation methods for configuration data and a password-free authentication method. To our best knowledge, no configuration approach that is based on NFC and provides the same security features was presented yet.

1.4.4 Secured NFC Protocol

To facilitate secured NFC data transfer, a secured transport layer protocol is presented in this thesis. This protocol is capable of providing data confidentiality, integrity, and authenticity while operating underneath the application layer such that secured data transfer of arbitrary application data can be provided to NFC-based applications. Since NFC provides a limited bandwidth, the goal in designing this protocol is to provide security that is comparable to the Transport Layer Security (TLS) protocol that is used in the Internet, while being as efficient as possible. Thus, the presented protocol satisfies the Zero Round Trip Time (0-RTT) attribute where no complete round-trip is required to agree on a shared key.

Novel concepts: We present a secured transport layer protocol for NFC that removes the necessity to implement security measures in the transport layer, as is currently state of the art for NFC applications. The protocol is tailored for the limited bandwidth provided by NFC which makes it more feasible for NFC applications compared to other secured transport layer protocols. To our best knowledge, we are the first to demonstrate such a protocol for NFC.

1.5 Outline of this Thesis

The remainder of this thesis is organized as follows. In Chapter 2 the key terms related to this thesis are defined and important principles such as attacks and cryptographic primitives are presented. Chapter 3 summarizes and discusses existing work related to sensor-based security issues as well as to secured sensor configuration. The work conducted within the context of this thesis that is related to sensor-based security issues is then presented in Chapter 4. Secured sensor configuration approaches and the respective methods that were developed in the context of this thesis are then presented in Chapter 5. The previously presented work is then evaluated with respect to security and the induced overhead in Chapter 6. Finally, this thesis is concluded in Chapter 7 where all obtained results are summarized. In addition, this chapter also provides some suggestions for future work. All publications that this thesis is based on are then appended in Chapter 8.

2
Background

In this chapter, the key security terms related to this thesis are defined and briefly explained. Also, the basic system model of an insecure communication channel is defined. Security primitives that this thesis builds upon are introduced and briefly explained. In addition, contact-less NFC communication is presented and the technology's security is briefly analyzed.

2.1 Security Definitions

Related to security, the key attributes and requirements regarding secured systems, as well as attack methods used by potential adversaries, are discussed in this section.

2.1.1 Basic System Model

Figure 2.1 shows the basic system model that needs to be considered in any context where secure or insecure information exchange is performed. As can be seen, usually at least three entities, *Alice*, *Bob*, and *Eve*, need to be considered. Alice and Bob intend to exchange information which might be confidential. In general, the communication channel that Alice and Bob use needs to be considered untrustworthy, e.g., due to using wireless communication technologies. Depending on the threat model, the adversary *Eve* is considered to be capable of eavesdropping, manipulating, and inserting information into an ongoing information exchange between Alice and Bob. A potential presence of Eve is, however, not known to the communicating partners Alice and Bob.

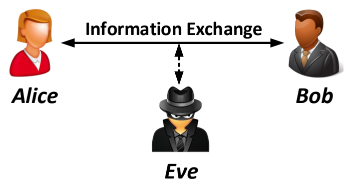


Figure 2.1: Basic security related system model.

2.1.2 CIA Triad

In information security, the three most crucial attributes are summarized as the so-called *CIA triad* [37]. Only if all three of the attributes *confidentiality, integrity*, and *availability* can be achieved, a proper level of security may be provided by a system.

Confidentiality. Confidential information should not be disclosed to an unauthorized entity that is not intended to read this information. Only authorized entities should be able to read such data. Thus, information that is sent over an untrusted channel needs to be protected such that it is not easily readable by unauthorized entities.

Integrity. Information should not be altered by an unauthorized entity in a malicious and unnoticed way. Modifications may include altering data as well as corrupting data, both of these modifications should be detected. Thus, additional information needs to be added to information that is sent over an untrustworthy channel to detect such modifications.

Availability. Information should be available to authorized entities at any time they intend to access this information. If an adversary is able to prevent usage of a system, the attack often is considered just as successful as an attack that would reveal or modify confidential information. Thus, the availability of data that is accessible over an untrusted channel needs to be protected by suitable countermeasures.

2.1.3 DAD Triad

Contrary to the CIA triad that is discussed in Section 2.1.2 the so-called *DAD triad* (*disclosure, alteration, denial* [37]) describes the three major methods used by adversaries for breaking one or multiple attributes of the CIA triad.

Disclosure. Data disclosure is achieved by unauthorized entities if access to confidential information can be gained. That is, information confidentiality is broken.

Alteration. Whenever information is altered by an unauthorized entity, data alteration occurs. If such modifications are not noticed, information integrity is broken.

Denial. By denying authorized entities access to information, denial takes place, and thus, the availability attribute is broken by the adversary.

2.2 Side Channels

The term *side channel* was initially used by Lampson [38] to describe observable events that are caused by code that is executed by a system. As a demonstrative example Lampson described CPU load, which can be heavily influenced by a process running on a system. An observer can then monitor this side channel information to gain knowledge about the running process. Typically, side channels are categorized into *active* and *passive* side channels.

Active. To perform active side channel attacks, the adversary typically needs physical access to the device under attack. Thus, these attacks are also called *physical attacks*. The adversary actively tampers with the device under attack by exploiting side channels such as the device's power supply [39]. For example, the adversary may provoke information leakage by injecting voltage spikes into the device under attack. The adversary also may physically destroy the device by removing protective layers, in order to collect information such as device voltages by probing internal data lines.

Passive. If side channel information is leaked by the device without the adversary *actively* provoking information leakage, so-called passive side channel attacks are performed by an adversary. In a typical setting, one entity just observes the leaked side channel information of another entity, typically a running process. By doing so, confidential information such as cryptographic keys can be revealed by observing side channel information such as timings, power consumption, or electromagnetic emanation [40–42].

2.3 Covert Channels

Side channel information that is *intentionally* triggered by a process can be used to build so-called *covert channels* where data is transferred over a channel that is not intended for data transfer between the involved entities. In general, a covert channel comprises a *sender*, a *receiver*, and a *side channel* as depicted in Figure 2.2.

Sender. The sender is trying to transfer data that is in its possession to the receiver. Due to system restrictions, access to conventional methods such as sockets and shared memory may be prevented. Thus, the sender *intentionally* triggers leakage via a side channel that must be observable by the receiver.

Side channel. A side channel needs the following two properties to be exploitable for building a covert channel: (i) The sender needs to be capable of manipulating the side channel. (ii) The receiver needs to be able to observe the leaked side channel information and ideally distinguish transferred data from normal system behavior.

Receiver. The receiver intends to receive data that the sender is in possession of. Since communication between sender and receiver might be monitored or restricted, the receiver observes leaked side channel information, and is thus capable of receiving the sender's transferred data. In addition to receiving data, sender and receiver usually also need to be synchronized to allow data to be transferred successfully.

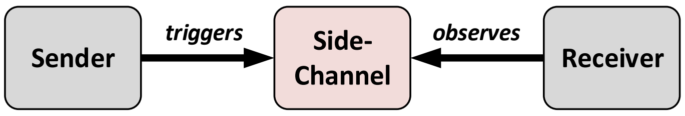


Figure 2.2: Basic architecture of a covert channel.

2.4 Tamper-Resistant Hardware

Most cryptographic algorithms can be implemented efficiently in hardware regarding their performance and power consumption [32, 43]. However, such optimized implementations typically leak side channel information that can be exploited for revealing confidential data such as encryption keys [44]. Other than these passive side channel attacks, physical attacks can also be performed on hardware to reveal confidential data [27]. To prevent security breaches by mitigating such attacks, so-called *tamper-resistant* hardware [45] can be used. Such hardware usually provides both, a protected execution environment for running security-critical code as well as protected storage for confidential data. However, due to increased security, a trade-off in terms of computation power needs to be made. In general, tamper-resistant hardware is not as powerful as general-purpose controllers or dedicated hardware. Thus, Sabt et al. [46] proposed to split a system's execution environment into a secured world and a normal world. This so-called *security by isolation* principle has also been applied in modern Central Processing Unit (CPU) designs such as ARM's Trustzone [47]. The provided level of security of tamper-resistant hardware is often assessed in certification processes such as the Common Criteria (CC) security evaluation [48].

2.5 Cryptographic Primitives

The following cryptographic primitives are used for various purposes throughout this thesis.

2.5.1 Symmetric Cryptography

As its name might suggest, symmetric cryptography requires the same cryptographic key to be used for data encryption as well as for data decryption. The security attribute provided by such algorithms is confidentiality. Since any additional third party that is in possession of this key is capable of reading, sending, and manipulating data, the key is considered a shared secret and needs to be kept private [49]. Today, the most well-known and widely used symmetric cryptography algorithm is the Advanced Encryption Standard (AES) [50]. AES can efficiently be implemented in hardware [51]; however, such implementations often are not tamper resistant, and thus, are prone to attacks due to leaking side channel information [52].

2.5.2 Message Authentication Codes

Similarly to cryptographic hash functions, a Message Authentication Code (MAC) typically is a function that is capable of mapping data of arbitrary size onto a small amount of data of fixed size. However, in addition to validating that data was not modified (integrity), a MAC also authenticates data. That is, it is also validated whether the data originates from the stated source (authenticity) [49]. Similar to symmetric cryptography, a shared secret is required to create and verify the MAC. MAC algorithms can be created by using other cryptographic primitives such as cryptographic hash functions [53], or symmetric cryptographic algorithms [54].

2.5.3 Authenticated Encryption

While symmetric cryptography is capable of providing data confidentiality, a MAC can provide data integrity and authenticity. Thus, by combining these two cryptographic primitives in a secure way, data confidentiality, integrity, and authenticity can be provided. This so-called AE provides three modes of operation of which Encrypt-then-MAC is usually suggested due to its provided level of security [55]. In this mode, the plain text is encrypted first; the resulting cipher text is then used to compute a MAC. Both cipher text and MAC then need to be transmitted. A more recent addition to the AE algorithm is so-called associated data that is only added to the MAC calculation but not encrypted [56]. Thus, in this mode of operation, a cipher text, a plain text, and a MAC are transmitted.

2.5.4 Key-Exchange Protocols

Key-Exchange (KE) or key-agreement protocols facilitate a secured KE for two or more communication partners that need to rely on an unprotected, and thus, untrusted communication channel such as the Internet. During the KE phase all involved communication partners can influence the KE process. The final key is then composed of input from all involved communication partners [57]. Although information for composing this final key is exchanged over the untrusted communication channel, an adversary that can eavesdrop the respective information is unable to compose the final key. The Diffie-Hellman (DH) [58] KE protocol is considered to be the most well-known protocol of this class and is used, for instance, for KE in the widely used TLS protocol [59]. To add authentication to the KE process, a so-called Encrypted KE (EKE) [60] is performed. In such a protocol, shared knowledge between communication partners is integrated into the KE process in a way that only authorized partners are capable of agreeing on a final key.

2.5.5 Asymmetric Cryptography

Asymmetric cryptography, also referred to as Public-Key Cryptography (PKC) relies on public-private key pairs. Contrary to symmetric cryptography where both parties share a common secret, in asymmetric cryptography the private key is kept secret, while the public key can be disseminated to other entities. Any entity can encrypt information using the public key, while only the entity holding the private key is capable of decrypting this information. In addition to encrypting and decrypting data, asymmetric cryptography can also be used for data authentication. The entity that is in possession of the private key signs information using this key, while all entities that are in possession of the corresponding public key can verify this data and that the sender was in possession of the correct private key. Two of the most well-known algorithms of this category are Rivest-Shamir-Adleman (RSA) [61] and Elliptic-Curve Cryptography (ECC) [62].

2.5.6 Certificates

In asymmetric cryptography, public keys are usually not authenticated, and thus, attacks could be performed where an adversary impersonates another entity by replacing public keys [63]. Therefore, public keys are usually distributed using so-called Public-Key Infrastructures (PKIs) and certificates where signatures are used to provide authentication. Such certificates are usually signed by a Trusted Third Party (TTP), the so-called Certificate Authority (CA) [64].

2.5.7 Device Attestation

To verify the trustworthiness of devices, so-called attestation processes can be used. In such a process, the verifier challenges a device to prove its trustworthiness [65]. The device that is being attested either has an existing local verification procedure or gets one from the verifier before the actual verification process. In most cases, the memory of devices is verified in such processes. If the verifier and the entity that is being attested are located on different devices, attestation is performed remotely [66]. Attestation is mostly used to discover unauthorized software modifications of a system, for example, malicious changes to software that is running on a device [67].

2.6 Near-Field Communication

NFC is a contact-less communication standard [68, 69] that is based on Radio Frequency Identification (RFID) technology and related standards. The Radio Frequency (RF) used by NFC is $13.56\,\mathrm{MHz}$ which is similar to High Frequency (HF) RFID. The typical communication range of NFC is approximately $10\,\mathrm{cm}$ over which NFC is capable of providing data rates of up to $848\,\mathrm{kbps}$ using Amplitude-Shift Keying (ASK) as a modulation scheme. Generally, NFC is considered to be a well-established technology in various domains such as payment, access control, and ticketing [70–73]. One of the key factors of success for NFC is the ease-of-use of bringing two devices into close proximity to initiate a data transfer [74]. Recently, the IoT is also believed to become a major domain for NFC applications [75] to link the real world with the digital world. However, this will also pose new challenges for NFC technology such as standardized secured protocols.

2.6.1 Security of NFC

Due to the limited communication range of NFC, the term *security by proximity* was established. However, although NFC communication is typically limited to some centimeters, eavesdropping of NFC communication data has been shown to be possible over distances of up to $10\,\mathrm{m}$ for active and up to $1\,\mathrm{m}$ for passive communication [76]. Haselsteiner and Breitfuß [76] list and discuss the following possible attacks that may target NFC communication.

Eavesdropping. Since communication between NFC devices is based on RF waves, eavesdropping is an obvious issue for NFC since an adversary is able to eavesdrop communication between two entities by placing an appropriate antenna within the range in that the respective RF waves can be received.

Data corruption. An adversary that is not interested in transmitted data but still wants to manipulate a system's functionality, might try to corrupt transferred data by interfering with an ongoing NFC communication in a way such that transferred data is corrupted.

Data modification. If an adversary intends to send manipulated data to a device, ongoing NFC communication can be manipulated by interfering according to the respective coding and modulation scheme. For instance, a '0' value could be turned to '1' by raising a signal level.

Data insertion. If the adversary not only wants to manipulate single bits of information but insert complete messages into an ongoing NFC communication between two entities, data might be sent while one entity pauses, for instance, due to information processing taking a certain amount of time.

Man-in-the-Middle. In a Man-in-the-Middle (MITM) attack two entities, *Alice* and *Bob*, want to communicate with each other. However, without Alice and Bob noticing, a third and malicious entity *Mallory* is placed between Alice and Bob and communicates with both. Mallory then impersonates Bob when communicating with Alice, while impersonating Alice when communicating with Bob. Thus, Mallory is able to see all ongoing communication between Alice and Bob. However, since NFC only supports relatively short communication ranges, MITM attacks that target NFC communication are hard to mount.

Replay. If an adversary is capable of capturing valid NFC communication, replay attacks can be performed by sending this valid captured data to the device under attack. For instance, if communication for unlocking a door is captured, the adversary then might be capable of reusing the captured data for unlocking said door.

Denial-of-Service. An adversary that successfully makes resources unavailable for its intended users performs a so-called DoS attack.

When analyzing NFC concerning its security vulnerabilities in industrial scenarios, Plosz et al. [77] conclude that many attacks are not prevented by the NFC standard. Van Damme and Wouters [78] further state why NFC communication needs to be protected by suitable measures:

> *We can conclude that even if the NFC standard foresees some features that makes [sic] the attacker's life harder, perfect security can only be obtained when dedicated cryptography is used to establish a secure channel between communicating devices.*

3 Related Work

In this chapter, literature related to the topics of this thesis is discussed. The chapter is split into two sections regarding sensor-based security issues in Section 3.1 and smart sensor configuration approaches in Section 3.2. To highlight the difference of this thesis' contributions and related work, a comparison of key parameters is given at the end of each section.

3.1 Sensor-Based Security Issues

In literature, sensor-related security and the respective security issues haven been mainly discussed in the context of CPSs (e.g., [79–81]) and Wireless Sensor Networks (WSNs) (e.g., [14, 82]). Cárdenas et al. [80] have defined various attack scenarios that could target a CPS and that are shown in Figure 3.1. In this figure, attacks of class (A1) directly target the physical process while attacks of class (A2) either induce false control information $\tilde{u}$ in a way that this information differs from correct control information u ($\tilde{u} \neq u$) or deny this information (DoS attack). Both of these attack classes are not related to sensors whereas attack class (A3) includes an adversary injecting false sensor information $\tilde{y}$ that differs from correct sensor information y ($\tilde{y} \neq y$) or denying this information to the controller (DoS attack). If false sensor information is injected into a system, a so-called *deception attack* is performed by the adversary. Literature related to such attacks is discussed in Section 3.1.1.

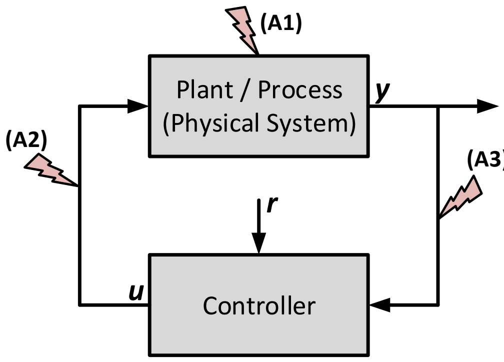


Figure 3.1: Potential attacks targeting a CPS as defined by Cárdenas et al. [80].

For WSNs and smart sensors in the IoT, several studies have highlighted that default authentication credentials as well as unchanged default configurations may lead to severe security breaches [18, 83, 84]. In addition, Cam-Winget et al. [85] have discussed weaknesses of remote-access channels for firmware and configuration updates that can often be exploited. Thus, work related to weak authentication schemes is presented in Section 3.1.2.

Finally, sensor data has not only been exploited for various malicious activities such as spying on users or for industrial espionage [86, 87], but also as side-channel information that may leak other confidential information [88]. Such side-channel information may then also be used for building sensor-based covert channels [89]. Therefore, work related to sensor-based side-channels and covert channels is presented in Section 3.1.3.

3.1.1 Deception Attacks

As has been mentioned by Cárdenas et al. [80], so-called *deception attacks* tamper with systems by manipulating sensor data or by injecting false sensor data. Such deception attacks usually are easy to detect if a monitoring system is in place [90]. However, to avoid the detection of such attacks, adversaries have improved deception attacks and so-called stealthy deception attacks have been built that are harder to detect [91, 92]. Probably the most well-known attack in this category has been widely covered in media as Stuxnet attack [93]. In this attack, malicious code has been injected into the control systems of uranium enrichment plants with the goal of destroying these facilities. The attack has been performed in two phases, where in the first phase legitimate sensor data has been learned that could then be injected to the correct controller code in order to conceal malicious system behavior that has been intended to destroy the system under attack. This approach has started a new era of cyber-attacks. Rather than stealing or manipulating data, the attacker's target is to physically destroy a system which might cause severe damage and also harm human beings. Such safety-related issues have been mentioned as a problem for any CPS [22].

In literature, deception attacks that target Supervisory Control and Data Acquisition (SCADA) systems (e.g., for energy or water management) are a well-covered topic. Amin et al. [94] have presented a deception attack that targets a canal system in France. The successful attack has been based on manipulating the system controller's behavior by modifying sensor measurements such that water can be stolen without being noticed. Amin et al. [95] have stated that an adversary has to have knowledge of (i) the system dynamics, (ii) the diagnostic system, and (iii) sensor-related signals in order to successfully perform stealthy deception attacks targeting SCADA systems.

Teixeira et al. [96] have demonstrated that state-of-the-art outlier detectors may be used to detect simple deception attacks in an energy-management system. However, the authors also have concluded that stealthy deception attacks that inject false but plausible sensor data into the control system cannot be detected by such detectors. The authors further have suggested either adding additional sensors to the system or securing sensors as possible countermeasures to mitigate stealthy deception attacks [97, 98]. Bobba et al. [97] have identified so-called basic measurements that are sufficient for a state estimator to work correctly, and have used all other measurements to detect injected malicious data. Similarly, Dan and Sandberg [98] have suggested to encrypt measurements of selected sensors to protect the system. The authors have stated that it is not feasible to expect every sensor to be protected, and thus, the attack surface should at least be minimized by protecting as many sensors as possible.

Ding et al. [99] as well as Ma et al. [100] have presented an approach where distributed filters have been designed that not only rely on measurements from an individual sensor but also neighboring sensors' data while also considering the respective topology. The authors have stated that the error of such filters can be minimized, even in the presence of deception attacks. Although such filter-based approaches are capable of mitigating certain deception attacks, the presence of malicious data that influences system behavior in a negative way cannot be eliminated completely.

A special form of stealthy deception attacks, so-called replay attacks, have been discussed by Mo and Sinopoli [101]. In a replay attack, sensor values are captured by the adversary and re-sent to the controller at a later time. As a countermeasure, the authors have proposed a failure detector that, depending on the controller's performance, offers a certain detection rate but no guaranteed detection of an ongoing replay attack.

In addition to countermeasures based on control theory, approaches utilizing cryptographic primitives such as encryption and one-way functions have also been proposed in literature [102, 103]. However, most works have used inadequate cryptographic primitives such as the outdated Data Encryption Standard (DES) or MD5 algorithms. Many publications have dealt with confidentiality, integrity, and authenticity of sensor data in CPSs [104], however, without having considered the detection of deception attacks (e.g., using blockchains for protecting sensor data [105]).

A comparison of approaches presented in this thesis with related work is given in Table 3.1. As a main difference to related work, the approach presented in this thesis provides confidentiality, integrity, authenticity, and tamper-resistance by proposing to use hardware security modules. Also, we discuss attacks that can be performed on other state-of-the-art protection mechanisms and demonstrate JEEC as a countermeasure that was, to our best knowledge, not yet proposed for deception attack mitigation.

Table 3.1: Comparison with related work for deception attacks

	Confidentiality	Integrity	Authenticity	Attack Detection	Remark
Basic measurements [97]	✗	✗	✗	✓	Redundancy complicates attacks
Encrypt selected sensors [98]	✓	✗	✗	✓	Attacks only complicated
Distributed filters [99, 100]	✗	✓	✗	✗	Impact of attack minimized
Failure detector [101]	✗	✗	✗	✓	Attack might not be detected
Encryption [102]	✓	✗	✗	✗	Only confidentiality provided
Encryption and hash function [103]	✓	✓	✗	✓	Combination of outdated algorithms
Blockchain for sensor data [105]	✓	✓	✓	✗	No deception attack detection
This thesis	✓	✓	✓	✓	AE combined with error correction

3.1.2 Weak Authentication Schemes

Weak, or even default username and password combinations have been a major security weakness for smart sensors and the IoT as recent incidents such as the Mirai botnet have shown. According to Kolias et al. [15] only 62 username and password combinations have been used to capture over 400,000 devices. Tam et al. [106] have analyzed the security versus convenience trade-off of passwords. The authors have concluded that not only are users choosing too weak passwords, but they are not changing default passwords at all. If users are forced to choose strong passwords, in many cases the users will write down the passwords or store them on electronic devices such as their smartphones.

To mitigate this issue, so-called two-factor authentication has often been proposed to increase security [107]. In such an authentication scheme, the user typically has to be in possession of two objects for authenticating themselves. Typically, the first object is a shared secret such as a password. The second object usually is a physical thing that the user must be in possession of, such as a smart card [108]. Wang et al. [109] have discussed the issues of such schemes applied to distributed systems. For example, the loss of one authentication factor such as the password or a smart card which cannot be easily accounted for.

Das [107] has proposed an efficient two-factor user authentication scheme that should be especially suited for resource-constrained devices such as smart sensors. The proposed protocol, however, has been based on one-way functions that are prone to attacks such as password guessing or stolen smart cards. In addition, changing passwords has not been supported. Vaidya et al. [110] have improved Das' approach such that it is robust against these issues, including stolen smart card attacks. To achieve this, the authors have suggested to include a central entity such as a gateway node into the authentication protocol. Such central entities can then be used in a system for key management and storing other device-specific configuration data, as has been discussed by Delgado-Mohatar et al. [111].

All of these presented two-factor user authentication schemes have in common that the user is required to remember at least one factor that is the password. A lost or even stolen password thus leads to security issues in such systems. To mitigate these issues, other forms of authentication have been proposed, such as using biometric data [112]. For smart sensors, the actual sensor might be used to collect this biometric data, as has been proposed, for instance, by Choi et al. [113] for collecting heart beat data. Gafurov et al. [114] have suggested to use an accelerometer sensor to identify users based on their gait, while Okumara et al. [115] have used accelerometer sensor data to identify a user's arm sweep action.

All discussed approaches have in common, that some information is stored locally on resource-constrained devices. Thus, as has been highlighted by Benenson et al. [116], credentials can be obtained by adversaries through so-called node capture attacks where physical attacks are performed on captured devices. As a potential countermeasure, Almeshekah et al. [117] have presented an approach where actual credentials are replaced by some dummy credentials while the actual credentials are protected by measures such as tamper-resistant hardware. In addition, contrary to the approach presented in this thesis, none of the above approaches is capable of *automatically* inferring authentication credentials based on certain triggers such as configuration changes. By doing so, the issue of default passwords can also be mitigated.

A comparison of approaches presented in this thesis with related work is given in Table 3.2. As a main distinction, our presented approach not only provides two-factor authentication, but also offers a central back-end where credentials are managed as well as tamper-resistant hardware that mitigates side-channel attacks. In addition, our approach provides automated authentication credential derivation based on configuration changes. To our best knowledge, we are the first to present such an approach.

Table 3.2: Comparison with related work for user authentication.

	Two-Factor	Biometric	Managed	Inferring	Tramper Res.	Remark
Simple Two-Factor [107]	✓	✗	✗	✗	✗	Based on one-way functions
Improved Two-Factor [110]	✓	✗	✓	✗	✗	Third entity in authentication scheme
Managed keys [111]	✓	✗	✓	✗	✗	Central key management instance
Biometric [113–115]	✓	✓	✗	✓	✗	Biometric data to derive credentials
Dummy credentials [117]	✓	✗	✓	✗	✓	Tamper resistant storage
This thesis	✓	✗	✓	✓	✓	Configuration triggers cred. inferring

3.1.3 Side and Covert Channels

Most sensor-based side and covert-channel implementations have been based on principles that originate from other technologies such as network-, memory-, or cache-based covert channels. Thus, first a brief overview of these types of covert channels is given. Typically, these different types can be categorized based on the covert channel's data rate as shown in Figure 3.2.

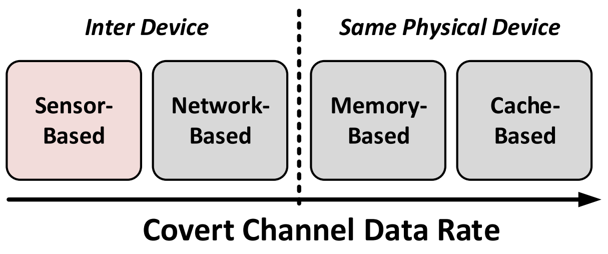


Figure 3.2: Covert channel classification based on achievable data rate.

3.1.3.1 Cache-Based Covert Channels

Modern processors and systems typically leak side-channel information since most of these systems are optimized for performance, energy efficiency, or both [118]. One of the side channels that can be exploited in such optimized systems is cache memory. Such cache-based side channels do not exploit weaknesses in the implementation of the Operating System (OS), but solely rely on the hardware-related timing difference between cache hits and cache misses. If one process is capable of intentionally causing cache hits or cache misses for another process, a covert channel according to the definition given in Section 2.3 can be built. Cache misses can be provoked by a process if all data from specific cache regions is flushed [119, 120]. Since cache is only accessible to the CPU, cache-based covert channels can only be built between processes residing on the same CPU. In general, cache-based covert channels are considered the fastest type of covert channel due to the speed of cache memory. However, any process running on a CPU can use cache memory, and thus, this type of covert channel is also exposed to a lot of interference from other processes.

To mitigate issues caused by interfering cache accesses, methods such as sender and receiver synchronization, data flow control, error detection, and error correction need to be applied. Maurice et al. [121] have demonstrated a covert channel that has provided bit rates of over $45\,\mathrm{KByte/s}$ while achieving a Bit Error Rate (BER) of 0.

3.1.3.2 Memory-Based Covert Channels

Memory is shared across cores in modern systems, and thus, memory-based covert channels can be established between processes that do not run on the same CPU. Similar to cache-based covert channels, most memory-based covert channels also exploit side-channel information that is leaked due to timing differences between certain events. Xiao et al. [122] have presented a covert channel that is based on memory deduplication. If two or more processes share identical physical memory pages and the copy-on-write mechanism is invoked, higher latency can be observed. Wu et al. [123] have used timing differences of memory accesses that are caused by locking the memory bus with atomic operations. The presented covert channel has provided bit rates of up to $747\,\mathrm{bit/s}$ while it also has achieved a BER of 0. Pessl et al. [124] have demonstrated a covert channel that is based on memory address mappings and exploiting row buffers. The authors have claimed that their covert channel offers a channel capacity of up to $2\,\mathrm{Mbit/s}$ which, as they have claimed, is four times as fast as memory-based covert channels that exploit the memory bus. However, the authors have not provided information on bit rates that can be achieved with a BER of 0.

3.1.3.3 Network-Based Covert Channels

Network-based covert channels are one of the earliest known attacks for stealthy data transfer. In contrast to cache-based and memory-based covert channels, network-based covert channels can be used to transfer data between processes that do not reside on the same physical device [125]. In case of network-based covert channels, information is mostly hidden in protocols of different network layers. Frikha et al. [126] have presented a covert channel where information is encoded in the sequence number of the 802.11 protocol's sequence number field. Another covert channel based on network layer protocols has been presented by Tuptuk and Hailes [89] where information is encoded in a sensor node's Received Signal Strength Indicator (RSSI). This value can be altered by the sender by simply modifying transmission power, and thus, is suitable even for very constrained devices that do not run any sophisticated networking protocol. Network-based covert channels that exploit higher-layer protocols often hide information in Transmission Control Protocol (TCP)/Internet Protocol (IP) header fields such as Time to Live (TTL), or timestamps [127–129]. However, in literature, network-based covert channels also have been presented that are independent of any network protocol but rather rely on timing differences, similar to cache-based and memory-based covert channels. Cabuk et al. [130] have demonstrated covert channels that encode information in the timing differences between received IP packets while in the approach that has been presented by Ji et al. [131], information is encoded in the length of transmitted messages.

3.1.3.4 Sensor-Based Covert Channels

The confidentiality of sensor data is of high importance to protect user privacy and to prevent industrial espionage. However, sensor data might not only be used directly for discovering confidential information but also to record side-channel information. This side-channel information can then be used to reveal confidential information. Sensors in mobile devices can be used in a malicious way, for instance, to reveal passwords by measuring vibrations or ambient light while a keyboard is used [23, 132].

Similarly, a covert channel can be built if one process is capable of intentionally triggering effects that can be measured by another process via a sensor. The term covert channel has been coined by Lampson [38] when he reported the first sensor-based covert channel. The author has exploited CPU load that can be increased by one malicious process and observed by another. Brouchier et al. [133] have exploited a device's temperature as side-channel for transmitting information in a stealthy way. These temperature-based covert channels have been shown to be capable of transmitting data even between air-gapped computers [24]. Another physical property that has been used to build covert channels is ultrasonic sound that is emitted by speakers and can be sensed by a microphone [134, 135], also allowing data transfer between different devices. Novak et al. [136] have demonstrated a covert channel that is built using light that is emitted by a device's flashlight and sensed by an ambient light sensor. Similar to that, Al-Haiqi et al. [137] have presented a covert channel that uses a device's accelerometer to measure vibrations that are caused by another process that is capable of controlling the device's vibration motor. Finally, Matyunin et al. [138] have demonstrated a covert channel that uses the ElectroMagnetic (EM) field to transmit information. The sender causes certain effects by triggering Input/Output (I/O) operations to encode data while the receiver can measure the change in the observed EM field.

In contrast to approaches that require access to an actuator to trigger physically observable effects, Tuptuk and Hailes [89] have shown a covert channel that is based on tampering with the Least Significant Bits (LSBs) of sensor measurements. The authors have argued that any sensor measurement should be considered a noisy estimate of the observed physical property, and thus, subtle changes to measured values are hard to detect. Similar to this approach, the covert channels that are presented in this thesis do not require access to actuators but tamper with configuration values of sensors in a way that malicious activity is obfuscated as normal sensor operation.

A comparison of the approach presented in this thesis with the discussed related work regarding sensor-based covert channels is given in Table 3.3. Compared to the state of the art, our presented attacks (both simple and complex approach) do not require actuators to generate physically measurable side-channel information, and thus, we claim that our presented attacks are harder to detect than other state-of-the-art sensor-based covert channels. However, the simple covert channel approach requires read and write access (r/w) to sensor registers while the more complex attack only requires read access.

3.2 Smart Sensor Configuration

In this section, we review work related to various key components of the smart sensor configuration approach that is presented in this thesis. The related work is split into subsections covering key provisioning, transferring general-purpose configuration data, and secured NFC protocols.

Table 3.3: Comparison with related work for sensor-based covert channels.

	Requires Actuator	Detectability	Reliability	Sensor Access (r/w)	Remark
CPU Load [38]	✗	high	low	r	High CPU load easy to detect
Temperature [24]	✗	high	low	r	Temperature controlled by CPU load
Sound [134, 135]	✓	fair	high	r	Ultrasonic sound inaudible for humans
Light [136]	✓	high	fair	r	Emitted light easily observable
Vibrations [137]	✓	high	high	r	Emitted vibrations easily observable
EM Field [138]	✗	fair	fair	r	A lot of interference by other devices
LSB Tampering [89]	✗	fair	high	r	Can be detected by redundancy
This thesis (simple)	✗	low	high	r/w	Needs privileges for configuring
This thesis (complex)	✗	low	high	r	Only triggers sensor readings

3.2.1 Key Provisioning

Secured key provisioning is a topic that has originated in the manufacturing of devices and smartcards where initial keys need to be transferred to the manufactured object. However, it is desirable to allow customers to change these keys [139] in order to eliminate the knowledge of devices' keys by manufacturers. This desire has been summarized in a principle called Bring Your Own Key (BYOK). The BYOK principle allows device users to change initially provisioned keys and use their self-generated keys for cryptographic functions. If users are also allowed to change parameters of the applied cryptographic functions such as a key length, the BYOK principle is extended to Bring Your Own Encryption (BYOE) [140].

Table 3.4: Comparison with related work for key provisioning.

	Secured Transfer	Authenticated	Feasible for Sensors	Attacks Infeasible	Remark
Connected car [141]	✗	✗	✓	✗	No security provided
Time/location aware [142]	✗	✓	✓	✗	Initial key unprotected; relay attacks
Co-presence [143]	✓	✓	✓	✗	Co-presence insufficient measure
Device shaking [144]	✓	✓	✗	✗	Infeasible for stationary sensors
Continuous gesture [145]	✓	✓	✗	✗	Infeasible for stationary sensors
Proximity-based [146]	✓	✓	✓	✗	Proximity insufficient measure
Smartphone gateway [147]	✓	✓	✓	✗	Vulnerable to MITM attacks
This thesis	✓	✓	✓	✓	End-to-end encrypted from backend

The most relevant use case for key updates by a customer is device pairing where keys need to be transferred in order to establish a secured channel between paired devices [148]. Wireless device pairing is often assisted by NFC which allows intuitive transfer of pairing information [149]. Steffen et al. [141] have discussed using NFC for various pairing activities inside a connected car, or for activating additional software components which also require keys to be transferred. Similar to that, Suomalainen [142] has discussed the practicability and security of using NFC-enabled smartphones for pairing IoT devices. As a security measure, the author has proposed to use the device's context that consists of location and time. However, relay attacks that bypass such security measures have been shown practicable for NFC [150]. Similarly, Miettinen et al. [143] have fingerprinted context information over time to allow device pairing without any user interaction. The authors have suggested co-presence of devices over time as the main measure which might not be suitable for scenarios where a malicious device can be placed unnoticed next to others with which the malicious device would then be paired. To mitigate such issues, Mayrhofer et al. [144] have proposed an approach where the devices that should be paired need to be shaken simultaneously. Using captured accelerometer data, a shared secret is then derived that is used for device pairing. However, this approach might not be suitable for smart sensors that are stationary, for instance, temperature sensors that are wall-mounted. Ahmed et al. [145] have presented a similar approach, where a continuous gesture is requested by one device that needs to be performed by the second device in the pairing process. To mitigate limitations for stationary devices, Zhang et al. [146] have proposed to use a smartphone as a gateway when performing a pairing process between two stationary devices. The smartphone is used to perform movements, the resulting RSSI trace is recorded, and finally this sensor data is compared with the requested movement pattern. However, in this approach any person that has physical access to devices can perform such actions, and thus, perform the pairing process between arbitrary devices. Urien and Kiennert [147] thus have presented approaches that use a smartphone as bridge to an authorized key management server in the Internet. However, in their approach the smartphone is seen as trusted, and thus, an adversary is capable of stealing keys by running malicious code on the smartphone.

A comparison of the BYOK approach presented in this thesis and related work is given in Table 3.4. As the main difference to the state-of-the-art we claim that known attacks that target our approach are infeasible. This is based on the usage of tamper-resistant hardware in combination with our presented protocol. Possible issues of other approaches are listed in Table 3.4.

3.2.2 Configuration of General-Purpose Data

Device configuration is a prominent topic in the IoT due to security issues caused by using default configurations [17, 18]. Related work regarding the most important configuration parameters, encryption keys, is discussed in Section 3.2.1. Thus, in this section, general-purpose configuration data is considered. That is, a configuration may contain confidential information such as Wi-Fi keys, but also non-confidential information such as sampling frequencies or thresholds.

One approach to sensor configuration is the self-configuration of devices that not only minimizes connection and organizational overhead [151], but may also remove the need for a configuration interface at all. That is, although not a security measure per definition, attacks that target this interface could effectively be mitigated. He et al. [152] have presented a neural network for self-configuration of WSNs that is capable of finding node clusters using the topology of deployed nodes, and thus, improving network efficiency. Similarly, Fritze et al. [153] have demonstrated an approach for self-configuring multi-sensor systems that automatically perform configurations for sensor connection and fusion. Although convenient for some tasks, all self-configuration approaches have in common that initial configurations still need to be applied using traditional methods as well as the inability to self-configure confidential parameters such as encryption keys.

Due to most devices being connected to the Internet, many approaches have been presented where the configuration interface is made accessible to the Internet. Nastic et al. [154] have presented a cloud-based approach for automatic provisioning and configuration management. The authors have claimed that due to the large amount of devices that need to be configured, a centralized configuration management instance is vital for operating large systems. Perera et al. [155] have presented a framework for automatic sensor discovery and configuration. The authors also have proposed using a centralized instance for configuration management. In addition to a direct connection to sensors, the authors also have suggested using smartphones that are transported by humans or robots for bridging the connection between centralized configuration management instance and sensor. However, allowing device configuration over the Internet imposes security risks for systems that could be mitigated by only allowing local access to the configuration interfaces.

Thus, NFC technology has been proposed by Alimi and Pasquet [156] for modifying device functionality after distribution to customers. In their approach, a payment application has been adapted using NFC. Similarly, Wu et al. [157] have proposed to update computational RFID tags that are capable of performing sensing activities. The update is transferred using the standardized Electronic Product Code (EPC) protocol. The authors have demonstrated that configuration updates could be applied to RFID tags over distances of up to several meters using their approach. Finally, Haase et al. [158] have presented an NFC-based configuration approach tailored to resource-constrained sensors. The authors have demonstrated transferring arbitrary configuration data using OTS smartphones. In addition, the latency of transferring large amounts of payload over NFC also has been evaluated and considered to be feasible for most use cases.

Presented work that is related to the configuration of sensors is compared to the approach presented in this thesis in Table 3.5. Compared to related work, our approach provides data confidentiality, integrity, and authenticity for configuration data while supporting arbitrary payload data. In addition, configurations can be managed on our so-called configuration back-end, configuration data can be transferred to offline mobile configuration devices, and configuration attestation is supported. Also, in our approach, the same configuration interface is used for configuration updates during the smart sensor's entire lifecycle. To our best knowledge, we are the first to present such an NFC-based configuration approach.

3.2.3 Secured NFC Protocols

Typically, NFC connections are protected by suitable measures due to the possibility of attacks, as discussed in Section 2.6.1. However, although standards for the NFC transport layer, such as the NFC Data Exchange Format (NDEF) protocol exist, no sufficiently secured transport layer protocol for NFC has been presented yet. The NDEF protocol contains so-called signature records [159] that are intended for protecting a message's integrity. However, these signature records already have been shown to be susceptible to certain attacks [160], and thus, cannot be considered a sufficient security measure. Eun et al. [161] have presented a privacy-preserving NFC protocol that is based on randomly-generated identities and a management instance. The presented protocol is capable of providing authorization; however it does not consider data confidentiality and integrity.

Table 3.5: Comparison with related work for sensor configuration.

	Payload Protected	Arbitrary Payload	Self-Configuration	Managed / Scalable	Offline Configuration	Entire Lifecycle	Attestation	Remark
Key provisioning	✓	✗	✗	✗	✓	✗	✗	Only keys supported
Neural network [152]	✗	✗	✓	✓	✗	✗	✗	Network topology learned
Sensor fusion [153]	✗	✗	✓	✓	✗	✗	✗	Autonomous sensor fusion
Cloud-based [154]	✗	✓	✗	✓	✗	✗	✗	Configuration management
Bridged [155]	✗	✓	✗	✓	✗	✗	✗	Smartphone bridge
Post distribution [156]	✓	✗	✗	✓	✓	✗	✗	Adapt application via NFC
RFID-based [157]	✗	✓	✗	✗	✓	✗	✗	EPC protocol
NFC-based [158]	✓	✓	✗	✗	✓	✓	✗	Initial key not protected
This thesis	✓	✓	✗	✓	✓	✓	✓	Managed, NFC-based

Similarly, Odelu et al. [162] have presented an authentication protocol that also provides authentication based on using pseudonyms generated in a secured way. Here also, no data confidentiality or integrity is provided. Toyoda and Sasase [163] have proposed a secret sharing mechanism that is suitable for NFC. However, the protocol itself does neither provide authorization nor data confidentiality, integrity, or authenticity. Instead, it rather needs to be seen as an enabling protocol that helps to solve the key distribution problem. Li et al. [164] have presented mechanisms for authentication and authorization between RFID tags and readers to establish a trust relationship between these two entities as well as enable a protected data channel. Finally, Urien and Piramuthu [165] have presented a secured transport layer protocol for NFC that is based on the TLS protocol. The protocol is capable of providing data confidentiality, integrity, and authenticity. However, compared to the approach presented in this thesis, the presented protocol entails a larger overhead in terms of round trips which should be minimized for NFC communications.

Since no suitable secured NFC transport-layer protocol has been presented yet, many application-specific solutions have been proposed in literature. All of these solutions have in common that they are tailored for a specific application, and thus, have to be considered as being application-layer protocols. In general, such protocols are not generalized well enough to be applicable for other applications. Especially in the payment sector, many application-specific security solutions have been presented due to the confidentiality of data that needs to be transferred in this sector. Kadambi et al. [166] have proposed to extend the Payment Card Industry (PCI) Data Security Standard (DSS) in such a way that NFC-enabled mobile devices such as smartphones may also be used for payment transactions. The PCI DSS regulates what data is allowed to be stored, as well as the required security measures. Similar to that, Ceipidor et al. [167] have presented an approach that adds authentication as well as data confidentiality to the existing Europay, Mastercard, and Visa (EMV) standard for mobile proximity payment. As a form of device pairing in the payment sector, usually bringing two devices in close proximity is considered as a sufficient security measure. However, to prevent relay attacks, Halevi et al. [168] have suggested to use sensor data to verify whether both devices are close to each other.

Another sector where security is of high importance is the healthcare sector. Here, data confidentiality needs to be provided to protect user privacy. Dünebeil et al. [169] have proposed using protected NFC tags to store a patient's medical information. This information can then be used by authorized personnel such as caretakers or emergency responders. Since such tags need to be placed at highly visible locations, a high level of security is critical. Sethia et al. [170] have presented an approach for a healthcare data management system that also relies on NFC-enabled smarthpones and NFC tags. In their approach, patient data is stored on a protected health card that could either be a traditional smartcard or data could be stored on an NFC-enabled smartphone. Besides the payment and healthcare sectors, unconventional new use cases have also emerged where a secured NFC channel is required. For instance, Busold et al. [171] have presented a car immobilizer framework that can be used for flexible car access policies.

Related work for secured NFC protocols is compared to the approach presented in this thesis in Table 3.6. Contrary to other approaches, we present an approach that provides confidentiality, integrity, and authenticity on the NFC transport layer. Thus, our approach enables building secured NFC-based applications without the need to implement security-relevant functionality in the application layer.

Table 3.6: Comparison with related work for secured NFC protocols.

	Authorization	**Confidentiality**	**Integrity**	**Authenticity**	**Efficient**	**Transport Layer**	**Remark**
Signature records [159]	✗	✗	✓	✗	✓	✓	Considered broken
Authentication [161, 162]	✓	✗	✗	✗	✓	✓	Only authorization provided
Key distribution [163]	✗	✗	✗	✗	✓	✓	Enabling protocol
RFID authentication [164]	✓	✓	✗	✗	✓	✓	Trust relationship
TLS over NFC [165]	✓	✓	✓	✓	✗	✓	Larger overhead
Payment sector [166–168]	✓	✓	✓	✓	✓	✗	Application specific
Healthcare sector [169, 170]	✓	✓	✓	✓	✓	✗	Application specific
Car immobilizer [171]	✓	✓	✓	✓	✓	✗	Application specific
This thesis	✓	✓	✓	✓	✓	✓	Secured Transport Layer

4 Sensor-Based Security Issues

In this chapter we present two different types of attacks that target systems via sensor interfaces. The first type, Networked Control System (NCS) attacks, demonstrates security issues of systems where sensor information is already protected by encrypting data. The second type, sensor-based covert channels, demonstrates methods to exploit sensors for building stealthy communication channels that are capable of bypassing process isolation methods.

4.1 NCS Attacks

In this section we first define the system model that is used to demonstrate NCS-based attacks. After that we show why applying encryption in an NCS may be problematic, define a new attack type, and propose a countermeasure to mitigate such attacks. The content of this section is mainly based on work published in the paper *Towards Trustworthy Data in Networked Control Systems: A Hardware-Based Approach* [172].

4.1.1 System Model

The system model that we are considering to model attacks in an NFC is shown in Figure 4.1 and comprises the following entities:

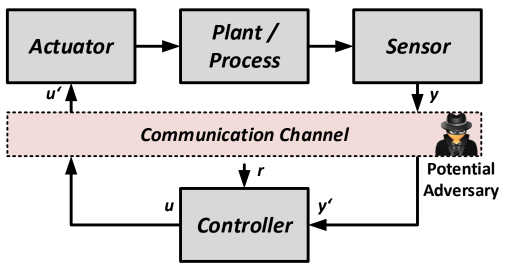


Figure 4.1: System model considered for modeling deception attacks in an NCS.

Plant / Process. The plant or process that is controlled by this NCS.

Sensor. The sensor that is used for monitoring the plant's or physical process' controlled output. The measurements of the sensor are denoted as y.

Controller. The controller generates an actuation signal u that is based on the received sensor measurements y' and the reference input r.

Actuator. The actuator receives the actuation signal u' and based on this signal influences the plant or physical process.

Communication channel. The communication channel that is used to transfer controlled plant output y from the plant or physical process to the controller and the actuating signal u from the controller to the actuator. We do not make any assumption about the type of communication channel that is used. However, we assume the presence of an adversary that can eavesdrop or modify the transferred signals such that $y' \neq y$ and $u' \neq u$.

For evaluating the proposed methods, we implemented the system model shown in Figure 4.1 in MATLAB/Simulink using the TrueTime toolbox [173] that is capable of simulating networks including attributes such as delays and bit errors. We modelled our network as a TCP network without bit errors and a delay of $10\,\mathrm{ms}$ per network segment which is tolerated by the used control algorithm. As a process we decided to use a simple DC motor that can be modelled by the transfer function given in (4.1). The transfer function of such a DC motor only depends on the torque constant in the numerator (which we chose as 1000), and thus, is very simple.

$$G(s) = \frac{1000}{s(s+1)} \tag{4.1}$$

The controlled output of this process is the angular position of the DC servo. To control the DC motor, a Proportional-Derivative (PD) controller [174] is used. Both the process model and the tuned PD controller can be found in TrueTime's examples. Figure 4.2 shows the modeled NCS.

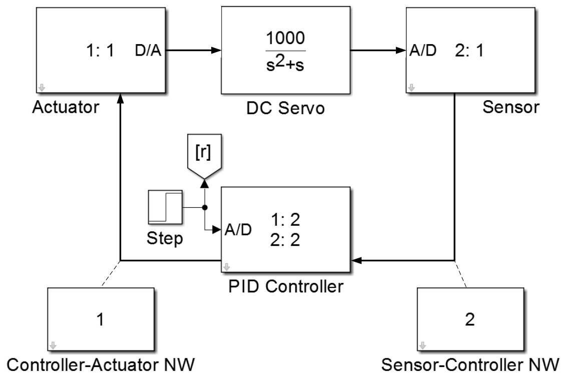


Figure 4.2: Implemented NCS system model with network segments.

4.1.2 Issues of Encryption as a Countermeasure

To protect data in an NCS that is sent over the insecure communication channel, encryption is often suggested as a potential countermeasure in literature (e.g., [175, 176]). However, simply encrypting data either using homomorphic encryption or non-homomorphic encryption in an NCS may lead to the same or even more severe problems as sending plaintext data.

Homomorphic encryption. In homomorphic encryption systems, computations can be performed on the ciphertext with the decrypted plaintext matching the performed computations [177]. For example, the ciphertext of an encrypted numeric value can be doubled. The resulting plaintext after decryption then would contain the doubled initial numeric value. For an NCS, this behaviour might be beneficial for performance reasons, but highly unfavorable from a security point of view. As an example, an adversary might be able to capture some sensor value, and add this value to any subsequent packet that is sent over the network, thus, provoking malicious system behavior.

Non-homomorphic encryption. If non-homomorphic encryption schemes such as AES are applied on sensor data, no deterministic mathematical operation can be performed on encrypted values. However, if only data confidentiality is protected without integrity checks, changing *one bit* in a data packet may lead to severe malfunctions of the system. Table 4.1 highlights this issue where the last bit of a ciphertext CT is flipped such that if the resulting ciphertext CT' is decrypted, unusable plaintext data PT' is obtained. That is, an adversary only needs to be capable of switching single bits to perform powerful attacks on NCSs.

Table 4.1: Plaintext (PT), cyphertext (CT), corrupted cyphertext (CT'), and resulting plaintext (PT').

	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 4**
PT	`0x00000001`	`0x00000002`	`0x00000003`	`0x00000004`
CT	`0xDE154CCE`	`0x18E65A6E`	`0xBD9A0593`	`0xE1B8250`$\boxed{7}$
CT'	`0xDE154CCE`	`0x18E65A6E`	`0xBD9A0593`	`0xE1B8250`$\boxed{6}$
PT'	`0x2D3DB30D`	`0xE89541F5`	`0x9AFD9AED`	`0x03BD8985`

4.1.2.1 Protecting Confidentiality and Integrity of Data

Due to the issues of only protecting data confidentiality by using homomorphic and non-homomorphic encryption, a naive approach would be to apply methods that can protect data confidentiality *as well as integrity*. For instance, AE could be applied in such a system. Using this approach, malicious packets where an arbitrary number of bits have been modified can be detected. The system then could simply drop these packets, since certain control algorithms such as the used PD controller are robust against some lost packets. However, this may also impact the systems stability as shown in Figure 4.3 where a 25 % packet loss leads to the system requiring longer to reach the desired reference value. In Figure 4.4, the system never stabilizes at the desired reference value due to a packet loss of 50 %.

4.1.2.2 Bit-Flip Attacks on protected NCSs

Due to the above demonstrated stability issues of NCS caused by dropping maliciously modified packets, a new type of attacks can be found. This is based on the observation that it is relatively easy to flip bits in an electronic system [178] which may entail severe consequences (e.g., [179]). Instead of completely shutting down a system by running DoS attacks, an adversary may just be interested in causing system misbehaviour as demonstrated in the step responses shown in Figure 4.3 and Figure 4.4. We denote these type of attacks as *bit-flip attacks*. To demonstrate the practicability of this attack, Figure 4.5 demonstrates the Packet Error Rate (PER) that can be

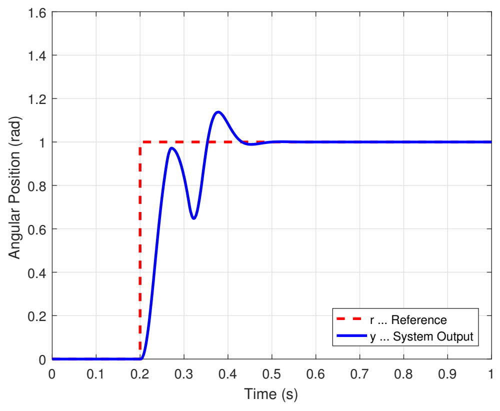


Figure 4.3: Step response 25% packet loss.

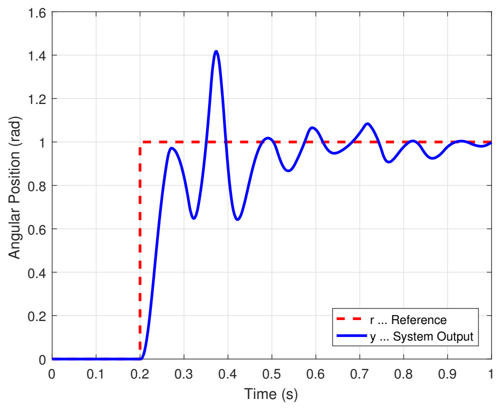


Figure 4.4: Step response 50% packet loss.

achieved by an adversary via certain BERs. As shown, BER of 10^{-3} can already lead to PER of $50\,\%$ which as we have shown may result in highly unstable systems.

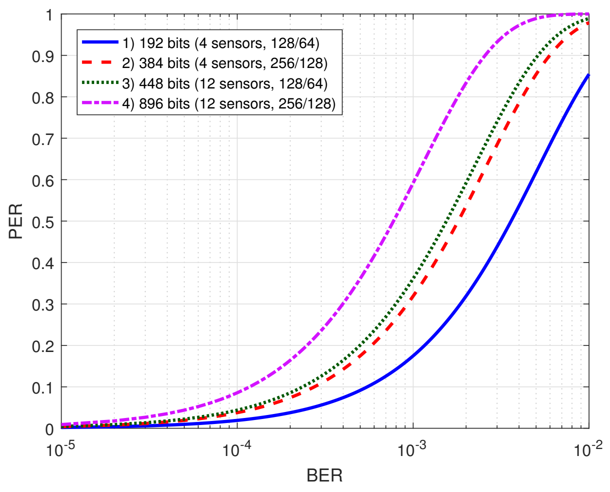


Figure 4.5: BER and resulting PER for different scenarios. Packet sizes vary due to data size (number of sensor values) and applied key lengths for encryption and MAC.

4.1.3 JEEC as a Countermeasure

To mitigate the issues caused by applying encryption in an NCS, we propose to combine encryption and Forward Error Correction (FEC), denoted as JEEC. This method is an established method in satellite communication for enhancing the security and reliability of transferred data. JEEC can be performed in a single step or sequentially. If encryption and FEC are performed in a single step [180], the performance of the system is improved compared to sequential execution of these two components [181]. However, in literature there is still a debate as to whether such single step encryption and FEC schemes provide the same level of security as conventional encryption schemes [182]. Thus, we propose to use sequential JEEC where encryption is followed by encoding the cyphertext by an FEC encoder as shown in Figure 4.6.

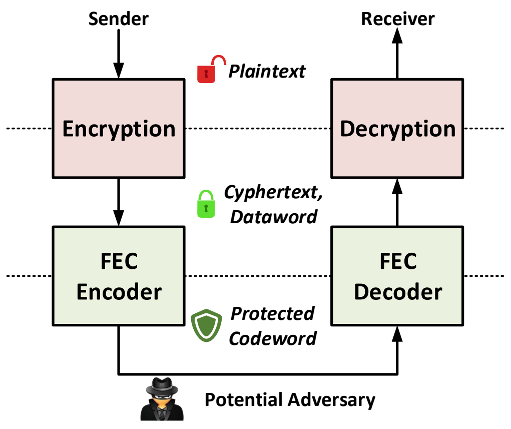


Figure 4.6: Sequential encryption and FEC.

In the studies done in this thesis, we combined AE to protect data confidentiality, integrity, and authenticity with so-called turbo codes that provide fast FEC [183]. This combination results in the following advantageous properties:

- AES is a well established encryption scheme with proven security.
- AES provides modes of operation for applying AE.
- Turbo codes provide fast FEC and can efficiently be implemented in hardware.
- The performance of turbo codes regarding BERs is close to the Shannon limit [184].

The two performance indicators of such a JEEC are (i) BER and (ii) performance. The BER of turbo codes is usually evaluated in a noisy channel which can also be seen as a communication channel with an adversary that tampers with transferred packets. For certain scenarios (non-changing Signal-to-Noise Ratio (SNR)), turbo code FEC is capable of lowering the BER by a factor of 10^4 compared to using no FEC [185]. If both AE and FEC are implemented in hardware, a deterministic delay is imposed to the system that can be handled by most control algorithms that are suitable for NCS. In our simulated system, modeling encryption and turbo codes with deterministic execution times results in the step response shown in Figure 4.7.

4.2 Sensor-Based Covert Channels

As discussed in Section 2.3, sensor-based covert channels are mostly based on two principles. Firstly, sensors are used in covert channels to observe physical properties that are triggered by the sender. And secondly, sensor data may be used in a covert channel, for instance, to hide information in the transmitted sensor measurements. In this section, we present four types of sensor-based covert channels that differ from these two principles. Before doing so, we define a system model that we use to present our attacks. After that, we demonstrate covert channels based on hiding information in unused registers or configuration bits of sensors. We then also present covert channels that are based on triggering sensor readings. These presented covert channels do

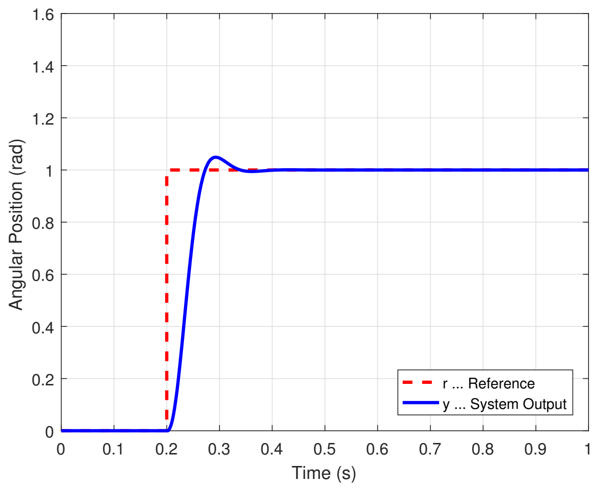


Figure 4.7: Step response of the DC motor system using JEEC to mitigate bit-flip attacks. In this scenario, no packet loss occours due to the modelled channel and the applied attack mitigation.

not only differ in the method that is used for encoding information, but also in the achievable data rate and the probability of the covert channel being detected (denoted as detectability). Figure 4.8 shows this trade-off. In addition, a red covert channel in this figure is hard to mitigate, while a green covert channel is easier to mitigate. To conclude this section, we present a sensor evaluation framework that implements hardware abstraction layers and flow control methods to easily validate whether sensors can be exploited for building covert channels. The content of this section is mainly based on work published in the paper *Sensing Danger: Exploiting Sensors to Build Covert Channels* [186].

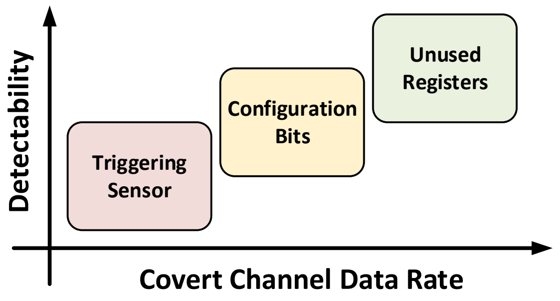


Figure 4.8: Trade-off between covert channel data rate and respective detectability.

4.2.1 System Model

We consider the system model shown in Figure 4.9 for exploiting weaknesses in sensors and to build sensor-based covert channels. This system model comprises at least two distinct processes and a shared sensor.

Process. At least two distinct processes are required in our system such that data can be transferred between these two processes using the presented covert channels. We do not make any assumption regarding the isolation between processes. That is, there might be no isolation at all or isolation techniques, such as sandboxes, virtual machines, or even different physical processors. Therefore, the two processes might be running on the same physical processor or on different processors. In any case, we assume that data exchange between these two processes might be blocked. In addition we do not make any assumption about the exact scheduling of these two processes. If, for instance, one process is scheduled more frequently than the other, the covert channel is still expected to work reliably.

Sensor. The actual sensor that is exploited for building sensor-based covert channels needs to be accessible to both involved processes. That is, process isolation must not prevent access to the sensor. We do not make any assumption regarding the type of sensor that is used in our system model. That is, the type of sensor might range from a simple temperature sensor to more complex sensors such as accelerometer and gyroscope combinations. In addition we also do not make any assumption about the technology that is used for connecting the sensor in our system. Therefore, technologies such as Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), or Ethernet should be supported.

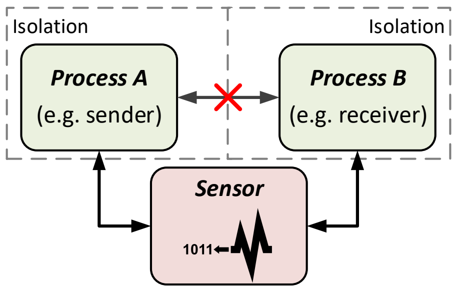


Figure 4.9: System model for building sensor-based covert channels.

4.2.2 Exploiting Unused Registers and Configuration Bits

In this section, covert channels that are based on exploiting unused sensor registers or configuration bits are presented. The presented covert channels are based on direct access to the sensor's registers. That is, read and write access for these registers is required to build the covert channels.

4.2.2.1 Unused Registers

Sensors often contain unused registers that are either reserved for future use or not required in the sensor's current mode of operation. Similar to presented network-based covert channels, these registers can be exploited by hiding information in them to transfer data to other processes.

Reserved registers. If registers are reserved but not used to publish information (e.g., status flags) or to store information such as configuration parameters, these sensor registers can be exploited for data transfer in a covert channel. Usually, these reserved registers are listed in the sensor's data sheet, and thus, easy to find. Although the data sheets often state that the registers must not be changed, they are usually still read- and writeable.

Not required registers. Registers that might be used in some modes of operation, but are unused in others, can also be exploited for data transfer in a covert channel. For instance, a sensor might define a register to set a threshold that is used when threshold monitoring is enabled. If the sensor is configured to not monitor a defined threshold, the respective register is not required by the current mode of operation. Configuration registers that define which other registers are required or not are usually readable. That is, a process may be capable of automatically finding non-required registers.

Covert Channel Design. The exploitation of unused registers facilitates a simple covert channel design. Information can simply be written into such registers by one process and read from the same register by another process. To signal successful reception of data, the register is then modified. In our approach we use the register's Most Significant Bit (MSB) as a flag; therefore, not all bits of a register can be used for data transfer.

Detectability. Although reserved registers can be written to freely, doing so might influence the sensor's functionality. Depending on the impact of writing to a certain register, covert channels that are based on exploiting unused registers could easily be detected based on misbehaving sensors.

Countermeasures. To mitigate covert channels that are exploiting unused registers, several countermeasures could be used. If write access to reserved registers is disabled, these registers cannot be exploited anymore. In addition, disabling write access to unused registers based on the sensor's current mode of operation helps mitigate the exploit action of these registers. Finally, if configuration registers are made write-only, non-required registers cannot be found in an automated way anymore. However, write-only configuration registers also complicate configuration of sensors, since sometimes either the initial value is needed or bit-wise operations such as AND, OR, and XOR would be required to update certain configuration bits without modifying others.

4.2.2.2 Configuration Bits

To configure sensors, so-called configuration registers are often used. For efficiency reasons, various configuration parameters can be combined. Also, the registers might contain reserved or unused bits that can be exploited, similar to exploiting completely unused registers.

Reserved bits. If configuration registers contain reserved bits that do not influence any configuration state, these bits can be exploited for transferring data. Similar to unused registers, the sensor's data sheet usually lists such reserved bits and often states that these bits must not be changed to avoid unwanted sensor behavior.

LSBs of configuration values. Numerical configuration values such as thresholds may be exploited for stealthy data transfer. For instance, the LSBs of a threshold value can be used to transfer data, similarly to exploiting header fields in network-based covert channels. If selected carefully, manipulating these configuration values has a negligible impact on the sensor's functionality.

Unused configuration bits. If the number of available configuration options is smaller than the maximum number of options that can be represented by the respective part of a configuration register, these bits can also be exploited for transferring data. For instance, many sensors have three modes of operation: shutdown (`0b00`), single shot (`0b01`), and continuous mode (`0b10`). For these three modes of operation, two bits are required, and thus, one bit can be changed without influencing the sensor's behavior if it is configured to operate in continuous sensing mode.

Covert Channel Design. Exploiting configuration bits to build covert channels follows the same principles as exploiting unused registers. First, exploitable configuration bits need to be determined. After that, these bits are used to transfer data in a covert channel. One process uses the selected bits to write information, while the other process reads the information from the respective parts of the selected registers. The receiver then confirms successful reception of data by modifying a certain value. That is, one bit is required as status flag to synchronize both involved processes. As a consequence, at least two configuration bits are required to build a covert channel based on exploiting such bits. Building such a covert channel with only one bit would require synchronization between sender and receiver which we assume to be infeasible. Compared to covert channels that exploit whole registers, these covert channels offer lower data rates.

Detectability. Similar to exploiting unused registers, the detectability of covert channels based on exploiting configuration bits also depends on the impact of the bits that are manipulated. For instance, a covert channel that only toggles unused configuration bits will have no impact on the sensor's functionality and might even be mistaken for a badly programmed instead of malicious process. That is, on the one hand covert channels based on exploiting configuration bits are harder to detect, but on the other hand also offer lower data rates compared to covert channels based on exploiting unused registers.

Countermeasures. There are also various potential countermeasures that could help mitigate covert channels based on exploiting configuration bits. Similar to disabling write access to reserved registers, write access to reserved configuration bits can also be disabled, thus mitigating covert channels that exploit this type of configuration bits. If configuration registers are made write only as we proposed previously, the recipient of transferred data is not able to read the information anymore. Therefore, this countermeasure also helps mitigate covert channels based on exploiting configuration bits. However, as mentioned earlier, bit-wise operations will be required for configuration registers to allow changing selected bits without modifying others.

4.2.3 Exploiting the Triggering of Sensors

Both previously discussed exploits (based on unused registers and on configuration bits) require read and write access to the same sensor. Due to this, countermeasures can easily be implemented to mitigate these issues. However, even if such countermeasures are implemented at a sensor, it is still possible to build sensor-based covert channels based on the following two methods.

4.2.3.1 Triggering Simple Sensors

If one process is capable of triggering events that update read-only registers, these registers can be exploited to build sensor-based covert channels. For instance, most sensors include a register where status flags indicate a finished sensing process. That is, one process may trigger the sensor, while the other process is monitoring these status flags. Information in such a scenario can be encoded in multiple ways.

Timing differences. Information can be encoded in the timing intervals between two sensor readings. For example, a binary `'1'` could be transferred by triggering two sensor readings with an interval of 100 ms. A binary `'0'` would then be transferred by triggering the sensor with a different interval, for instance, 10 ms. The process that should receive the data then needs to observe the respective status bits that indicate a finished sensing process and measure the timing intervals to decode the corresponding hidden data. However, this approach has the drawback that the process that needs to monitor the status bits needs to poll this information with a high frequency to provide accurate timings. In addition, both involved processes need to be synchronized since there is no possibility of confirming successful reception of data.

Direct encoding. If there is more than one status bit that can be triggered, information can be directly encoded using the available status bits. For example, if a sensor is capable of sensing more than one physical property, multiple status flags will also be present. That is, one status bit can be used for transferring data, while the other status bit can be used as a control flag. If all status flags would be used for data transfer, the two states were no status flag is set due to the sensing being still in progress and the data word containing only zeros could not be distinguished. Thus, we propose to use two status bits for transferring 1 bit of information and for synchronizing both involved processes.

Covert Channel Design. Due to the easier possibility of synchronizing both involved processes, we propose to use direct encoding in status bits. For simplicity reasons we are going to discuss the covert channel design for transmitting 1 bit using 2 status flags. In its default setting, both status flags are set to '0', indicating that no sensor reading is available. A '1' indicates that the respective sensor reading is available. The status flag is reset to '0' again by the sensor if the value is read. In our proposed covert channel design, the process that wants to transmit data simultaneously triggers both sensors, thus setting both status flags to '1'. Immediately after the sensor finished the sensing process, the process wanting to send data reads one sensor value, and thus, resets the respective status flag. The receiving process monitors the status flags. Upon reading that both status flags are set to '1', the process is informed that a data bit is following. After then reading the respective information, the receiving process resets the second status bit by reading the remaining sensor value. The sending process is thus informed that information was read by the other process. Table 4.2 summarizes the states that can be implemented using 2 status bits. As shown in this table, transferring 1-bit words results in clearly distinguishable states whereas using 2-bit words results in ambiguous states for the status bit values '00' and '11'.

Detectability. Compared to writing information directly into a sensor's register, covert channels that are based on triggering sensor readings are harder to detect. In this covert channel design,

Table 4.2: Status flags for two sensors and the available states for a 2-bit word and a 1-bit word respectively.

S1	**S2**	**2-bit word**	**1-bit word**
0	0	’00’ *and* no data	no data
0	1	’01’	’0’
1	0	’10’	’1’
1	1	’11’ *and* sensor ready	sensor ready

only the process that wants to transmit information is triggering the sensor, while the sending and receiving processes are reading sensor values and status flags. Therefore, a system monitor might not detect malicious behavior such as two processes alternately changing a sensor register’s value. In addition, if the roles of sender and receiver are switched regularly for bidirectional communication, the system behavior is comparable to two processes that alternately trigger sensors, read the respective sensor values, and poll status flags for determining the sensor’s availability.

Countermeasures. Mitigating covert channels that are based on triggering sensors is more complex compared to mitigating covert channels that require write access to sensor registers. In principle, any link between triggering a sensor and information that can be observed by other processes needs to be removed. Thus, to mitigate covert channels that are based on triggering sensors, a sensor management instance is required that encapsulates sensor access. For example, Android’s sensor manager only allows processes to register for sensor data. The sensor manager then determines the required sensor configuration such that all registered processes can be served. Whenever a new sensor reading is available, all registered processes are then notified using interrupts. Therefore, a managed sensor approach that removes any status flag dedicated to indicating available sensor readings mitigates the presented covert channel.

4.2.3.2 Triggering Managed Sensors

If the previously discussed countermeasure of managing sensors is badly implemented, sensor-based covert channels can still be build. Usually in a managed sensor environment, processes need to register for sensor readings. Android, for instance, uses such a managed sensor approach [187] where registered processes are notified of new sensor readings via interrupts. When registering for a sensor reading in Android, the process needs to specify the desired sampling period as a parameter. However, as also stated in Android’s documentation, this sampling period is only a suggested delay that might be altered by other applications. Since the altered sampling frequency can easily be detected by one process and triggered by another, a covert channel can be built exploiting this functionality.

Covert Channel Design. Based on the observation that one process is able to influence the sampling period of another process, a frequency encoded covert channel can be built that exploits sensor managers that are built like Android’s implementation. To transmit data, the current sensor reading frequency is monitored by both processes. The process that wants to transmit data then registers to the sensor manager with a lower frequency that can be observed by the other process. If two sensor reading frequencies below the previous sensor reading frequency can be found, binary values '0' and '1' can be frequency encoded, similarly to a Frequency Shift Keying (FSK) encoding. Figure 4.10 shows an example where a binary sequence is encoded using different sensor reading frequencies. Contrary to the other three presented covert channels, the process that receives data cannot easily acknowledge the successful reception of data in this approach. Thus, after sending its data, the process that transmits data needs to un-register from the sensor manager and thereby hand over the sender role to the other process that can then acknowledge the received data.

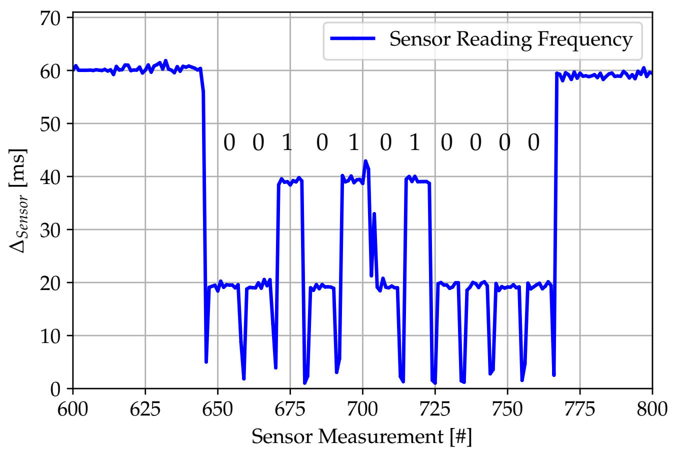


Figure 4.10: Information encoded in different sensor reading frequencies. These measurements were obtained using the ambient light sensor on a OnePlus5 running Android 8.0.

Detectability. The switching of sensor reading frequencies due to other processes registering and un-registering listeners is expected system behavior. For example, Android expects processes to un-register from sensors if the application is minimized. Thus, covert channels that exploit managed sensors using this approach cannot be easily detected. However, we think that if sensor access is audited, malicious access patterns could be detected if the auditing tool is trained sufficiently. Static code analysis tools such as the popular FlowDroid [188] for Android applications are however currently not capable of detecting our presented covert channels.

Countermeasures. To mitigate the presented covert channel that exploits managed sensors, changes to how sensor reading frequencies are handled need to be implemented. One possible countermeasure would be to disallow arbitrary frequencies and only support a set of well-defined values. These values need to be selected in a way that they are multiples of each other, for example, 10 ms, 20 ms, 40 ms, and so on. By defining such sensor reading frequencies, the sensor internally can be operated at the lowest selected frequency, while each process receives sensor values with its actual defined sensor reading frequencies.

4.2.4 Sensor Evaluation Framework

To facilitate easy vulnerability testing of sensors, we implemented a modular test framework. This framework is structured into the following four abstraction layers and is shown in Figure 4.11:

1. The lowest layer (access abstraction) implements sensor access via various technologies.
2. The sensor abstraction layer implements sensor-specific aspects such as register mappings.
3. All presented exploits are implemented in the framework's exploit abstraction layer.
4. Building blocks that are relevant for the covert channel's reliability are implemented in the covert channel abstraction layer. In this section, these building blocks are briefly presented.

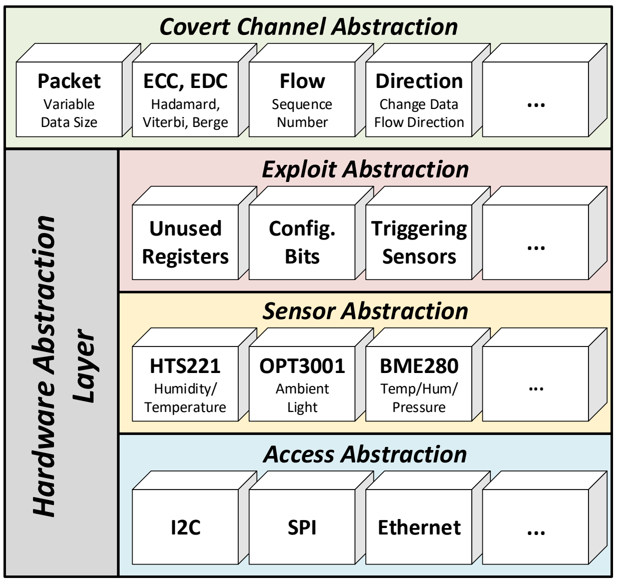


Figure 4.11: Sensor evaluation framework with different abstraction layers.

4.2.4.1 Need for Error Detection and Correction

All presented covert channels are based on information that is transferred using sensors. As other processes (other than the processes that are transmitting and receiving data) might also access the sensor, information might be lost. For instance, other processes might also trigger sensor readings. Therefore, these sensor-based covert channels need to be considered a noisy channel. According to Shannon [189], an error-free data transmission over a noisy channel can be achieved if transferred data is sufficiently encoded using appropriate encoding schemes. Thus, in our proposed covert channels we used Error Correcting Codes as well as Error Detecting Codes (EDCs).

4.2.4.2 Packet Structure and Flow

In our proposed covert channels we use two types of packets, Request (REQ) packets and Response (RES) packets. Figure 4.12 shows the structure of both packet types.

REQ packet. The REQ packet only contains a Sequence Number (SQN) encoded by a Hadamard error correcting code. In general, such a code encodes a k-bit message in a 2^k-bit codeword. Due to the exponential growth of packet size, we use a 2-bit SQN which results in a total packet size of 4 bits. The Hadamard coding scheme is proven optimal for $k \leq 7$ [190] and can recover errors as long as less than half of the bits are flipped.

RES packet. All RES packets contain a type bit that classifies each packet as data or command packet. Commands include, for instance, functionality to stop data transmission or to reverse data direction. Similarly to REQ packets, RES packets also contain a 2-bit SQN. In addition, each packet contains either data- or command-specific information. Instead of using error correcting codes, we add EDCs to RES messages due to the previously discussed exponential growth in size. We propose using a Berger EDC [191] that supports checking a maximum of $n = 2^k - 1$ bits of information with a k-bit code. Transmission errors are thus not corrected but detected and handled by our proposed communication flow approach.

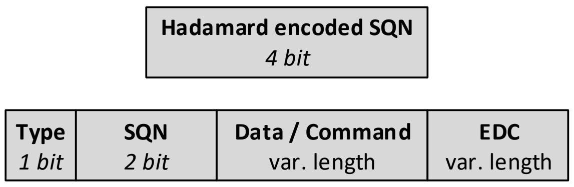


Figure 4.12: REQ packet (top) and RES packet (bottom) structures, respectively.

Data flow. Communication flow in our proposed covert channels is organized as a request-response mechanism. One successful request-response round trip comprises the reception of a REQ packet by the sender that is then answered by sending an RES packet. This approach is similar to the HyperText Transfer Protocol (HTTP), where actual data is always transferred in a response packet. To manage communication flow, both packet types contain an SQN that is used to identify matching REQ and RES packets. The receiver is responsible for increasing the SQN after successfully receiving the desired RES packet. By repeatedly sending packets with the same SQN, receiver and sender can indicate that the expected packet was not yet received and a re-transmission is desired. The proposed data flow mechanism is shown in Figure 4.13. Re-transmissions can be caused in the following three scenarios:

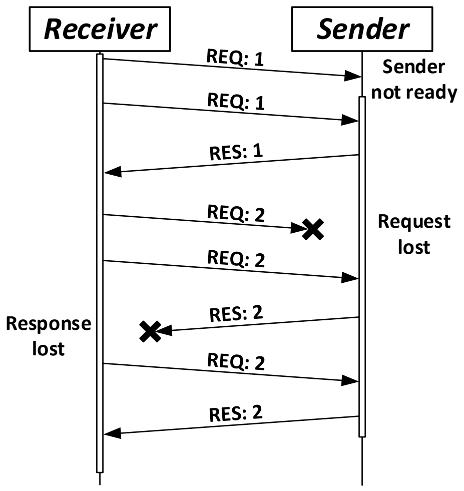


Figure 4.13: Data flow principle that is used in our proposed covert channels.

Sender not ready. When a covert channel is established, the sender might not be ready to transmit the requested data and no RES packet is thus sent as answer to the receiver's initial REQ packet. The receiver continuously transmits its initial REQ packet until a covert channel is successfully established due to the sender's response.

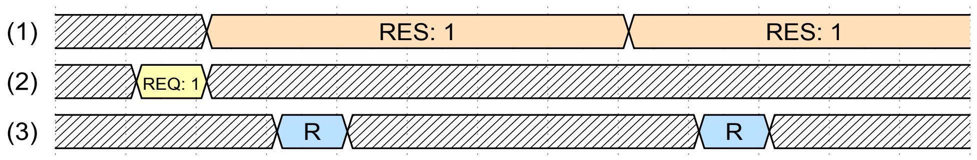
Request lost. A REQ packet may be lost while being transmitted due to a noisy covert channel. As in the first case, the receiver continuously transmits its REQ packet until a RES packet is successfully received from the sender.

Response lost. Similarly to REQ packets, RES packets can also be lost while being transmitted. In this case, the sender receives multiple REQ packets with the same SQN, recognizes that its packet must have been lost and re-transmits the same RES packet again.

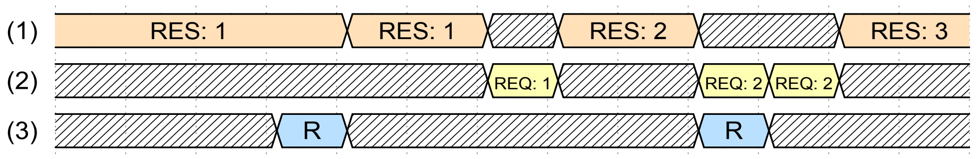
4.2.4.3 Bidirectional Communication and Adaptive Packet Length.

As already briefly mentioned, our proposed packet structure supports sending command packets. The presented framework supports commands to stop communication and reverse the data direction. That is, by switching sender and receiver roles, a fully bidirectional communication is supported by our proposed covert channels. In addition to these two commands, two commands to increase and decrease the payload data size are also supported. This method is used to improve covert channel reliability compared to static payload sizes.

Depending on the amount of accesses to the sensor that is used in a covert channel, an optimal payload size may be found. If too large packets are sent and other processes thus invalidate the sent packets, reliable data transfer might not be possible. If the RES packet size is set too small, the achievable covert channel data rate will be very low due to the imposed overhead. This issue is especially relevant if other processes access the sensor infrequently and no fixed interval of no other process accessing the sensor can thus be found. Therefore, we propose to use dynamic packet sizes, which allows decreasing the RES packet size whenever errors are detected by the EDC. Reducing the packet size will lead to fewer re-transmissions, but generally higher protocol overhead. Thus, packet size is increased if a certain amount of successfully transferred RES packets is reached. Figure 4.14 demonstrates this principle based on a simple example.



(a) Static RES packet size.



(b) Dynamic RES packet size.

Figure 4.14: Static and dynamic payload sizes. (1) Sender and (2) receiver access the sensor together with (3) another process that is reading the sensor (R).

5 Secured Sensor Configuration

Based on the attacks presented in Chapter 4 we conclude that sensors offer a variety of potential exploitable interfaces. We especially identified a sensor's configuration interface as a main weakness as highlighted by our presented covert channels. Therefore, in this chapter, we present methods to improve a sensor's security by facilitating a secured sensor configuration process. We first argue why NFC is used as communication technology based on the requirements listed in Section 1.3.2. We then present the system model we used for implementing the presented secured sensor configuration approach. After that, the secured smart sensor configuration approach is presented, followed by a method for attesting correct configuration states. We then present a method for password-free authentication prior to configuration updates. Each of these sections builds upon each other (see Figure 5.1), and thus, an architecture and framework for the secured configuration of smart sensors is presented. Additionally, a generalized and secured transport-layer protocol for NFC is presented that can be used for secured smart sensor configuration as well as for other NFC-related tasks that require a secured, yet efficient, NFC-based data transfer.

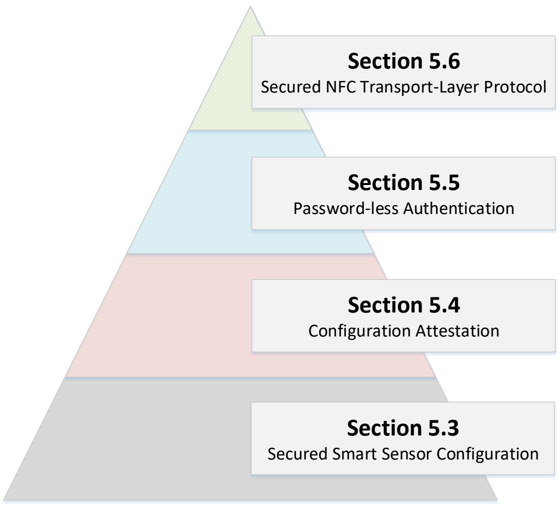

Section 5.6
Secured NFC Transport-Layer Protocol
Section 5.5
Password-less Authentication
Section 5.4
Configuration Attestation
Section 5.3
Secured Smart Sensor Configuration

Figure 5.1: Bottom-up relationship between sections in this chapter.

5.1 Selecting NFC as a Configuration Interface

The proposed smart-sensor configuration approach that is presented in the following sections is based on NFC. This section lists possible alternatives and discusses the advantages and drawbacks of these alternatives. Also, each technology is analyzed regarding its compatibility with the requirements for secured sensor configuration that were listed in Section 1.3.2. Table 5.1 summarizes this list and gives an overview of the requirements that are met by each technology.

Dedicated wired configuration interface. A wired configuration interface allows configuration data to be transferred to the smart sensor as well as powering the device during the configuration update process. In general, wired interfaces also provide higher data rates and higher reliability than wireless interfaces. However, a wired interface would require the smart sensor to be connected during manufacturing and by the customer each time a configuration update needs to be applied.

RF Coil on Chip (CoC) configuration interface. An RF CoC configuration interface would allow the smart sensor to be configured wireless during manufacturing as well as by the customer. Not only can such an interface be used to transfer data, but also to power the chip during the configuration update process [192]. However, a dedicated device for communicating with such an RF CoC configuration interface is required which would cause additional costs for customers. In addition, due to the relatively small area of the antenna, the sender and receiver must be positioned precisely to allow for data transfer which does not go hand in hand with high usability.

RF Coil on Module (CoM) configuration interface. An RF CoM configuration interface is very similar to an RF CoC configuration interface in terms of its properties. Due to the larger area of the antenna, data transfer may be possible over larger distances and the positioning of the sender and receiver does not have to be as precise as in the RF CoC case.

RFID interface. Due to its nature, an RFID interface may be used for configuration purposes as well as, for example, for locating smart sensors. Thus, such an interface may provide higher value due to its flexibility. Similarly to RF CoC and CoM configuration interfaces, RFID also allows data as well as power to be wirelessly transferred. Compared to the previous two wireless interfaces, RFID requires a larger antenna area which allows successful communication over larger distances and makes positioning the sender and receiver easier. Ultra-High Frequency (UHF) RFID in particular supports distances of up to $100\,\mathrm{m}$ which makes the approach highly practical. In addition, many modern smartphones are capable of communicating with RFID tags and RFID readers that provide higher transmitting power are available OTS. In RFID, the roles of reader and tag are fixed, and thus, bidirectional communication is only partly supported.

NFC interface. As discussed in Section 2.6, NFC can be seen as an extension to RFID that fully supports bidirectional communication in the so-called peer-to-peer mode. Similarly to the previously discussed wireless technologies, NFC also allows data as well as power to be wirelessly transferred. The maximum communication range of NFC usually is limited to a couple of centimeters, which makes positioning sender and receiver sufficiently easy while minimizing negative effects due to interference with other devices and possible attacks due to the high communication range. Also similarly to RFID, the NFC interface may also be used for other purposes than configuring the smart sensor, which makes it also more valuable due to its flexibility.

Bluetooth (BT) / ZigBee (ZB) interface. Technologies such as BT or ZB are usually considered more powerful than the previously discussed wireless technologies. These technologies offer high bandwidths as well as high communication ranges and allow bidirectional data transfer. Also, almost all modern smartphones include a BT interface which would allow customers to rely on existing OTS hardware. However, power transfer over these technologies is not possible, and thus, the device for which a configuration update needs to be transferred needs to be connected to a power supply. This may be practical during the smart sensor's normal operation, but is inconvenient during manufacturing.

Wi-Fi interface. Wi-Fi generally is considered the most powerful of the mentioned wireless technologies. It provides the highest bandwidth and modern smartphones as well as laptops include Wi-Fi interfaces which would allow customers to use their existing OTS hardware for configuration updates. However, similarly to Bluetooth and ZigBee, Wi-Fi also does not support wireless power transfer, which makes the approach impractical for configuring the smart sensor during manufacturing. In addition, Wi-Fi requires higher amounts of energy compared to the other discussed wireless technologies, which might also make this technology infeasible for resource-constrained smart sensors.

Table 5.1: Comparison of possible communication technologies for smart sensor configuration.

	Req1	Req2	Req3	Req4	Req5	Req6	Comment
Wired	✓	✗	✓	✗	✗	✗	Requires additional HW
CoC	✓	✗	✓	✗	✓	✗	Requires additional HW, hard to use
CoM	✓	✗	✓	✗	✓	✗	Requires additional HW, hard to use
RFID	✓	✓	✓	✓	✓	✓	Range and unidirectional communication
NFC	✓	✓	✓	✓	✓	✓	Application specific security
BT/ZB	✓	✓	✓	✓	✗	✓	Require power supply
Wi-Fi	✓	✓	✗	✓	✗	✓	Requires power supply, not efficient

5.2 System Model

The comparison of possible communication technologies in Table 5.1 highlights that NFC is very well suited for a smart sensor configuration approach, based on the requirements listed in Section 1.3.2. However, we have to note that NFC as a technology alone does not completely fulfill Req4 that also requires the configuration approach to scale well to large numbers of devices. Thus, we propose the following system model for an NFC-based configuration approach to account for Req4 and also to implement security-related mechanisms that will be presented in later sections of this chapter. The system model is shown in Figure 5.2 and comprises the following three entities:

Smart sensors. The smart sensors that should be configured using the secured smart sensor configuration approach that is presented in this thesis. We do not assume any limitation on the number of smart sensors that can be managed by our approach. Thus, the approach should scale to an arbitrary number of smart sensors. The smart sensor itself may be placed in an uncontrollable environment such as outside of buildings. Therefore, in this thesis we assume the presence of adversaries that may be able to gain physical access to these smart sensors. To protect the smart sensor against these adversaries, the smart sensor model presented in Section 5.2.1 is used.

Mobile configuration device. The mobile configuration device is used to transfer configuration data to the smart sensors. To allow the system to scale well to large numbers of smart sensors, we also do not assume any limitation in the number of existing mobile configuration devices. In the approach presented in this thesis, OTS devices such as smartphones should be used as mobile configuration devices, and thus, these devices need to be considered as untrustworthy. Therefore, any configuration data that is transferred by these devices needs to be protected such that no misuse by an untrusted mobile configuration device is possible.

Configuration back-end. The system model includes exactly *one* so-called configuration back-end. This entity is responsible for managing all configurations that are currently applied to the smart sensors. The configuration back-end is considered sufficiently secured in this thesis, and thus, as a trusted entity in our system model. The protection of the back-end itself is considered out of scope for this thesis.

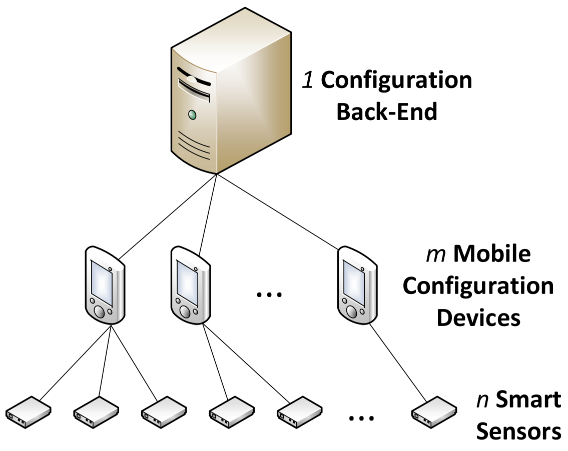


Figure 5.2: System model used in the presented secured smart sensor configuration approach.

5.2.1 Extended Smart Sensor Model

To mitigate attacks that are enabled by adversaries that may be able to gain physical access to the smart sensor, the smart sensor model that is shown in Figure 1.1 is extended by a so-called Security Controller (SC). This extended smart sensor model is depicted in Figure 5.3. As can be seen there, the SC is connected to the general-purpose microcontroller. All security-related operations and storage of confidential information is performed by the SC. All other operations such as data acquisition or aggregation are performed by the more powerful general-purpose microcontroller. This principle of splitting storage and execution environment into a *normal* and *protected* world is often referred to as security by isolation [46]. The SC provides the following three main functionalities to the overall system functionality:

NFC interface. To facilitate NFC-based configuration updates, the SC adds NFC functionality to the smart sensor. This NFC interface is not only used for data transfer, but is also capable of powering the SC if an NFC reader is put into close proximity. The NFC interface only acts as a tag, and thus, is considered a passive NFC device that is not capable of communicating with other passive NFC devices. Data that is sent via this interface is processed by the SC.

Protected storage. The SC provides tamper-resistant storage that is used for storing any confidential information such as encryption keys. Even if an adversary is able to gain physical access to the device, the adversary is efficiently hindered from extracting any information by attacks such as side-channel attacks.

Protected execution. All security-relevant operations are executed in the SC's protected execution environment. Similarly to the protected storage, the protected execution environment also is tamper-resistant. That is, even physical attacks such as side-channel attacks that may be used to extract keys are efficiently mitigated.

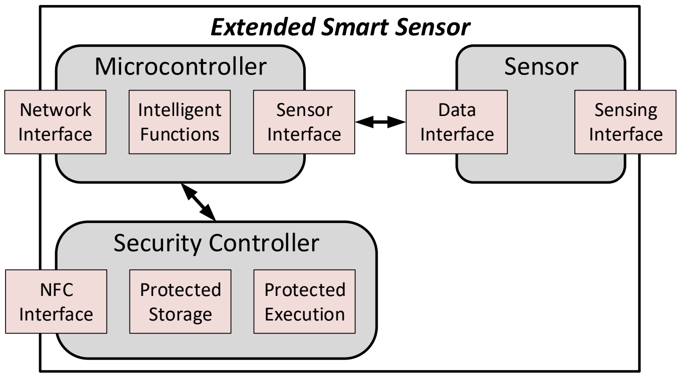


Figure 5.3: Block diagram of extended smart sensor model.

5.3 Secured Smart Sensor Configuration

For applying configuration updates to smart sensors in a secured way using a mobile configuration device and NFC as communication technology, the general system model and the extended smart sensor model already presented in Section 5.2 are used. This section presents the implemented protocol and the respective security measures that were taken to protect the confidentiality, integrity, and authenticity of transferred configuration data. The content of this section is mainly based on work published in the paper *Secured and Easy-to-Use NFC-Based Device Configuration for the Internet of Things* [193].

To evaluate the presented approaches, the following three requirements regarding usability, security, and performance are defined:

1. **Usability.** The usability of our presented approach must be measurably better than the usability of other configuration approaches. Instead of user studies, a comparison to state-of-the-art configuration methods should be used to highlight the improved usability.
2. **Security.** A threat analysis should be performed to highlight security issues of unprotected configuration interfaces. Our presented approach should mitigate the identified security issues. Residual risks should only remain for issues that are infeasible to mitigate.
3. **Performance.** The performance of our presented approach must be evaluated based on the

overall time that is consumed when applying a configuration update. The time should not exceed $1\,\mathrm{s}$ for a reasonable configuration size such as $1\,\mathrm{kB}$.

5.3.1 Untrustworthy Mobile Configuration Device

As already mentioned in Section 5.2, any OTS smartphone that offers NFC capability should be usable as a mobile configuration device. Since the integrity of software running on such a device cannot be easily verified, we consider the device untrustworthy. As a consequence, configuration data should not be readable on the mobile configuration device but rather should be transferred end-to-end protected from the configuration back-end to the smart sensor. That is, the mobile configuration device acts as a gateway between configuration back-end and smart sensor.

5.3.2 QR / NFC Hybrid Approach

Since NFC is used for the smart sensor's configuration interface, configuration data from the mobile configuration device to the smart sensor is transferred using NFC. However, transferring data from the configuration back-end to the mobile configuration device using the same technology would require so-called NFC readers. This may be acceptable in a setting were a large number of smart sensors are managed by our presented approach, for instance, in a company. However, for end-users a requirement for an additional NFC reader would significantly increase the cost of such a configuration approach, and thus, would lead to end-users refusing to use the presented approach. We therefore suggest a Quick-Response (QR) Code and NFC hybrid approach, as depicted in Figure 5.4. In a configuration update process, the following two phases need to be performed:

1. Configuration data is edited on the configuration back-end's User Interface (UI). After finishing the editing of configuration data, a QR code is shown by the UI which needs to be scanned by the mobile configuration device. This QR code then transfers the latest configuration data from the configuration back-end to the mobile configuration device.
2. Configuration data that is stored on the mobile configuration device can be transferred to any smart sensor that is equipped with an NFC interface.

Since the amount of data that can be encoded in a QR code is limited to a maximum of roughly $2900\,\mathrm{byte}$ [194], we propose two different modes of operation for QR-code-based data transfer.

Inline. If the configuration that should be transferred is smaller than $2900\,\mathrm{byte}$ the complete configuration can be encoded in the QR code. Therefore, we denote this mode of operation as *inline QR code*. These inline QR codes do not require an active network connection on the mobile configuration device and thus would also work on smartphones without network coverage, for example, in industrial settings.

URL. If the size of configuration data that needs to be transferred is larger than $2900\,\mathrm{byte}$ the QR code only contains an Uniform Resource Locator (URL) pointing to the configuration data on the configuration back-end from where it needs to be downloaded using a working network connections such as Wi-Fi or 3G/4G. Therefore, this type of QR code is denoted as *URL QR code*.

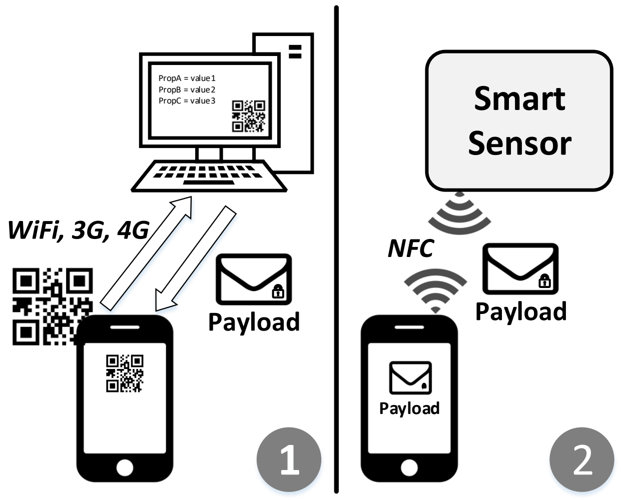


Figure 5.4: Hybrid approach used for transferring configuration data.

5.3.3 Security Measures

The extended smart sensor model that was already presented in Section 5.2.1 already highlights the increased security due to mitigating physical attacks that may target the smart sensor. However, as pointed out in Section 2.6.1, NFC communication can be eavesdropped up to a distance of $10\,\mathrm{m}$. Also, unprotected QR codes can easily be read by any device that is equipped with a camera. Since configuration data may contain confidential information, additional security measures need to be taken in order to protect configuration data that is transferred either via QR code or via NFC in our approach. To protect the confidentiality, integrity, and authenticity of configuration data, AE is used in our approach. Also, the following information is added to the encrypted configuration data:

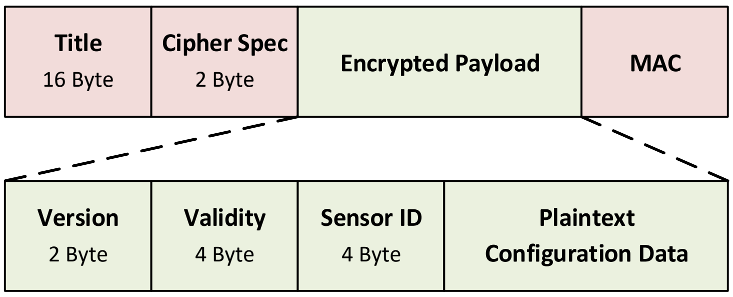


Figure 5.5: Protected configuration packet that is transferred in our approach.

Version. The version number identifies the specific configuration that should be applied. This information protects the smart sensor from replay attacks were an outdated configuration version should be applied over a newer configuration version.

Validity. Each configuration has a certain validity until which it can be applied to the smart sensor. This information protects the smart sensor from valid but never applied configuration packets that an adversary may want to apply.

Sensor ID. The sensor ID defines for which smart sensor the configuration package is intended. This protects the smart sensor from attacks where an adversary is capable of capturing a protected configuration packet and subsequently tries to apply this configuration package to a different smart sensor.

In addition to the encrypted information, the following plaintext information is also transferred:

Title. The title of the respective configuration packet. This information helps the end-user to identify the correct configuration and is only transferred from configuration back-end to mobile configuration device.

Cipher spec. The cipher spec field is used to define which encryption algorithm and which key length should be used for the AE algorithm. For example, *AES-GCM* with a key length of $256\,\mathrm{bits}$ may be configured in this field. Therefore, the cipher spec information needs to be transferred to the smart sensor.

MAC. The MAC that is generated by the AE algorithm in the encrypt-then-MAC approach. It has to be noted that this differs to the publications [193, 195] on which this chapter is based. This information also needs to be transferred to the smart sensor.

Apart from the above-mentioned title information, all other data is transferred in both approaches, via QR code from configuration back-end to mobile configuration device as well as from mobile configuration device to smart sensor. That is, the confidentiality, integrity, and authenticity of configuration data is at all times protected by AE.

After transmitting a configuration package to the smart sensor, all checks that are shown in Figure 5.6 are performed. The mobile configuration device is informed whether the configuration was successfully applied or not, but no reason is given in case of error.

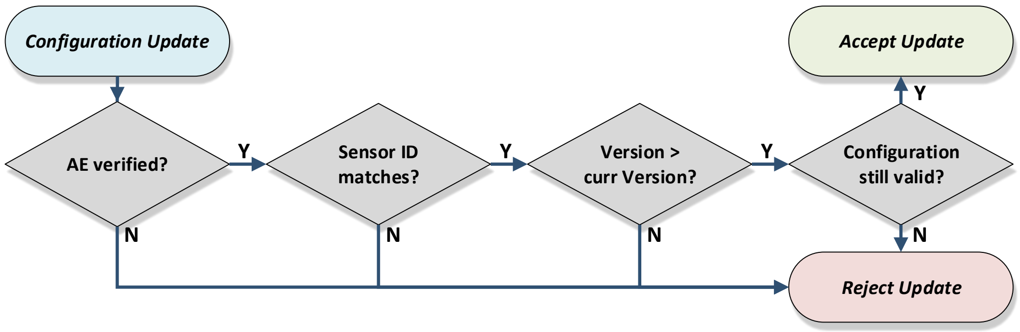


Figure 5.6: Checks that are performed before applying a transferred configuration packet.

5.4 Configuration Attestation

Although the mobile configuration device is notified of successfully applied configuration updates, no validation regarding the correct usage of configuration data is possible in the previously presented configuration approach. In addition, protected storage space is limited and non-confidential configuration parameters such as sampling frequencies could thus be stored at the general-purpose microcontroller's unprotected memory. However, this memory is usually not protected against malicious activities that might modify its content. Therefore, we suggest to extend this approach by an attestation mechanism that is presented in this section and that can verify the smart sensor's configuration data. The content of this section is mainly based on work published in the paper *Hardware-Secured Configuration and Two-Layer Attestation Architecture for Smart Sensors* [196].

Since each smart sensor is equipped with an SC for performing secured configuration updates, a trusted hardware module that can be seen as a Root of Trust (RoT) exists in each device. Additionally, in our system model (see Section 5.2) we assume that the configuration back-end is sufficiently protected against any type of attack. Therefore, the configuration back-end can also be seen as an RoT. Since the proposed system model includes two potential RoTs, we suggest denoting the smart sensor's RoT as Local RoT (LRoT) and the configuration back-end as Global RoT (GRoT), as shown in Figure 5.7. The general-purpose microcontroller of a smart sensor is denoted as *no trust zone* since neither storage nor execution environment provide the high level of protection that is offered by the tamper-resistant SC.

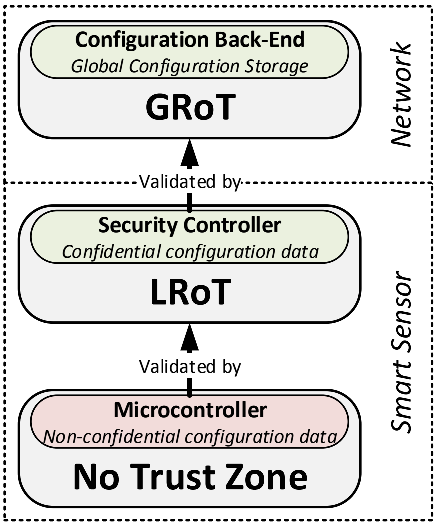


Figure 5.7: Attestation system model that contains two RoTs.

As outlined in Figure 5.7, both RoTs are used for attestation purposes. The LRoT validates the general-purpose microcontroller's memory while the GRoT is used to validate the SC's state. However, since we assume the LRoT to be a trusted entity, only a verification process will be performed instead of attestation, as will be explained later in this section. The verification is used to synchronize the configuration back-end's database with the actual applied configuration data such that the password-free authentication process that will be presented in Section 5.5 can be implemented. Further advantages of such a two-layered attestation process are as follows:

- **Overhead:** Compared to traditional remote attestation processes, the network overhead is minimized since only a reduced amount of data needs to be transferred for verifying the correctly applied configuration to the GRoT.
- **Security:** If local attestation is used, network access can be denied to malicious smart

sensor nodes. Thus, the overall security of a system can be increased. For instance, in a WSN, nodes need to transfer network traffic of untrusted nodes during a remote attestation process. During this process, the malicious node already could perform harmful activities such as compromising the routing tree of such a network. If such nodes are locally attested, nodes only need to communicate with trustworthy neighbouring nodes.

Although both RoTs perform validation tasks, the actual processes differ and thus, are discussed in detail in the following two sections.

5.4.1 Local Attestation

Non-confidential data on the general-purpose controller comprises configuration data as well as application code (binaries). Since malicious configurations and malicious code can be harmful to the smart sensor's functionality, we suggest to attest both using the LRoT. However, due to the different nature of these two components, we propose to use two different attestation techniques.

Application code. Applications (binaries) usually consist of static memory content that is changed less frequently than other memory content. According to Yang et al. [67], the memory required by binaries can usually be assumed to be two orders of magnitudes larger than memory required for other data such as configuration data. Thus, we suggest using basic binary attestation or advanced versions thereof such as pseudo-random memory traversal [65] to attest application code. Due to the large size, limited memory, and using advanced binary attestation methods, attacks such as pre-computation or memory copy attacks can be mitigated. To facilitate an easy binary attestation process, additional so-called *attestation information* needs to be included in each configuration package.

Non-confidential configuration data. Since configuration data requires less memory compared to application code, memory copy attacks would be easy to implement. Therefore, we propose to use so-called *property-based attestation* [197] for attesting non-confidential configuration data. Such an attestation method requires functionality that is currently not available in many OTS tamper-resistant components such as Trusted Platform Modules (TPMs). However, the required functionality can be implemented in the protected execution environment that is provided by the SC that we propose in our extended smart sensor model (see Section 5.2.1). To implement property-based attestation, certificate information is required which can also be included in configuration packages that are used when applying configuration updates.

Both of these approaches are then jointly used by the LRoT to decide if network access is *granted* or *denied* to the respective smart sensor node. Denying network access can be achieved by protecting information such as the network stack with methods such as so-called *sealed-storage*.

5.4.2 Global Attestation

Confidential configuration data is stored in the SC's protected storage and also protected during transport by the security measures that are presentd in Section 5.3. Therefore, the correctness of such configuration parameters can be assumed in our presented architecture and does thus not need to be attested. However, the correctness of confidential configuration data is still checked to synchronize the configuration back-end's database with the actually applied configurations. For instance, if a malicious service technician does not apply security updates to any or some selected sensors, the GRoT is capable of detecting such behavior. This attestation step can either also use property-based attestation or, to limit the overhead of transferred data, only report the applied configuration version to the configuration back-end. A functioning synchronization between smart sensor configuration and the configuration back-end is then crucial for the functionality of the password-free authentication that is presented in Section 5.5.

An overview of the validated memory content and the applied validation techniques of both, the LRoT and GRoT, are shown in Figure 5.8. This figure also highlights the two-layered attestation approach and the result both attestation layers entail.

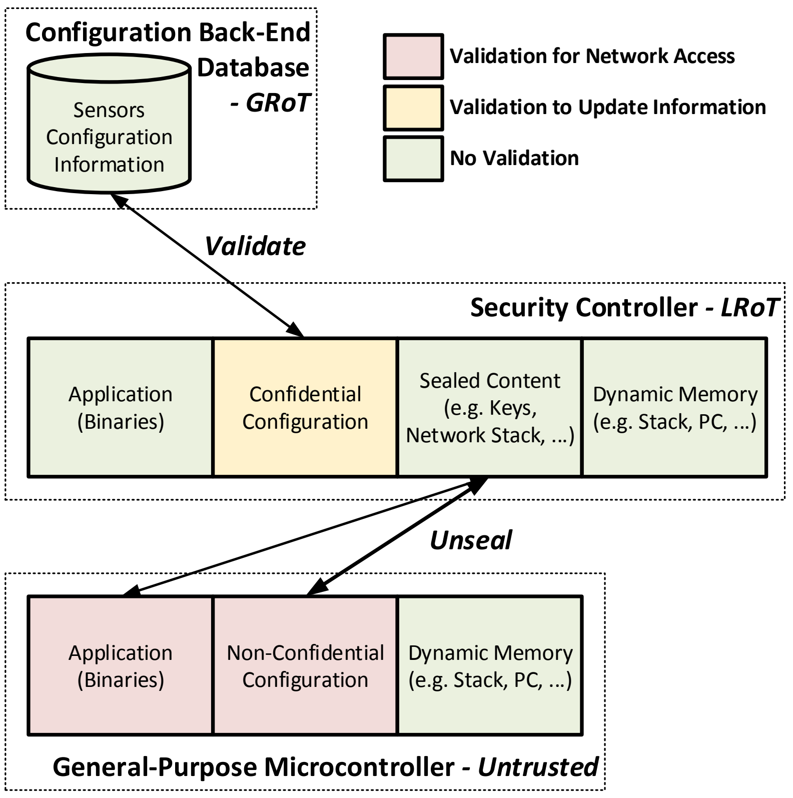

Configuration Back-End Database - GRoT
Sensors Configuration Information
Validation for Network Access
Validation to Update Information
No Validation
Validate
Security Controller - LRoT
Application (Binaries)
Confidential Configuration
Sealed Content (e.g. Keys, Network Stack, ...)
Dynamic Memory (e.g. Stack, PC, ...)
Unseal
Application (Binaries)
Non-Confidential Configuration
Dynamic Memory (e.g. Stack, PC, ...)
General-Purpose Microcontroller - Untrusted

Figure 5.8: Two-layer attestation in detail.

5.5 Password-Free Authentication

The secured smart sensor configuration approach combined with configuration attestation is capable of protecting confidential configuration data and synchronizing the state between smart sensor and configuration back-end. However, one drawback of the presented approaches is that for the applied symmetric encryption schemes, key material needs to be distributed in the system. The following two methods could be used to mitigate the key distribution problem:

Asymmetric cryptography. Asymmetric cryptography is widely used to mitigate the key distribution problem. However, compared to symmetric cryptography, longer keys are required and the performance of asymmetric cryptography in general is worse compared to the performance of symmetric cryptography [198].

KE protocols. Using KE protocols, involved entities can jointly agree on a common shared secret over an insecure communication channel. Thus, keys do not need to be distributed before having to establish a secured communication channel. However, traditional KE protocols such as DH do not provide authentication, and thus, any two entities could agree on a common shared secret.

As argued above, both possible countermeasures come with a drawback that needs to be mitigated. Therefore, we propose to use an EKE protocol for authenticating while agreeing on a common shared secret. In particular, we propose to use the so-called Simple PAssword-based KE (SPAKE) algorithm proposed by Abdalla et al. [199]. In this protocol, an EKE can be performed that relies on the well-known DH KE protocol. Figure 5.9 shows the protocol's principle that is applied to the Elliptic-Curve Diffie-Hellman (ECDH) protocol. The adapted protocol is presented in this section. The content of this section is mainly based on work published in the paper *Automated Authentication Credential Derivation for the Secured Configuration of IoT Devices* [200].

$$
\begin{array}{ccc}
\text{Alice} & \begin{array}{c}\text{agree on public}\\ \text{parameters } G, H(\cdot)\end{array} & \text{Bob} \\
\hline
a \leftarrow rand(\mathbb{F}_p) & & b \leftarrow rand(\mathbb{F}_p) \\
A \leftarrow aG & & B \leftarrow bG \\
M \leftarrow h_A G & & N \leftarrow h_B G \\
 & \xrightarrow{A^* \;\leftarrow\; A + pM} & \\
 & \xleftarrow{B^* \;\leftarrow\; B + pN} & \\
N \leftarrow h_B G & & M \leftarrow h_A G \\
K_A \leftarrow a(B^* + (pN)^{-1}) & & K_B \leftarrow b(A^* + (pM)^{-1}) \\
 & K \leftarrow K_A \overset{!}{=} K_B & \\
 & SK \;\leftarrow\; H(h_A, h_B, A^*, B^*, p, K) & \\
\hline
\end{array}
$$

Figure 5.9: Version 2 of the SPAKE protocol that is based on ECDH. Here, G is the ECC generator point, $H(\cdot)$ is a one-way function, h_A and h_B are Alice's and Bob's hashed identities, and p is a shared secret between Alice and Bob that should be based on the current configuraiton C_k in our approach.

We suggest to apply the basic ECDH-based SPAKE2 algorithm that is shown in Figure 5.9, but base the common shared secret on the current configuration: $p := H(C_k)$. That is, any Session Key (SK) that is generated by this approach is now based on the currently applied configuration C_k as well as the standard ECDH principle. This implies that any configuration update will also automatically change the common shared secret p. For a configuration update that is sent from Alice to Bob, the following steps are performed:

1. Alice and Bob want to perform a secured configuration update. Both currently share the common knowledge that the k^{th} configuration C_k is applied by Bob.

2. Alice now prepares the $(k+1)^{th}$ configuration C_{k+1} for Bob which should be encrypted. Thus, Alice and Bob perform an EKE based on C_k which yields SK_k. Alice uses this key to protect C_{k+1}, while Bob uses SK_k to decrypt C_{k+1}.
3. If Bob is able to successfully decrypt and verify the configuration update C_{k+1} and successfully verify all parameters as explained in Section 5.3, the configuration update is applied and Alice is notified of the update result by Bob.
4. If Bob successfully applies C_{k+1}, he and Alice now share the same common secret that this configuration is applied and can generate new SKs based on this configuration. If the configuration update is rejected by Bob, he and Alice will still be able to use C_k as a basis for future KEs.

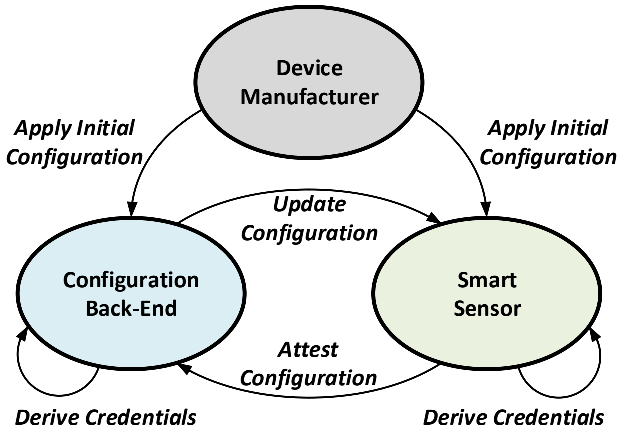
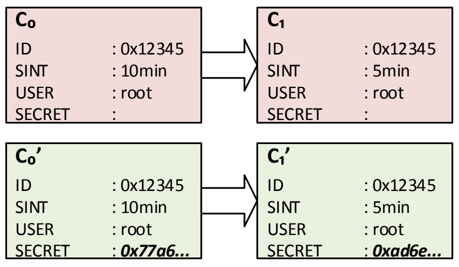


Figure 5.10: Proposed principle of common shared secrets that are based on configurations. Configuration updates are performed by different entities during the smart sensor's lifecycle.

Using this principle, configurations and thus common shared secrets between configuration back-end and smart sensor are synchronized during the smart sensor's entire lifecycle. The principle for the entire lifecycle is depicted in Figure 5.10. This principle entails two advantages that are not only capable of increasing the system's security but also improve the usability of the system. These advantages are as follows:



C_0

ID	: 0x12345
SINT	: 10min
USER	: root
SECRET	:

C_1

ID	: 0x12345
SINT	: 5min
USER	: root
SECRET	:

C_0'

ID	: 0x12345
SINT	: 10min
USER	: root
SECRET	: ***0x77a6...***

C_1'

ID	: 0x12345
SINT	: 5min
USER	: root
SECRET	: ***0xad6e...***

Figure 5.11: Example of a configuration update that automatically triggers the creation of a new common shared secret. If the user changes the sampling interval ($SINT$) of a smart sensor, only this value is changed between initial configuration C_0 and configuration C_1 in a traditional system. Using our proposed approach, for both initial configuration C_0' and subsequent configuration C_1' a shared secret is automatically derived.

- The security of each smart sensor will be improved by the presented approach since one of the biggest security weaknesses, using default authentication credentials, can be mitigated. Since any configuration update entails a new configuration version, the common shared secret will also be automatically updated when users update their device's configuration. That is, simple configuration updates such as changing the sampling frequency of s smart sensor entails a new common shared secret between configuration back-end and smart sensor, as shown in the simple example in Figure 5.11.
- Using this automated shared secret derivation process, neither do users need to change initial authentication credentials, nor do users need to remember sophisticated username and password combinations. Instead, the configuration back-end automatically synchronizes with the configured smart sensor and thus can be seen as a password manager for users. This however, entails the drawback that the configuration back-end now contains confidential information that users may not want to share. Therefore, we additionally suggest to support so-called local and mobile configuration back-ends that only fetch the initial configuration C_0 from the device manufacturer's configuration back-end and then manage the users configurations going forward. These local configuration back-ends could either be hosted on a server by the user or running on a mobile device. In both cases, the trustworthiness of these devices needs to be assumed. The principles of local configuration back-ends and the resulting system models are shown in Figure 5.12.

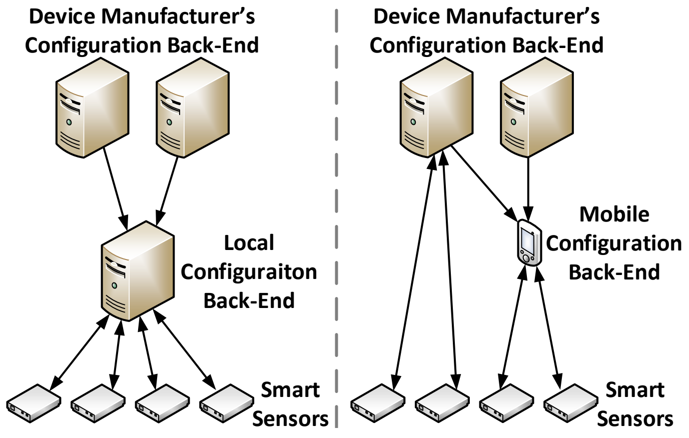

Device Manufacturer's Configuration Back-End
Local Configuraiton Back-End
Smart Sensors
Device Manufacturer's Configuration Back-End
Mobile Configuration Back-End
Smart Sensors

Figure 5.12: On the left hand-side, the user self-hosts a local configuration back-end on a server instance that is assumed to be sufficiently secured against attacks. On the right-hand side the user relies on the device manufacturer's configuration back-end for some smart sensors and uses a mobile configuration back-end for others. In both cases, only the initial configuration C_0 is fetched from the device manufacturer's configuration back-end.

5.6 Secured NFC Transport-Layer Protocol

The approaches presented in Section 5.3 - Section 5.5 propose a secured configuration approach that is capable of protecting the confidentiality, integrity, and authenticity of configuration data, while offering convenience to end users. Since the approach is tailored for configuration tasks, it offers no flexibility for other tasks. Therefore, in this section we propose a generalized protocol that may be used for any task where a protected NFC channel is required. The content of this section is mainly based on work published in the paper *QSNFC: Quick and Secured Near Field Communication for the Internet of Things* [201]. Since our proposed protocol also offers performance benefits compared to other protocols such as TLS, we denote it as Quick and Secured NFC (QSNFC). These performance benefits make our presented protocol especially suited for NFC use cases such as our presented NFC-based configuration approach. However, the protocol is not limited to this use case and therefore not all protocol properties are tailored for requirements of our presented NFC-based configuration approach (e.g., 0-RTT property).

Since most secured transport-layer protocols are designed to be used in the Internet, the terms *client* and *server* are common in this context. However, in the context of NFC these two terms are not commonly used. Therefore, we define them first:

QSNFC Client. In general, a client contacts the server and by doing so establishes a connection. Similarly to that, the QSNFC client also tries to establish a secured communication channel with the QSNFC server. Since the client is initiating the NFC communication, it can be seen as the *active component* in NFC terminology.

QSNFC Server. The QSNFC server is contacted by the QSNFC client in the context of establishing a secured communication channel as well as in an NFC context. Therefore, the QSNFC server could be seen as the *passive component* in NFC terminology.

5.6.1 Secured Transport-Layer

To provide the required flexibility, we propose a protocol that, contrary to most other secured NFC communication approaches, can be seen as a transport-layer protocol. Thus, the need to implement application-specific security mechanisms is eliminated. NFC applications can instead rely on a transparent security layer provided by our proposed protocol. Figure 5.13 shows the classification in the TCP/IP layer model of QSNFC compared to the protocol stacks of TLS and Datagram TLS (DTLS). Similarly to these protocols, QSNFC is placed directly underneath the actual applications and provides its capabilities for secured data transfer to the upper layer. To transport data, the NDEF protocol is used by QSNFC whereas TLS and DTLS rely on TCP and User Datagram Protocol (UDP) respectively. NDEF then uses so-called Application Protocol Data Units (APDUs) to transfer data over the NFC link layer.

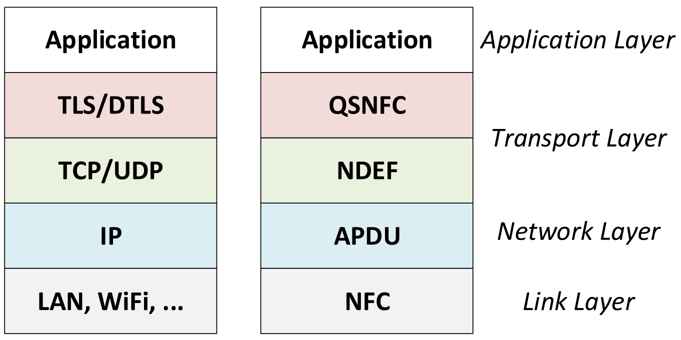


Figure 5.13: Protocol stacks based on the TCP/IP layer model for TLS/DTLS and QSNFC respectively.

5.6.2 0-RTT Property

To facilitate the secured communication channel that is provided by QSNFC, a KE process needs to be performed. Traditional protocols such as TLS (until version 1.2) require two round trips for this KE process. First, parameters such as the used algorithm or key lengths are agreed on in the so-called *client hello-server hello* round trip communication. After that, the actual KE is performed. Contrary to that, so called 0-RTT protocols do not require these two round trips; instead, protected data transfer can immediately be started, at least for recurring connections. Such recurring connections are defined as connections between communication partners that already communicated with each other in the past. Figure 5.14 illustrates the communication principle. The upcoming TLS version 1.3 [202] as well as our proposed protocol QSNFC support this 0-RTT principle. Similarly to Google's QUIC protocol [203] that also supports 0-RTT connection establishment for UDP connections, QSNFC uses cached information to speed up connection establishment for recurring connections.

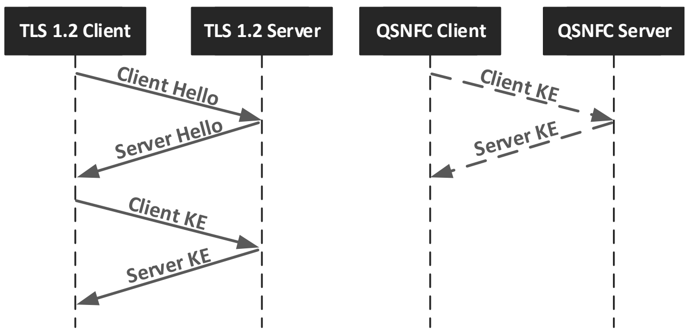


Figure 5.14: Round trips required for TLS 1.2 and QSNFC respectively.

5.6.3 QSNFC Protocol in Detail

In this section, the QSNFC protocol that we propose as a quick and secured NFC transport-layer protocol is discussed in detail.

5.6.3.1 Connection Establishment

Since the 0-RTT property can only be fulfilled for recurring connections, we propose to define a so-called *initial handshake* as well as *subsequent handshakes* for QSNFC.

Initial handshake. On the first connection attempt, a client has no information about the respective server and a so-called initial handshake thus needs to be performed between these two entities. To initiate the handshake, the client sends a so-called *inchoate* Client Hello (CH) message to the server that only contains the client's ID, a fresh nonce, and the crypto primitives that are supported by the client.

The server recognizes this inchoate information and replies with a Reject (RJ) message that contains the server's long-term DH public value, a fresh nonce, a certificate chain, a signature of the long-term DH public value, the server's ID and a timestamp protected by AE, and a signature of the nonce that was received by the client. If the client is already in possession of a (partial) certificate chain (e.g. from previous sessions), an optional certificate hash chain is also included in the inchoate CH message. The server then only returns required certificates instead of the complete certificate chain. The complete initial handshake is shown in detail in Figure 5.15.

Upon receiving the server's RJ message, the client can verify the integrity of both nonces, the selected crypto primitives, and authenticate the server's long-term DH public value using the respective certificate chain and signature. After successfully verifying this information, the client sends a *complete* CH message containing the client's ephemeral DH public value as well as an optional payload that is already protected by a key generated from the server's long-term DH public value and the client's ephemeral DH public value.

If a handshake fails, the server responds with an Abort (AB) message instead of a RJ message. Potential causes for aborting a handshake are, for instance, insufficient crypto primitives proposed by the client or from forged messages that are detected. AB messages contain the server's long-term DH public value and a signature thereof, a fresh nonce, the reason for aborting the handshake, as well as a signature of the client's and server's nonces and the abort reason. Upon receiving an AB message from the server, the client has to restart connection establishment via an initial handshake.

Subsequent handshake. In case the QSNFC client is already in possession of the QSNFC server's long-term DH public value (via a previous initial handshake), a shared secret can already be calculated. Thus, the client can send a so-called *complete* CH message without having sent an *inchoate* CH message first. The complete CH message contains the client's ephemeral public key as well as a fresh nonce protected by AE. Since the client is already in possession of the server's long-term DH public value, a shared secret using this value and its own ephemeral public key can be calculated. Then, data can be secured by this key and included into the complete CH message.

If the subsequent handshake is successfully verified by the server, a Server Hello (SH) message is sent by the QSNFC server as response. This SH message contains the server's ephemeral DH public value and a fresh nonce, as well as optional data that is protected by AE. The complete subsequent handshake process is shown in detail in Figure 5.16.

After QSNFC client and server are in possession of each others' ephemeral DH public value, a so-called *forward-secure key* can be calculated. That is, after the SH message is received by the client, both communicating partners can switch to encrypting packets using this forward-secure keys. In addition, forward secrecy is also provided by using nonces and a message counter for client and server messages. This information provides freshness of all following messages and is sent as so-called additional authenticated data in addition to data that is protected by AE.

Standard data. After the subsequent handshake is completed, client and server can proceed in their communication with sending so-called Standard Data (SD) messages. These messages can be sent by the client as well as by the server, and thus, enable bidirectional NFC-communication that provides forward secrecy. The details for SD messages are shown in Figure 5.16. These messages include the sender's identity as well as data protected by AE.

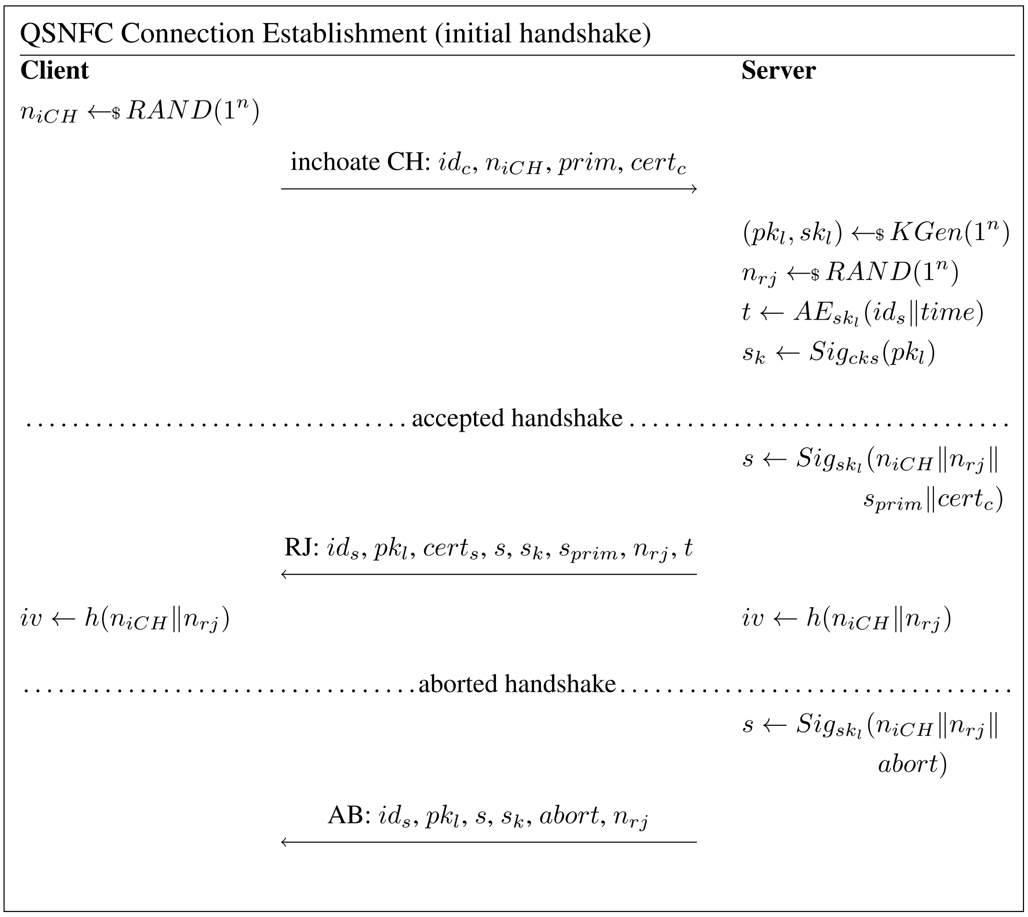

QSNFC Connection Establishment (initial handshake)
Client
Server
$n_{iCH} \leftarrow_{\$} RAND(1^n)$
inchoate CH: id_c, n_{iCH}, $prim$, $cert_c$
$(pk_l, sk_l) \leftarrow_{\$} KGen(1^n)$
$n_{rj} \leftarrow_{\$} RAND(1^n)$
$t \leftarrow AE_{sk_l}(id_s \| time)$
$s_k \leftarrow Sig_{cks}(pk_l)$
accepted handshake
$s \leftarrow Sig_{sk_l}(n_{iCH} \| n_{rj} \| s_{prim} \| cert_c)$
RJ: id_s, pk_l, $cert_s$, s, s_k, s_{prim}, n_{rj}, t
$iv \leftarrow h(n_{iCH} \| n_{rj})$
$iv \leftarrow h(n_{iCH} \| n_{rj})$
aborted handshake
$s \leftarrow Sig_{sk_l}(n_{iCH} \| n_{rj} \| abort)$
AB: id_s, pk_l, s, s_k, $abort$, n_{rj}

Figure 5.15: Initial handshake: The parameters are: the client's and server's ID ($id_{c|s}$), the client's supported crypto primitives ($prim$), the crypo suite (s_{prim}) chosen by the server, the long-term DH public and secret value (pk_l, sk_l), cached certificates (optional, from previous sessions) by the client ($cert_c$), the server's certificate chain ($cert_s$), n_{iCH} & n_{rj} (both fresh nonces of length n bytes). $RAND(1^n)$ denotes a function which returns n random bytes. The key cks is the private key corresponding to the server certificate's public key.

5.6.3.2 Connection Tear-Down and Cached Data Replacement

Although in the context of NFC no connection concept exists, we introduced connections by the previously proposed QSNFC connection establishment process. During this process, key material and other information regarding the other communication partner are exchanged and cached at the respective entity. Still, since there is no concept such as out-of-order packet reception or multiple streams that are known from TCP, no dedicated packet for ending a connection is required in QSNFC. However, since storage space may be constrained, only a limited amount of data regarding communication partners can be stored. Therefore, we propose to use cache replacement strategies for managing stored information.

5.6.4 Example Use-Cases and Limitations of QSNFC

As mentioned previously, QSNFC introduces client and server roles in NFC. Depending on the context in which the proposed protocol is applied, these roles might switch due to the QSNFC's requirements. This is due to the fact that the client must be able to validate the certificate chain and a higher ranked certificate might thus need to be fetched from a third entity.

Smartcard and Reader. In this scenario, the NFC reader acts as passive device and provides the required energy to power the smartcard. Therefore, the reader is assigned the client role since only this device can initiate the connection establishment. In addition, only the reader will have the capability to validate the smartcard's (server's) certificate chain if a networking connection is required to do so.

Smartphone and Smart Sensor. Since most modern smartphones include NFC-reader capabilities, the smartphone is the entity that initiates the QSNFC handshake and thus acts as the QSNFC client. All modern smartphones are equipped with the option of connecting to the Internet which also is required if the IoT device's (server's) certificate chain needs to be validated using a network connection.

Machine-to-Machine (M2M). In an M2M communication scenario such as Robot-to-Machine communication, the client and server roles generaly cannot be pre-determined. Contrary to the first two scenarios, these roles should be assigned as required by the connection establishment sequence. In addition, it has to be considered that the client may need to be able to verify the server's certificate chain using a network connection.

Limitations. The proposed QSNFC protocol includes the validation and thus authentication of the QSNFC server. However, the QSNFC client is not authenticated in the proposed protocol. This is by design due to common scenarios such as smartcard and reader where one device is resource-constrained and thus neither offers sufficient storage for certificate chains nor offers the capability of using a network connection for validating these certificate chains.

QSNFC Connection Establishment (subsequent handshake and data transfer)

Client | **Server**

$(pk_c, sk_c) \leftarrow\$ KGen(1^n)$

$sk_i \leftarrow (sk_c, pk_l)$

$k_i \leftarrow KDF(sk_i, iv)$

$n_{cCH} \leftarrow\$ RAND(1^n)$

$c \leftarrow AE_{k_i}(n_{cCH} \| data, pk_c)$

$$\xrightarrow{\text{complete CH: } id_c,\, pk_c,\, n_{cCH},\, c,\, t}$$

$n_{sh} \leftarrow\$ RAND(1^n)$

$sk_i \leftarrow (sk_l, pk_c)$

$k_i \leftarrow KDF(sk_i, iv)$

$(pk_s, sk_s) \leftarrow\$ KGen(1^n)$

$c \leftarrow AE_{k_i}(n_{sh} \| data, pk_s)$

$$\xleftarrow{\text{SH: } id_s,\, pk_s,\, c}$$

Client	Server
$sk \leftarrow (sk_c, pk_s)$	$sk \leftarrow (sk_s, pk_c)$
$k \leftarrow KDF(sk, iv)$	$k \leftarrow KDF(sk, iv)$
$iv \leftarrow h(n_{cCH} \| n_{sh})$	$iv \leftarrow h(n_{cCH} \| n_{sh})$

.... standard data ....

$aad \leftarrow h(iv + \#SD_c)$

$c \leftarrow AE_k(data, aad)$

$$\xrightarrow{\text{SD: } id_c,\, c}$$

$aad \leftarrow h(iv + \#SD_s)$

$c \leftarrow AE_k(data, aad)$

$$\xleftarrow{\text{SD: } id_s,\, c}$$

Figure 5.16: Subsequent handshake and standard data: The messages above the dotted line represent the subsequent handshake. The messages underneath the dotted line represent SD messages which can be sent bidirectionally. The function $AE_k(data, aad)$ denotes AE of data including additional authenticated data (aad) using key k. The function $h(\cdot)$ denotes a one-way function. The parameters for the subsequent handshake are: the client's and server's ephemeral DH public and secret value ($pk_{c|s}$, $sk_{c|s}$), the initial key k_i, the final key k and n_{cCH} & n_{sh} (both fresh nonces of length n bytes). The function $KDF(\cdot)$ generates a shared key that is used by client and server for AE. $\#SD_{c|s}$ are message counters for client and server messages respectively and are used to verify freshness.

6 Evaluation

In this chapter, we present evaluation results for the four contributions of this thesis that were listed in Section 1.4. Depending on the context, we evaluate the feasibility, performance, reliability, efficiency, or usability of our presented approaches and also discuss potential limitations.

6.1 NCS Attacks

In this section, we highlight the impact of our presented NCS attacks on a system and also discuss the feasibility of our presented bit-flip attacks and consider limitations of the proposed JEEC countermeasure. We presented parts of this evaluation already when discussing potential countermeasures in Section 4.1.

6.1.1 Impact

To evaluate the impact on a system controlled by a networked controller, the step response of the system without any maliciously modified packet or lost packet is analyzed. This step response is shown in Figure 6.1. As can be seen, the controlled system reaches the desired reference in $200\,\mathrm{ms}$, minimally overshooting the desired value. If the system is attacked such that maliciously modified sensor measurements are used by the control algorithm, or packets are dropped by the applied security mechanism, the system may become unstable. Figure 6.2 shows the step response for a system where 25% of all packets are lost due do being maliciously modified. This system already requires $300\,\mathrm{ms}$ to reach the desired reference value with more overshooting happening. The system shown in Figure 6.3 is evaluated with 50% packet loss. This leads to a system that overshoots the desired reference value a lot during the entire analyzed time window of $800\,\mathrm{ms}$.

6.1.2 Feasibility of Attacks

As shown in Figure 4.5, a limited number of BERs can lead to PERs that have a highly negative impact on the functionality of an NCS. We presented multiple scenarios, where a BER of 10^{-3} may lead to PERs of 0.5 or even higher. As we have shown, this may result in an unstable system and lead to undesired system behavior that can even cause physical damage or threaten human lives. Contrary to other attacks such as DoS attacks where the adversary usually needs more resources to attack a system, only a limited number of deliberately induced errors is sufficient to bring a system down in our presented bit-flip attack.

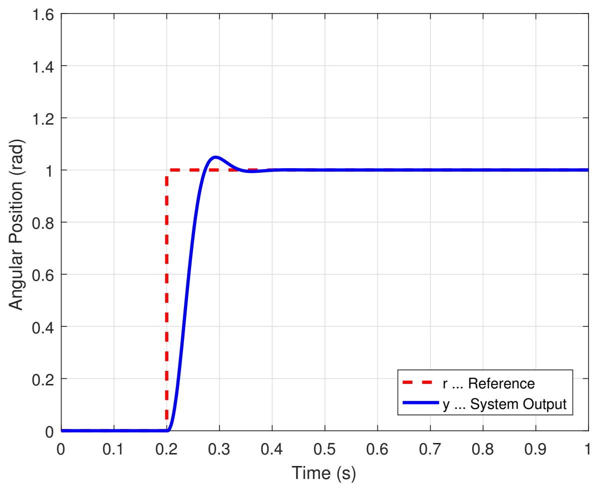


Figure 6.1: Step response of the evaluated system without any maliciously modified or lost packets.

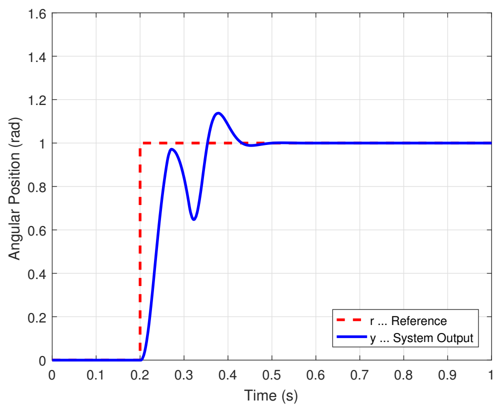


Figure 6.2: Step response 25% packet loss.

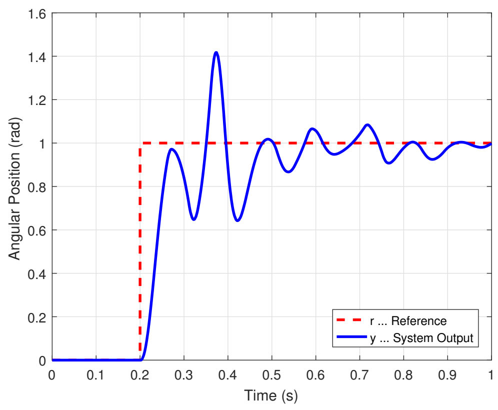


Figure 6.3: Step response 50% packet loss.

6.1.3 Limitations of Countermeasure

The presented countermeasure of using JEEC also entails the following two drawbacks:

1. If a communication technology such as Ethernet is used, at least EDC are already applied by the network stack to detect faulty packets and to request retransmission. Thus, error handling might be redundant to some degree if JEEC is used. However, we argue that JEEC still is a feasible countermeasure to mitigate the presented bit-flip attacks due to the following three reasons: (i) By using JEEC, no assumption regarding the used networking technology needs to be made when modeling an NCS. (ii) Unpredictable retransmissions of packets may have a negative impact on the NCS [204], (iii) JEEC that is performed in a single step offers higher efficiency. However, examining single-step JEEC schemes is considered out of scope of this thesis due to the topic's complexity.
2. Applying encryption and FEC introduces additional delay to the NCS. However, determinis-

tic delays can easily be modeled and accounted for in NCSs [205]. To provide deterministic delays, we propose to use dedicated hardware components for encryption/decryption as well as for FEC, as shown in Figure 6.4. By using such dedicated hardware components, a fixed delay can be provided. However, these components will also increase device costs.

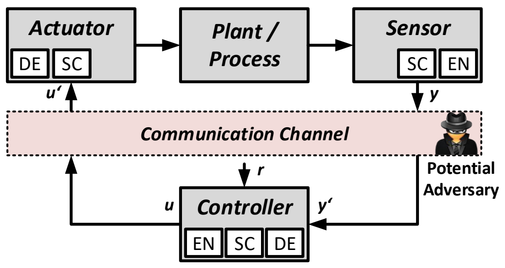


Figure 6.4: NCS system model extended by tamper-resistant SC, FEC encoding (EN) and decoding (DE).

6.1.4 Proof of Concept

A proof-of-concept implementation that demonstrates the feasibility of our presented approach was realized using the following hardware components:

1. 12 V DC motor with optical encoder.
2. First Raspberry Pi3 to read the encoder and send values over its Ethernet interface.
3. Second Raspberry Pi3 with two Ethernet interfaces (one USB-to-Ethernet) that acts as the controller. Cryptographic operations were implemented using OpenSSL.
4. Third Raspberry Pi3 with a Raspberry Pi Motorshield that acts as the actuator that includes the motor driver.

Using this setup, the impact of our proposed bit-flip attacks was demonstrated by randomly modifying single bits of the received sensor measurments. Depending on the setup (plaintext, AE, JEEC) different behaviours were demonstrated. However, no performance evaluation was performed in this setup since all cryptographic operations were performed in software. This is due to the reason that no available Raspberry Pi3 crypto shield did support symmetric cryptography.

6.2 Sensor-Based Covert Channels

In this section, we present evaluation results for the implemented sensor-based covert channels. We evaluated the presented covert channels regarding their achievable data rates and reliability on the following three different platforms. For each platform we list the sensors and (if required) registers that we exploited. We also discuss limitations of our covert channels.

1. CC2650 SensorTag running TI-RTOS 2.20 as OS
   a) Texas Instruments OPT3001 ambient light sensor, `Configuration`, `Low Limit`, `High Limit` registers

2. Raspberry Pi 3 including the SenseHAT extension, running Raspbian Strech as OS

   a) STMicroelectronics HTS221 humidity and temperature sensor, `HTS221_TMP_OUT_L` and `HTS221_HUM_OUT_L` registers
   b) STMicroelectronics LPS25H pressure sensor, `LPS25H_REF_P_XL`, `LPS25H_PRESS_OUT_XL`, `LPS25H_TEMP_OUT_L`, `LPS25H_THS_P_L`, `LPS25H_RPDS_L` registers

3. OnePlus5 as well as Android Emulator, both running Android 8.0 as OS

   a) Ambient light sensor (generic interface)

6.2.1 Performance

The data rates we achieved on all three evaluation platforms are shown in Figure 6.5. Of the three used evaluation platforms, all presented types of sensor-based covert channels could only be implemented on the Raspberry Pi 3 with SenseHAT extension board. The CC2650 SensorTag only allowed to implement covert channels based on exploiting writeable registers, while on Android's managed sensor framework only covert channels based on triggering sensor readings could be implemented. One interesting aspect we noticed is that the covert channels implemented on the CC2650 SensorTag achieved higher data rates than the same implementations on the Raspberry PI 3. This is due to the deterministic scheduling principle of TI-RTOS where processes are scheduled alternately, compared to Raspbian's non-deterministic approach. The data rates achieved by covert channels based on triggering sensor readings achieved considerable lower data rates compared to covert channels based on exploiting writeable registers. That is due to the fact that the sensing process itself takes a certain amount of time and thus limits the number of interactions that can be used for stealthy data transfer in a covert channel. Compared to other covert channels that are based on exploiting hardware such as caches or memory, the achieved data rate is relatively low. The data rate of our presented covert channels is limited by the performance of a sensor register's read and write operations. In case of a covert channel based on triggering a sensor, the data rate is limited by the sampling frequency of the respective sensor. Additionally, the introduced overhead due to our mechanisms that ensure a reliable data transfer lower the achievable data rate.

Figure 6.6 demonstrates different static payload sizes under different noise profiles. That is, we simulated user processes that interact with the sensor in different intervals and thus interfere with our covert channel's data transfer. For instance, the process labelled `User 5s` in Figure 4.14(a) reads the sensor every five seconds and potentially interferes with an ongoing data transfer of our covert channel. One or even multiple of such user processes that access the same sensor as our covert channel may be running in a real-world setting. For this evaluation, 4 kB of data where transferred over a covert channel that is based on triggering sensors. As can be seen, larger packet sizes decrease the overall transfer time in most scenarios due to causing less communication overhead. However, if other user processes interact with the sensor more frequently, packet loss is too high and transfer time grows significantly. Our proposed dynamic packet size approach, however, is capable of mitigating such issues.

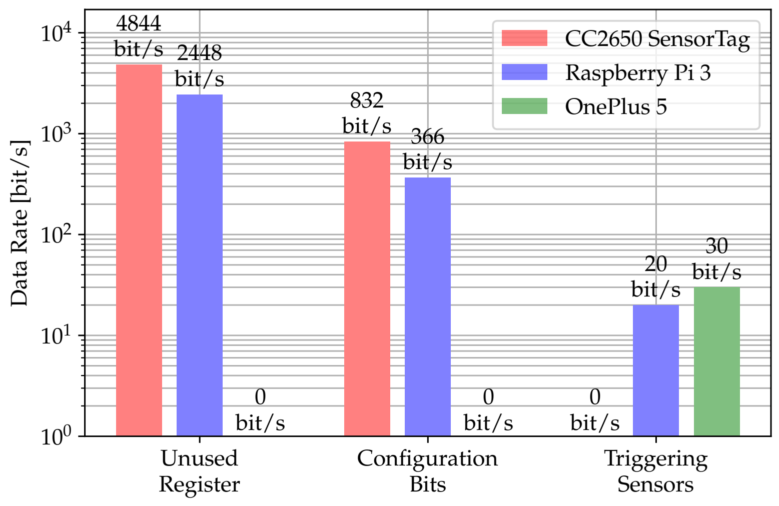


Figure 6.5: Data rates of different sensor-based covert channels on the three evaluation platforms we used. Attack types that are not supported by the respective platform are indicated with a data rate of 0 bits/s

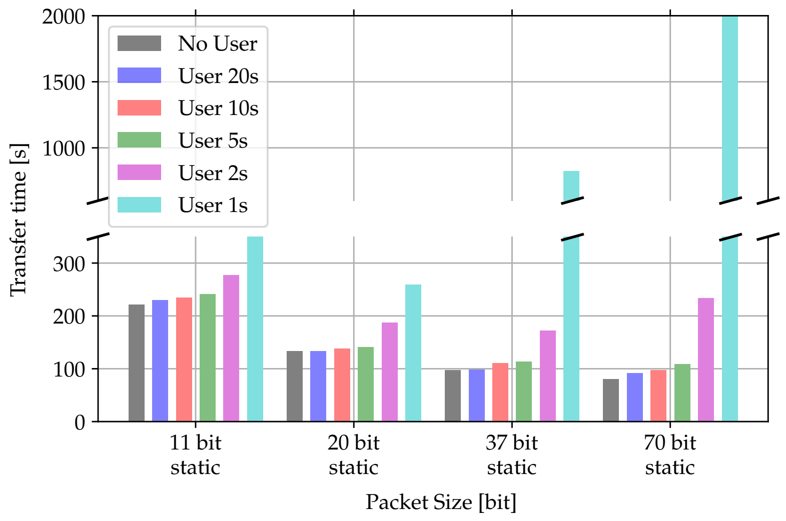
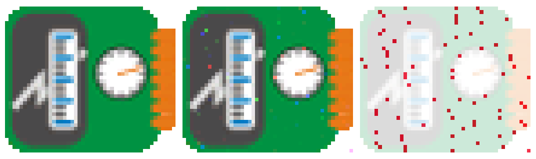


Figure 6.6: Data rates on Raspberry PI 3 with different static payload sizes. The specified user intervals indicate the frequency (interval between measurements) in which a user process accesses the sensor and thus interferes with our covert channel.

6.2.2 Reliability

To evaluate the covert channel's reliability, we test transferring an image of a sensor over a noisy channel. Figure 6.7 shows the source image as well as the received image and highlighted errors in the image if error detection is disabled and no retransmissions are thus made. Enabling EDC and retransmissions in a channel without noise adds an additional overhead of 6 % in transmission time. If we introduce a user process that accesses the sensor every 10 s, transfer time is increased and roughly 10 % of the packets needed to be retransmitted (123 out of 1062 required packets).

However, the image is then transferred successfully without any errors.



(a) Source. (b) Received. (c) Errors.

Figure 6.7: Data transfer with EDC and retransmissions disabled.

Finally, we evaluated dynamic switching of packet sizes which should improve covert channel performance in noisy settings. Figure 6.8 shows a scenario where a user process starts to access the sensor every second while data transfers are already running. In this figure, each black dot represents the user process accessing the sensor while each blue, red, and green dot represents a successfully transferred data packet by the respective covert channel. In the evaluated scenario, we compare two static packet sizes (11 bit and 37 bit) with our proposed dynamic packet size approach. The dynamic approach is configured such that it can switch between packet sizes of 11, 20, and 37 bit. We can identify three phases in this evaluation:

- 0 s – 30 s: There is no interference. Both covert channels with static size transfer packets with their respective payload size. The dynamic size covert channel also transfers packets with a payload size of 37 bit since there is no failed transfer.
- 30 s – 60 s: A user process starts to interfere, accessing the sensor every second. Both, the static covert channel and the dynamic covert channel with 37 bit payload size, cannot transfer packets anymore, while the covert channel with 11 bit payload size can transfer packets with a lower frequency due to losing some packets.
- 60 s – 80 s: The dynamic packet size approach has successfully decreased packet size and synchronized this information between sender and receiver. Data transfer starts again using 20 bit payload sized packets. As packets are still lost, the payload size is further decreased as can be seen on the smaller intervals after 70 s.

Although the payload size chosen by the dynamic approach might not be the optimum choice for a fixed setting, the approach allows the covert channel to function reliably for changing settings. Therefore, we think our proposed approach offers the best reliability for an unknown setting where infrequent interfering sensor accesses need to be expected.

6.2.3 Limitations

Depending on the type of covert channel implementation that is chosen, read as well as write access to the sensor might be required. In addition, both involved processes need to be able to alternately access the sensor. Depending on the technology used to access the sensor, advanced methods such as multi-master approaches might be required, for instance, when using I2C.

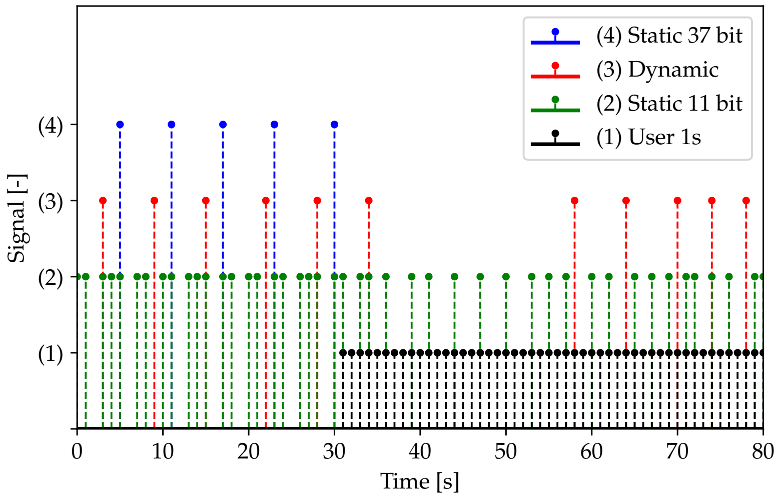


Figure 6.8: Evaluation of dynamic packet size.

6.3 Secured Sensor Configuration

In this section, our proposed secured sensor configuration approach is evaluated. We evaluate the usability based on a real-world example, conduct an extensive threat analysis to highlight the achieved level of security, discuss performance of secured NFC-based data transfer, and finally list limitations of our proposed approach. For all performed evaluations, the evaluation stick shown in Figure 6.9 that was developed in cooperation with our industrial partner Infineon was used. This stick includes an SC and offers an NFC as well as a Universal Serial Bus (USB) interface.

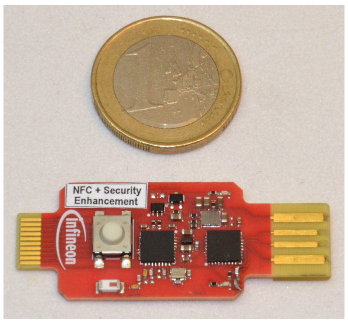


Figure 6.9: NFC + Security enhancement evaluation stick.

6.3.1 Usability

To evaluate the usability achieved by our proposed secured configuration approach, we compare the steps necessary to change initial configuration parameters for (i) our proposed approach and (ii) the well-known *temp stick* smart sensor[3] in the following two lists. The steps required by this

[3] https://tempstick.com/manuals/setup-guide-temp-stick-th.pdf

sensor can be seen as state of the art since also mostly all other pairing mechanisms (e.g., Google's Chromecast) require the installation of an application and the creation of a user account.

Our Approach

1. Install configuration app
2. Go to configuration website
3. Log in using sensor ID and password
4. Update current configuration
5. Read QR code with smartphone
6. Touch smartphone and sensor to transfer configuration

Temp Stick^{TM}

1. Install configuration app
2. Go to configuration website or launch app
3. Create account or log in using existing one
4. Pair sensor with account using sensor ID
5. Connect smartphone to WiFi network `Sensor Setup <sensor ID>`
6. Open website `http://10.10.1.1`
7. Configure WiFi network and WiFi password
8. Connect smartphone to previous WiFi network
9. Go to configuration website or open configuration app
10. Update current configuration and apply to sensor via Internet and WiFi

We argue that from these two lists the higher usability of our approach for initial configuration updates can be concluded. Especially for non-technophile persons the changing of WiFi networks and opening websites using an IP address might be a complex task that requires assistance. However, it has to be noted that for recurring configuration updates the required steps are similar for both compared approaches.

6.3.2 Security

To evaluate the level of security that can be achieved by our secured sensor configuration approach, we conduct a threat analysis [206]. In this analysis, the involved Entities (**E**) and Assets (**A**) that need to be protected are identified first. We then list potential Threats (**T**) with corresponding Countermeasures (**C**) that are provided by our proposed approach. If a threat cannot be entirely mitigated by our approach, the Residual Risks (**R**) are also discussed. The impact of all threats is categorized according to the STRIDE [207] threat model (Soofing, Tampering, Repudiation, Information disclosure, Denial-of-Service, Elevation of privileges).

6.3.2.1 Entities

The following entities are identified in our proposed secured sensor configuration approach. To better specify the scope of this thesis, we list assumptions that were made about certain entities.

E1 The sensor that is being configured.

E2 The owner of the sensor that is being configured.

E3 The device manufacturer of the sensor that is being configured.

E4 The device manufacturer's configuration back-end. Since we do not cover security aspects for this entity in this thesis, we assume that the entity is sufficiently secured against attacks such that no loss of confidential data is caused by this entity.

E5 The device owner's local or mobile configuration back-end. We also assume that this entity is sufficiently protected against attacks such that a loss of confidential data needs to be considered here.

E6 A person that applies configuration updates to the sensor that is being configured. This entity might be different to the device owner, especially in industrial settings where potentially untrusted personnel is used to apply configuration updates.

E7 An adversary that intends to perform malicious activities. We do not make any assumption about the score of attacks the adversary may be able to perform. However, for this analysis we limit attacks to the sensor's configuration interface. For this interface, we consider both remote and local physical attacks.

6.3.2.2 Assets

The following assets that need to be protected are identified in our threat analysis.

A1 The correct functionality of the sensor that is being configured must be protected. Our proposed configuration approach should prevent malicious activities that compromise this correct functionality in any way.

A2 Configuration data that is transferred may include confidential information. Thus, it is essential to protect its confidentiality, integrity, and authenticity.

6.3.2.3 Threats

After identifying the entities and assets for this threat analysis, the actual threats are finally identified. We do not claim that the presented list of threats is exhaustive but rather is a list of threats we at the time of writing this thesis believe are most relevant for the security of the proposed secured sensor configuration approach.

T01 The sensor's configuration interface might be subjected to remote attacks.

STRIDE: S, R, I, D, E

C01 Threat is mitigated by using an NFC-based configuration interface that requires the adversary to be in close proximity to target the configuration interface.

T02 An adversary that is in close proximity to the sensor's configuration interface might try to eavesdrop configuration data that is transferred to the sensor via the proposed NFC interface.

STRIDE: I

C02 Threat is mitigated by protecting the confidentiality of transferred configuration data with AE and the proposed protocol.

T03 An adversary that is in close proximity to the sensor's configuration interface might try to manipulate configuration data that is transferred to the sensor via the NFC interface.

STRIDE: S, R, E

C03 Threat is mitigated by protecting the integrity and authenticity of transferred configuration data with AE and the proposed protocol.

T04 An adversary that is in close proximity to the sensor's configuration interface might try to capture valid configuration packets and try to apply these packets either at other sensors or at a later time at the intended sensor to negatively impact its functionality.

STRIDE: S, R, D

C04 These so-called replay attacks are mitigated by our proposed approach where sensor ID and configuration version number are included in the configuration packets. Thus, the sensor is capable of rejecting malicious configuration packets.

T05 An adversary might be capable of eavesdropping NFC communication during the key agreement process. Subsequently, the adversary could be able to eavesdrop and manipulate transferred configuration data.

STRIDE: S, R, I, D, E

C05 Our proposed authenticated KE approach is based on the DH KE algorithm. This algorithm is robust against MITM attacks during KE.

T06 An adversary might learn the sensor's initial configuration C_0 or any subsequent configuration C_k which are used for authenticated KE. In addition, the adversary might also be able to capture any ongoing data that is transferred. The adversary thus might be able to infer the current session key and decrypt transferred confidential information.

STRIDE: I

C06 Due to adding random information that is generated by the true random number generator of the proposed SC, the adversary is not able to learn all information that would be required to derive session keys that both involved entities agreed on.

T07 The end-user might not change the initial configuration C_0 and no authentication credentials can thus be derived automatically. The adversary easily can authenticate itself against this device and perform malicious activities.

STRIDE: S, T, R, I, D, E

C07 Although our approach cannot force end-users to change their initial configuration, a smart sensor running its default configuration is not that useful since it is not connected to any WiFi network or similar. Additionally, our proposed attestation approach is capable of finding devices that are running default configurations, which is especially useful in industrial settings.

R07 As mentioned, our proposed approach is not capable of forcing end-users to change their initial configurations, so a residual risk remains.

T08 If it is assumed that an adversary is able to apply a malicious configuration, for example, due to a sensor running on initial configuration, the device might perform malicious activities that may impact other devices or a system.

STRIDE: S, T, I, D, E

C08 Our proposed two-layered attestation approach will prevent the sensor from accessing the network. Thus, no other device or system will be negatively impacted by a sensor running unintended or malicious configuration parameters.

T09 An entity might get the task to apply configuration updates to sensors, for instance, in an industrial setting. However, if this entity does not apply the configuration update intentionally or unintentionally this might result in the sensor not functioning as expected.

STRIDE: T, D

C09 Using our proposed two-layered attestation process for secured sensor configuration, the configuration status of each sensor can be monitored. Thus, it is easy to detect sensors where a desired configuration update was not applied.

T10 A device manufacturer may include intentional back doors into the configuration interface for internal use. An adversary might learn of this back door and exploit it for attacks.

STRIDE: S, T, R, I, D, E

C10 Using the proposed tamper-resistant hardware includes a CC certification process that is capable of mitigating this threat.

T11 The cryptographic algorithms chosen by the device manufacturer might be susceptible to attacks due to weaknesses in the algorithm or too short keys.

STRIDE: S, T, R, I, D, E

C11 In our proposed approach, all cryptographic relevant operations are performed in tamper-resistant hardware. Using such hardware includes a CC certification process that is capable of mitigating this threat.

T12 An adversary might perform DoS attacks targeting the NFC-based configuration interface.

STRIDE: D

C12 Due to the configuration interface being only available via NFC, the adversary has to be in close proximity to the device under attack. Also, the interface and all relevant operations are handled by the dedicated SC. Thus, normal sensor operation is not impacted since the DoS attack only impacts this SC.

R12 However, DoS attacks cannot be completely mitigated by our approach. Thus, if an adversary is capable of attacking the interface, no other configuration updates may be applied during such attacks.

T13 An adversary that has physical access to the sensor may perform physical attacks to reveal confidential information such as keys or confidential configuration data.

STRIDE: T, I, E

C13 All confidential information is stored in the SC's protected storage that is tamper-resistant and can thus mitigate such physical attacks.

T14 An adversary that has physical access to the sensor may perform physical attacks to manipulate sensor functionality. For instance, the configuration interface may be disabled by physically damaging the NFC antenna.

STRIDE: D

R14 Our proposed approach is not capable of mitigating such attacks.

Summarizing the threat analysis in Figure 6.10, there are 14 identified threats of which 11 are completely mitigated, 2 are partially mitigated, and 1 cannot be mitigated.

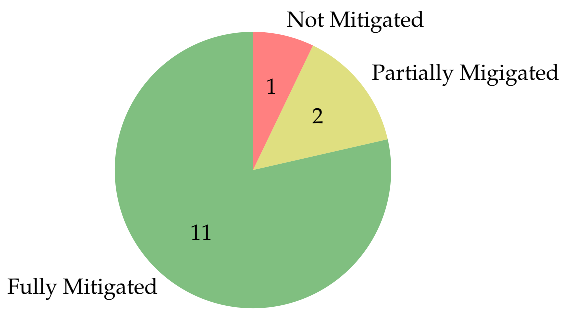


Figure 6.10: Summary of threat analysis.

6.3.3 Performance

To measure the performance of NFC-based data transfer, we evaluate various payload sizes using a Raspberry PI 3 with NFC shield as well as a Lenovo P2 Android smartphone. Data is transferred to the evaluation stick shown in Figure 6.9. The measured time includes NFC data transfer to the stick, decryption and storage in the secured storage, and confirming the configuration storage. Figure 6.11 shows the obtained results for payload sizes ranging from 128 bytes to 4096 bytes. As can be concluded from this figure, the configuration time for smaller payload sizes does not grow linearly due to the protocol overhead. For lager payload sizes, however, almost linear growth in configuration times can be observed. Compared to the Android smartphone, the Raspberry PI 3 with attached NFC shield achieved lower configuration times due to the shield being a more powerful reader than the NFC readers included in current smartphones. We argue that configuration times of roughly 1 s are still acceptable for users regarding the system's usability. Thus, we would suggest to limit configuration payload sizes to a maximum of 2048 bytes. Since larger configuration sizes that are transmitted using our proposed configuration approach will result in transfer times of 2 s and more, user acceptance of such an approach will decline due to the negative usability caused by such time consuming data transfers.

6.3.4 Limitations

As discussed when evaluating the performance of our proposed secured sensor configuration approach, payload sizes larger than 2024 bytes require configuration times larger than 1 s. The

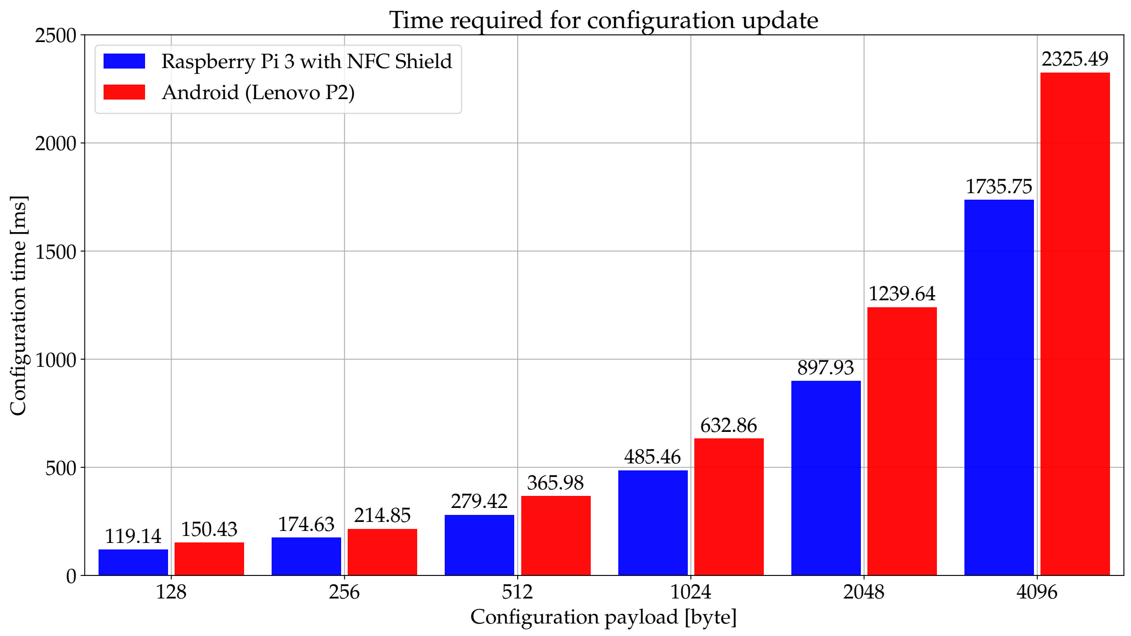


Figure 6.11: Time required to perform secured configuration updates with various payload sizes using NFC Shield on a Raspberry PI 3 and a Lenovo P2 Android smartphone as active devices.

reasons for this rather long runtime are the limited data rate provided by NFC and the processing time required by the tamper-resistant but resource constrained SC. Also, local attestation using the same SC might not be possible in all proposed events. To mitigate this limitations, attestation could be performed less frequently or more powerful dedicated hardware components could be integrated.

6.4 Secured NFC Protocol

In this section, the security provided by QSNFC as well as the performance of QSNFC's 0-RTT handshakes is evaluated. As an evaluation metric, we compare the protocol overhead of different scenarios. In addition, the protocol's limitations are discussed in this section.

6.4.1 Security

Each packet that is transmitted using our proposed QSNFC protocol contains a protected section for securing confidential payload. QSNFC uses AE with either initial keys or ephemeral forward-secure keys to protect the secured payload. Depending on the type of key used, two levels of secrecy can be provided: (i) Initial data that is secured using initial keys is protected at a level similar to TLS session resumption with session tickets. (ii) If the forward-secure keys are used, even greater secrecy can be provided since these keys are ephemeral. However, depending on the application and use-case, only one message round trip is probably needed. In this case, the initial keys will only be used to protect the data, without ever using the forward-secure keys. Any information that is transmitted unsecured in QSNFC (e.g., server and client identifiers or public keys) is considered non-critical such that an adversary who learns this information would gain no advantage. To highlight the provided security, we analyze QSNFC's countermeasures for each of the threats to NFC that were identified by Haselsteiner and Breitfuß [76]. We also list additional attack types that are mitigated by QSNFC.

Eavesdropping. Confidential information in QSNFC is transferred protected by AE in any message. Therefore, an adversary would only be able to learn public information such as identifiers or public keys.

Data corruption, data modification, data insertion. An adversary would not be able to corrupt, modify, or insert data in the secured payload section of QSNFC without such failures being detected by AE and thus by the QSNFC protocol. However, DoS attacks are possible by corrupting data and therefore, could cause messages to be invalid.

DoS attacks. DoS attacks cannot be mitigated by QSNFC (and any other wireless or contactless communication protocol) since data corruption can only be detected but not prevented.

MITM attacks. By relying on certificates for authentication and on a DH based KE, MITM attacks are effectively mitigated by QSNFC.

Replay attacks. By using nonces and message counters to ensure message freshness, QSNFC is capable of mitigating replay attacks. That is, if an adversary can record messages, replay attacks using these recorded messages are detected and mitigated by QSNFC.

Resumed handshake. Messages exchanged during initial handshake not only contain the client's and server's identity, but also a nonce. These nonces are used to prevent adversaries for resuming an initial handshake that was initiated by two entities (certificates are only validated in the initial handshake).

Physical attacks. Cryptographic operations that are required for our proposed KE process can either be performed in software or in a dedicated SCs. If such SCs are used, a higher level of security can be provided for QSNFC due to the mitigation of physical attacks.

6.4.2 Performance

To evaluate QSNFC's performance, the following four scenarios are evaluated:

1. Full handshake needs to be performed and the certificate needs to be transferred during this full handshake. The certificate is compressed to limit the overhead.
2. Full handshake needs to be performed, but the server's certificate is already cached by the client. This scenario might happen after aborted handshakes.
3. The server is already known to the client and required information is cached. Only the subsequent handshake needs to be performed.
4. After handshakes are performed, data is transferred using SD packets.

The results of this evaluation are shown in Figure 6.12. Of course, the overhead for a full handshake that requires transfer of a certificate or even certificate chain depends on the size of the certificate chain that is transferred. In the evaluated scenario, only a leaf certificate was transferred. That is, the shown overhead must be seen as an optimum in this given scenario. As can be seen from the evaluation, the 0-RTT property for recurring handshakes significantly reduced overhead compared to both full handshake variants. The overhead is actually only 42 bytes larger than the standard data transfer after the handshake. Comparing the minimal overhead for a full handshake with certificate transfer and subsequent handshakes reveals that overhead can be reduced by at least 90 % by using our proposed subsequent handshake approach in QSNFC.

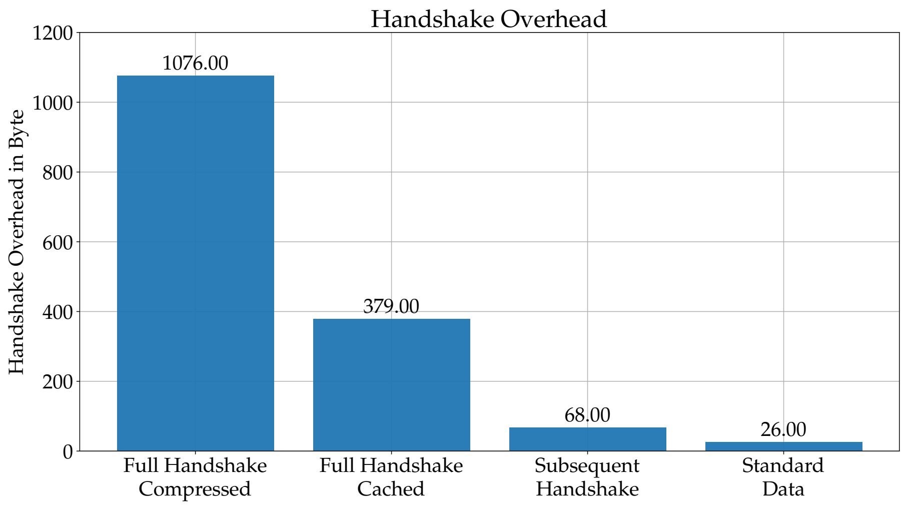


Figure 6.12: QSNFC overhead for transferring compressed certificates or cached certificate information in full handshake (initial followed by subsequent), subsequent handshake, and standard data.

6.4.3 Limitations

Since certificates need to be validated during the handshake process, a limitation of the QSNFC protocol is that either a working network connection must exist or all required higher-layer certificates must be stored on the device. In an offline approach where certificates are stored on the device, certificate revocation cannot be performed easily. The QSNFC protocol also does not support mutual authentication since only the server is required to provide a certificate that is then validated by the client. This decision was made consciously since at least one NFC device is often very resource-constrained (e.g., smartcards) and certificate storage and validation would thus be infeasible for such devices.

7 Conclusion and Future Work

In this chapter we conclude this thesis by summarizing our findings and contributions. We also discuss possible directions for future work that could originate from topics covered in this thesis.

7.1 Conclusion

The results obtained throughout this thesis are twofold. Firstly, we investigated sensor-related security issues and demonstrated novel attacks. Secondly, we identified unprotected sensor interfaces, in particular the configuration interface, as a main security weakness and therefore proposed a secured sensor configuration approach.

The first type of security issue we identified is the protection of sensor data. Usually, measurements in NCSs are transferred either unprotected or protected by encryption. We demonstrated that this approach may be self-defeating in a sense that it makes other attacks easier. Although the confidentiality, integrity, and authenticity of sensor data may be protected, adversaries can attack the system due to this protection. We highlighted this issue, defined so-called bit-flip attacks in NCSs, and proposed a simple yet effective countermeasure to mitigate the drawbacks of applying cryptography in NCSs. The second type of sensor-related security issue we identified are unprotected sensor interfaces. This includes the sensing interface itself but also unprotected configuration interfaces. We exploited both of these interfaces to build sensor-based covert channels, through which we transferred data between two otherwise isolated processes. In total, we presented three different sensor-based covert channels that provide a trade-off between the achievable data rate of the respective covert channel and the probability of such a covert channel being detected. While two of our presented covert channels require read and write access through the sensor's configuration interface, the third type of covert channel exploits information generation by triggering the sensor's measurement process. We demonstrated our covert channels on well-known platforms such as Linux and Android where even state-of-the-art code analysis tools designed to detect covert channels are unable to identify our proposed exploits. Again, we presented simple yet effective countermeasures to mitigate the covert channels we demonstrated. In addition, we also provided a framework to facilitate testing if a given sensor exhibits the discussed weaknesses. We believe that the presented security issues are only the tip of an iceberg, and many other sensor-related security issues as well as more sophisticated countermeasures can be found. However, we consider raising the awareness for sensor-related security issues by demonstrating these security issues a main contribution.

In this thesis, we also presented an NFC-based configuration approach for sensors that, in our opinion, is capable of mitigating security issues arising from unprotected or insufficiently protected configuration interfaces of sensors. We did, however, no only propose this NFC-based configuration interface but a complete architecture for sensor configuration. The configuration interface that we proposed to be included in sensors is based on the usage of resource-efficient dedicated security hardware that is tailored for resource-constrained devices such as smartcards. Therefore, our presented security methods are also intended to be efficient. Furthermore, the configuration interface we proposed in this thesis should be usable through the sensor's entire lifecycle. That is, initial configurations as well as any subsequent configuration updates can be applied using the same interface. To secure the configuration process, we applied state-of-the-art cryptography with relevant information to mitigate issues such as replay attacks. We also proposed a novel two-layered attestation process that not only reduces attestation overhead but also improves security by denying network access to malicious nodes. To also improve the usability of our secured configuration approach, we further demonstrated that authenticated KE does not require the user to enter credentials but instead is capable of deriving these credentials from current configuration information. In combination, all the mentioned contributions result in a configuration framework that provides, as we think, a good trade-off between security, efficiency, and usability. Our evaluations demonstrate that 78 % of identified security threats that target a sensor's configuration interface can be completely mitigated, while only 7 % of these threats cannot be mitigated at all. We further demonstrated that this level of security is achieved while providing very good usability as well as efficiency and performance that is acceptable for most configuration scenarios. As the final contribution of this thesis, we presented a generalized transport-layer protocol for NFC that provides security to NFC applications, similarly to what HTTP Secure (HTTPS) provides to TCP-based networking applications. Therefore, the protocol we proposed in this thesis mitigates the need to implement application-specific security mechanisms as currently is done for most NFC applications. We demonstrated that the presented protocol is capable of reducing the connection-establishment overhead by more than 90 % by fulfilling the 0-RTT requirement for recurring connections. Thus, we think, the proposed protocol offers the security level and efficiency that is required by NFC applications.

7.2 Future Work

In this section, we discuss potential future work for sensor-based security issues as well as for secured sensor configuration.

7.2.1 Sensor-Based Security Issues

In this thesis, we proposed using JEEC to mitigate bit-flip attacks on sensor data in NCSs. Although we denoted the approach we suggested as being JEEC, we applied encryption followed by FEC in a sequential way. In literature, however, attempts at combining encryption and error correction in a single step exist. Although some approaches use the well-known AES encryption scheme, most approaches use less-known approaches (e.g., [208]) for which security needs to be proven by the respective community. Also, AES-based approaches, where internal matrices are extended to provide a one-step encryption and error correction, need to be extensively evaluated regarding their security properties. One interesting aspect could also be evaluating the impact of using stream ciphers instead of block ciphers in an NCS where bit-flip attacks occur.

Regarding sensor-based covert channels, ample of future work could be done in two directions. Firstly, we believe our presented covert channels are only the beginning and that many other exploitable issues in sensors could be found for building other sensor-based covert channels. Secondly, additional and more enhanced security measures need to be devised to mitigate existing and newly found security issues caused by sensors and their respective interfaces. Our presented countermeasures introduce limitations to the system such as only supporting sampling frequencies that are multiples of each other. These limitations may be limited by future research to find more advanced countermeasures, such as privilege-based sensor access or granting sensor access to single processes based on time slots.

7.2.2 Secured Sensor Configuration

In the area of secured sensor configuration, future research could be conducted to further improve any of the three requirements discussed in Section 1.3.2. The trade-off between security, efficiency, and usability is of course another relevant aspect also for future research. In addition to general research to improve any of these three requirements, we especially see potential for future work in the following topics that are related to the work presented in this thesis.

Attestation. We only presented initial ideas regarding a two-layered attestation approach using the exiting local trusted hardware module that we proposed to include in any smart sensor to facilitate a secured sensor configuration process. Since attestation is a very actively researched topic, we see potential for future research in this direction where our proposed architecture of LRoT and GRoT is used for novel and enhanced methods of two-layered attestation. We think that such approaches would not only be suitable for sensor networks where local attestation prevents malicious nodes from manipulating networks but could be used in any setting where networked devices communicate with each other.

Web-NFC. To further improve usability, we already conducted preliminary experiments with implementing the so-called Web NFC[4] draft standard on Android smartphones. This standard utilizes NFC capabilities that are provided by respective devices inside a JavaScript environment. In our preliminary experiments, we combined the Web NFC functionality with our proposed QSNFC protocol and an HTTPS secured communication to the server to establish an end-to-end encrypted communication channel between a potential configuration back-end and an NFC device with a smartphone as a gateway in between. This could not only improve the security of smart sensor configuration but also the usability of such a process. By using the proposed Web-NFC standard, no application would need to be installed on the smartphone and the process discussed in Section 6.3.1 could thus be improved tremendously. Therefore, we think future work should be done regarding Web NFC's capabilities for being applied in such a use case.

[4] https://w3c.github.io/web-nfc/

8 Publications

This thesis is based on the following peer-reviewed publications that appeared in journals, books, and conference proceedings. Figure 8.1 highlights the connection between the contributions of this thesis that are presented in Section 1.4 and these publications.

A. T. Ulz, T. Pieber, C. Steger, S. Haas, H. Bock, and R. Matischek, "Bring Your Own Key for the Industrial Internet of Things," in *Industrial Technology (ICIT), 2017 IEEE International Conference on*, pp. 1430–1435, IEEE, 2017

B. T. Ulz, T. Pieber, C. Steger, C. Lesjak, H. Bock, and R. Matischek, "SECURECONFIG: NFC and QR-Code based Hybrid Approach for Smart Sensor Configuration," in *Radio Frequency Identification (RFID), 11th IEEE International Conference on*, pp. 41–46, IEEE, 2017

C. T. Ulz, T. Pieber, C. Steger, S. Haas, R. Matischek, and H. Bock, "Hardware-Secured Configuration and Two-Layer Attestation Architecture for Smart Sensors," in *Digital System Design (DSD), 2017 Euromicro Conference on*, pp. 229–236, IEEE, 2017

D. T. Ulz, T. Pieber, C. Steger, R. Matischek, and H. Bock, "Towards Trustworthy Data in Networked Control Systems: A Hardware-Based Approach," in *Emerging Technologies and Factory Automation (ETFA), 22nd IEEE International Conference on*, pp. 1–8, IEEE, 2017

E. T. Ulz, T. Pieber, C. Steger, S. Haas, and R. Matischek, "Sneakernet on Wheels: Trustworthy NFC-based Robot to Machine Communication," in *Radio Frequency Identification - Technology & Application (RFID-TA), 2017 8th IEEE International Conference on*, pp. 260–265, IEEE, 2017

F. T. Ulz, T. Pieber, A. Höller, S. Haas, and C. Steger, "Secured and Easy-to-Use NFC-Based Device Configuration for the Internet of Things," *IEEE Journal of Radio Frequency Identification (JRFID)*, vol. 1, no. 1, pp. 75–84, 2017

G. T. Ulz, S. Haas, and C. Steger, "Cyber-Physical System and Internet of Things Security: An Overview," in *Solutions for Cyber-Physical Systems Ubiquity*, pp. 248–277, IGI Global, 2018

H. T. Ulz, T. Pieber, C. Steger, S. Haas, and R. Matischek, "QSNFC: Quick and Secured Near Field Communication for the Internet of Things," in *Radio Frequency Identification (RFID), 12th IEEE International Conference on*, pp. 1–8, IEEE, 2018

I. T. Ulz, T. Pieber, C. Steger, A. Höller, S. Haas, and R. Matischek, "Automated Authentication Credential Derivation for the Secured Configuration of IoT Devices," in *Industrial Embedded Systems (SIES), 13th IEEE International Symposium on*, pp. 1–8, IEEE, 2018

J. T. Ulz, M. Feldbacher, T. Pieber, and C. Steger, "Sensing Danger: Exploiting Sensors to Build Covert Channels," in *Information Systems Security and Privacy (ICISSP), Proceedings of the 5th International Conference on*, pp. 100–113, INSTICC, SciTePress, 2019

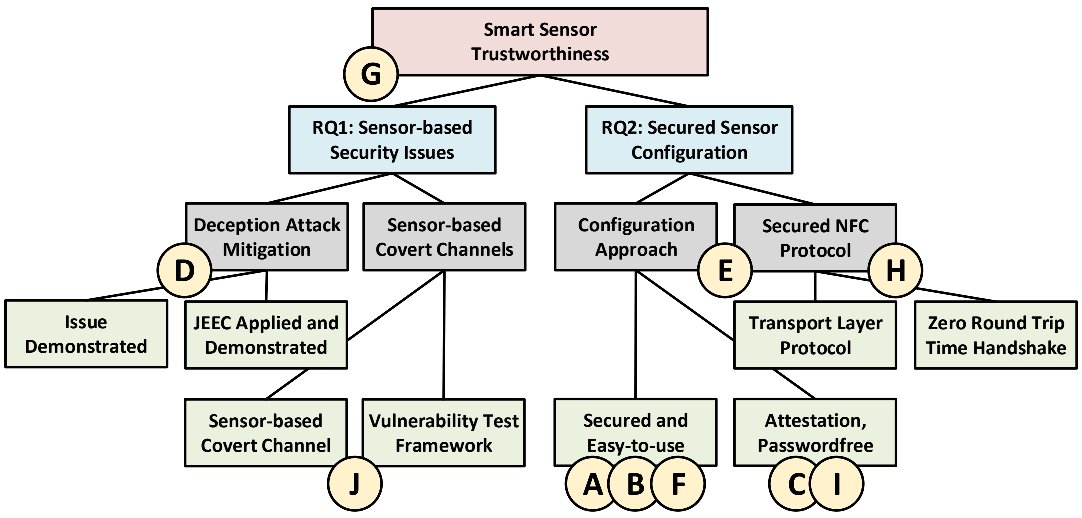

Smart Sensor Trustworthiness
G
RQ1: Sensor-based Security Issues
RQ2: Secured Sensor Configuration
Deception Attack Mitigation
D
Sensor-based Covert Channels
Configuration Approach
E
Secured NFC Protocol
H
Issue Demonstrated
JEEC Applied and Demonstrated
Transport Layer Protocol
Zero Round Trip Time Handshake
Sensor-based Covert Channel
J
Vulnerability Test Framework
Secured and Easy-to-use
A
B
F
Attestation, Passwordfree
C
I

Figure 8.1: Publications this thesis is based on and the contributions made by them.

Bring Your Own Key for the Industrial Internet of Things

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Sarah Haas, Holger Bock, Rainer Matischek
Development Center Graz
Infineon Technologies Austria AG
Graz, Austria
{sarah.haas, holger.bock, rainer.matischek}@infineon.com

***Abstract*—High tech strategies such as Industry 4.0 and Smart Manufacturing require industrial devices to be connected to the Internet. This movement towards interconnected industrial devices poses significant security risks as confidential data must be transferred and stored using untrustworthy channels and cloud servers. End-to-end private key cryptography is suitable to protect the confidentiality, integrity, and authenticity of data. However, private key cryptography has some drawbacks such as the so-called key distribution problem. A possible solution, factory installed keys, are untrustworthy as the two partners relying on end-to-end cryptography can not be sure that no other party is in possession of the used keys. To overcome these problems, the Bring Your Own Key (BYOK) principle based on Near Field Communication (NFC) and dedicated secured hardware is presented in this paper.**

***Index Terms*—Near Field Communication; Industrial Internet of Things; Industry 4.0; cryptography; keys; security controller.**

I. Introduction

The growth of the Industrial Internet of Things (IIoT) is driven by initiatives such as Industry 4.0 [1], Smart Manufacturing [2] or Cloud Manufacturing [3]. All of these initiatives promote the connection of production relevant devices to the Internet to quickly respond to changing customer demands, making them so-called smart factories [4]. Data in such smart factories does not only need to be transferred internally, but also to external partners to increase operational efficiency. Smart factories aim at increasing the operational efficiency through (i) minimizing unplanned downtime of production relevant equipment, (ii) improving the supply chain efficiency,

To decrease downtimes, the necessary maintenance, repair, and operations (MRO) schedules need to be optimized. Maintenance providers, for example, device vendords need to collect and analyze data such as equipment condition or operating hours [5] in order to predict optimal MRO schedules.

To increase supply chain efficiency, it is crucial to use Internet technologies and business-to-business supply chain applications [6]. In an IIoT context this includes the transmission of production data directly to suppliers such that the overhead of supply chain management can be minimized.

To be able to optimally monitor and control the internal production flow, a suitable smart factory architecture as well as protocols need to be chosen. A possible IoT protocol that is also suitable for industrial use cases is the Message Queue Telemetry Transport (MQTT) protocol that was designed for lightweight machine-to-machine communication [7]. MQTT is based on the publish/subscribe principle and through its architecture it is possible to transport data to internal as well as external partners such as maintenance providers [8].

In order to be able to transfer data to external communication partners such as maintenance providers and suppliers as well as to arbitrary internal devices, an MQTT broker that is connected to the Internet or even hosted by a third party can be used. In any case, the transport of confidential and production relevant information through the Internet requires the usage of appropriate cryptographic methods such as end-to-end encryption using TLS.

End-to-end encryption relying on asymmetric cryptography is infeasible for larger amounts of data; therefore, symmetric key cryptography needs to be used. Symmetric cryptography requires both the sender and receiver of the data to be in possession of the same shared key. Because no direct connection between sender and receiver can be established in protocols such as MQTT, key exchange algorithms such as Diffie-Hellman can not be used; a key distribution problem results from this scenario. A device vendor that is in possession of factory installed keys would need to distribute these keys to the equipment customer, who then would need to distribute some of these keys to its suppliers in a scenario such as illustrated in Fig. 1. Moreover, in such a scenario the device vendor would need to be trusted to securely and trustworthy handle all keys. For instance, if the device vendor would be selling devices to competing companies in the same business field, the device vendor would be in possession of keys that could be used for industrial espionage.

To mitigate these key related issues, we propose to apply the Bring Your Own Key (BYOK) principle. Using this principle, keys necessary for end-to-end encryption can be changed by device owners. For example, in the scenario illustrated in Fig. 1, the keys used to protect production relevant data can be changed such that the device vendor is no longer in possession of decryption keys for production data. To allow keys to be deployed and updated in a secured but intuitive manner, we present an NFC based approach that also uses dedicated security controllers (SC) to increase the security of our approach. To the best knowledge of the authors, no such approach was presented previously. Therefore, the contributions of this paper are as follows:

- We present a scenario in that the BYOK principle is applied to solve the problems arising from key distribution and trust issues in factory deployed keys.
- We present a secured and NFC based interface for IIoT devices to deploy and change keys.
- The presented NFC extension for IIoT devices is suitable for new devices and legacy devices alike.

The remainder of this paper is structured as follows. All involved technologies as well as related work to the BYOK principle are discussed in Section II. In Section III, our approach to change keys in an IIoT context is presented. The security implications of the proposed BYOK approach for manufacturing devices are then analyzed by means of a threat analysis in Section IV. A prototypical implementation of our approach is shown in Section V. In Section VI this paper is concluded and possible future work is discussed.

II. Background and Related Work

A. Near Field Communication (NFC)

NFC is a wireless communication technique that is based on a subset of radio-frequency identification (RFID) standards. Because NFC is based on RFID standards, NFC devices are compatible with existing RFID cards and tags [9]. NFC technology is based on inductive coupling and operates at a radio frequency of 13.56 MHz up to a range of approximately 10 centimeters with bit rates up to 848 kbits per second [10]. Connecting NFC devices is fundamentaly different than other technologies such as WiFi, Bluetooth or ZigBee. Two devices automatically establish an NFC connection if they are brought near to each other. Thus, connections need to be (i) actively initiated by a human operator and (ii) the operator typically needs to be in close proximity to the devices. On the one hand, NFC offers security advantages compared to other wireless technologies [11] because of these properties. On the other hand, bringing one device near to another to transfer data is an easy and intuitive principle for humans [12]. In addition, NFC devices can be operated in passive mode which allows NFC devices such as tags or contactless cards to be operated without a battery or power supply [13].

NFC is seen as a promising IoT technology that will *link the real world with the digital world* [14]. Nowadays, NFC (or RFID) is already used for a wide range of applications, the most prominent being the mobile payment sector [15]–[17]. Other application domains of NFC include ticketing [18]–[20], healthcare [21]–[23], or pairing of wireless devices [24], [25].

B. Authenticated Encryption (AE)

AE combines *symmetric cryptography* with *Message Authentication Codes (MAC)* in a secured way such that data confidentiality, integrity, and authenticity can be provided [30]. Symmetric cryptography relies on a shared key for encryption and decryption of data [31]. In our presented approach, the *Advanced Encryption Standard (AES)* is used that is considered to be cryptographically secure using keylenghts of 256 bit [32].

TABLE I
Comparison with related work

Work	Method	Remark
[24]–[26]	Pairing of wireless devices	Approaches provide no or only weak security as information for device pairing is not considered confidential.
[27]	TLS secured key exchange between smart cards	Proposed method for EAP-TLS enabled smart cards. This approach is not suitable for IIoT devices.
[28], [29]	Android device as NFC gateway to Internet	Internet access necessary which might not be possible in all industrial settings. Also, man-in-the-middle attacks could be performed on gateway.
Our approach	NFC device for secured key transport	Security properties discussed in threat analysis.

As MAC algorithm, a *keyed-hash message authentication code (HMAC)* [33] based on SHA-256 is used.

C. Security Controller (SC)

In our presented approach SC are used to offer a protected processing environment as well as secured storage for the transferred keys. SC can be embedded into systems similar to traditional processing units [34]. The property that distinguishes SC from conventional processing units is *tamper resistance* [35]. SC that provide tamper resistance mitigate physical attacks by using appropriate countermeasures that are tested by the *Common Criteria (CC) for Information Technology Security Evaluation* [36].

D. Bring Your Own Key (BYOK)

The BYOK principle originated from the *Bring Your Own Device (BYOD)* idea that allowed employees to use their own mobile phones, tablets and laptops in company networks [37]. These devices need to be secured such that they can be trusted to access a company's confidential data [38].

Similar to BYOD, the BYOK principle allows own keys to be used for cryptographic operations [39], [40]. BYOK is mostly associated with cloud computing, where data is end-to-end encrypted using keys provided by the customer. If in addition to keys also cryptographic methods are provided by a customer, the BYOK principle is extended to *Bring Your Own Encryption (BYOE)* [41].

The establishment of an end-to-end secured channel using keys provided by a BYOK method could be interpreted as a device pairing process as well. The pairing of wireless devices is often assisted by NFC technology [24]–[26]. Urien et al. [27] present an approach to securely exchange tokens between smart cards used in prepayment contexts. Related to keys, Urien and Kiennert [28], [29] introduce an NFC based system to update access authorizations of RFID locks. In their approach they use Android mobile phones to establish a Internet connection via NFC that is used to download keys from a key server to the RFID lock. The Internet connection required in this approach however can be a drawback because

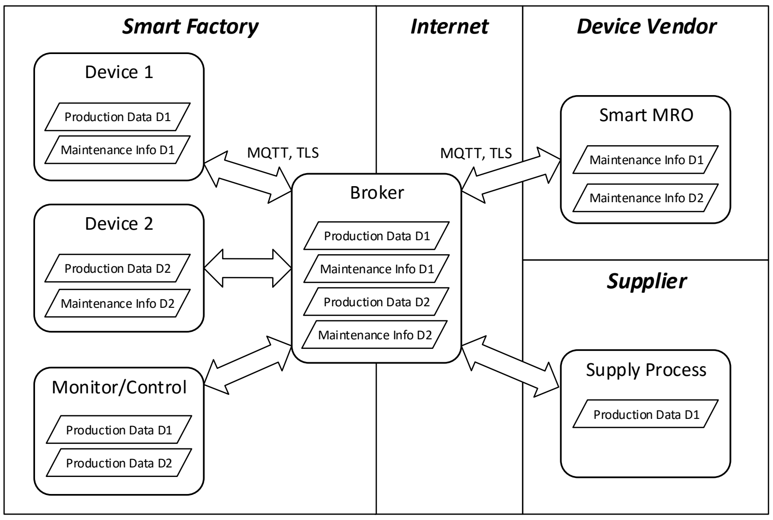


Fig. 1. Example of a smart factory with various publishers and subsribers of data, as envisioned in the Industry 4.0 initiative.

no configuration device tailored for IIoT use cases without network capabilities can be utilized. An overview of related work compared to our presented approach is given in Table I.

III. Bring Your Own Key

In general, the keys that are deployed using the BYOK principle need to be generated first. We propose two different scenarios to generate keys, depending on the trustworthiness of the used mobile device and the corresponding operator. Both scenarios are shown in Fig. 2.

1) If the mobile device and/or the personnel deploying keys are considered untrustworthy, keys are generated at a backend. The key material is then encrypted and transferred to the mobile device, from where the keys can be deployed at the manufacturing devices and the corresponding connection partners. The keys are protected from being extracted and used by an adversary due to the applied encryption.
2) If the mobile device *and* the personnel deploying keys are considered trustworthy, keys can be generated and encrypted directly at the mobile device. The key material then needs to be transferred to the manufacturing device and the corresponding connection partners.

NFC technology is used to transfer key material between devices in our approach. However, NFC does not provide cryptographic protection for transferred data. Therefore, we protect keys transferred in our NFC based BYOK approach by using AE to provide confidentiality, integrity and authenticity for these transferred keys. To encrypt and decrypt data using

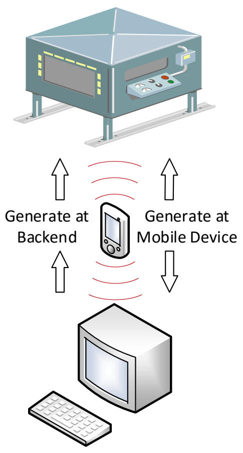


Fig. 2. Deploying new keys at manufacturing device and backend. The keys are either generated at the backend or at the mobile device itself.

AE, an initial key needs to be defined. If this is done by the equipment vendor, these key needs to be send to the equipment customer using a trusted channel. The equipment customer can change these initial keys immediately after delivery of

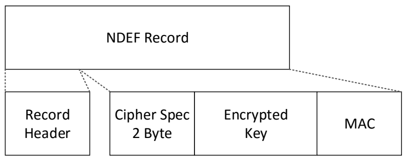


Fig. 3. NDEF Record containing a single transferred key protected using AE.

equipment from the vendor using the BYOK approach. Thus, the device customer is able to take control of their hardware. New keys protected by AE are transferred to the manufacturing equipment using NFC and the NFC Data Exchange Format (NDEF). NDEF packets can contain a number of NDEF Records that contain the actual data. In our approach, a single NDEF Record (see Fig. 3) contains the required header information as well as a cipher spec, the encrypted key and the MAC resulting from the AE encryption process. The cipher spec field contains information on which algorithms to use for decryption and MAC calculation. Encrypted key and MAC are sent sequentially, because the encrypt-then-MAC mode of operation was selected by us due to its security properties [42].

We propose the hardware extension for IIoT devices shown in Fig. 4, to provide the required NFC functionality as well as secured storage and execution environments for manufacturing devices. A host controller is used to connect manufacturing devices to the Internet by providing interfaces to the manufacturing device itself as well as to the Internet. In addition to that host controller, we propose to include a SC that provides an NFC interface as well as tamper resistance. The NFC interface is used to transfer keys from the mobile device to the manufacturing device. In addition to that, the SC can be powered through the NFC field, such that keys can be exchanged even if the manufacturing device is not connected to any power supply. The transferred keys are then decrypted and securely stored in the SC's memory that provides tamper resistance. Thus, it is infeasible for adversaries to extract keys transferred to and stored at the manufacturing device. The SC further provides tamper resistance for the cryptographic operations necessary during end-to-end encrypted data transfer via the Internet.

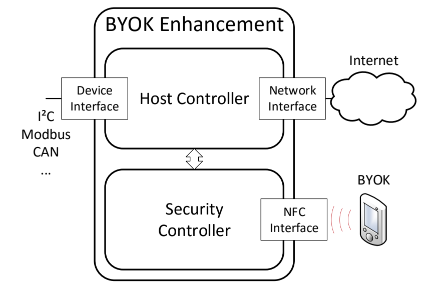


Fig. 4. BOYK enhancement providing interfaces to manufacturing device, the Internet and the mobile device used for key deployment.

IV. Threat Analysis

A threat analysis [43] was conducted to highlight security features and to demonstrate the achieved security level of our presented BYOK approach. This threat analysis lists all involved **Entities (E)**, **Assets (A)** that need to be protected, and **Threats (T)** resulting from our BYOK approach as well as **Countermeasures (C)**, **Residual Risks (R)** and **Assumptions (As)** regarding the threats. For all involved entities, assumptions regarding their trustworthiness are made.

- (E1) *Device Vendor:* (As1) assumed honest but curious
- (E2) *SC Vendor:* (As2) assumed trustworthy
- (E3) *Device Owner:* (As3) assumed trustworthy
- (E4) *External Communication Partners* (e.g. Supplier): (As4) assumed trustworthy
- (E5) *Hoster for Broker:* (As5) assumed untrustworthy
- (E6) *Person deploying Key:* (As6) assumed untrustworthy
- (E7) *Arbitrary Adversary:* (As7) assumed to be able to conduct online and physical attacks

After all entities and the corresponding assumptions are identified, the assets that need to be protected are determined.

- (A1) *Encrypted Data:* The data that is confidential and thus transferred secured by some key.
- (A2) *Keys:* All keys that are stored at any instance in the BYOK process. Loss of a key would result in a loss of confidentiality, integrity and authenticity of (A1).
- (A3) *Device Functionality:* A BYOK interface that is integrated into IIoT devices must not threaten the functionality of these devices in any way. If an adversary is able to harm the functionality of an IIoT device, physical entities and even human lives are threatened.

Considering all identified entities, assumptions, and assets, our presented BYOK approach can now be reviewed concerning potential threats. For each threat, we are going to list countermeasures and/or residual risks if a threat can not be mitigated. For each threat, the involved entities as well as the affected assets are listed as well.

- (T1) *Intentional or unintentional backdoors in device.*
  Entities/Assets: (E1), (E2); (A1), (A2), (A3)
  (C1) Threats investigated in CC EAL5+ certification process for the SC included in involved devices.
- (T2) *Weak or buggy cryptography.*
  Entities/Assets: (E1), (E2); (A1), (A2), (A3)
  (C2) Threats investigated in CC EAL5+ certification process for the SC included in involved devices.
- (T3) *Device vendor loses or distributes keys.*
  Entities/Assets: (E1); (A1), (A2), (A3)
  (C3) Initial keys are changed through BYOK approach. Device vendor does not own actually used keys.
- (T4) *Malicious mobile device or personnel.*
  Entities/Assets: (E6); (A1), (A2), (A3)

1433

(C4) Key material is transported protected by AE, if personnel and/or device are assumed to be untrustworthy.

- (T5) *Wrong keys deployed.*
  Entities/Assets: (E6); (A3)
  (R1) A malicious user that deploys wrong keys or does not update keys and thus attacks communication between devices is similar to a DoS attack that can not be mitigated by our approach.
- (T6) *Device owner does not change initial keys, uses weak keys or loses keys in a security breach.*
  Entities/Assets: (E3); (A1), (A2), (A3)
  (R2) Malicious behaviour by the device owner can not be mitigated by our approach.
- (T7) *Remote attacks targeting IIoT devices.*
  Entities/Assets: (E7); (A1), (A2), (A3)
  (C5) Due to the short communication range of NFC, remote attacks are limited to attackers having physical access to a smart factory.
  (C6) An adversary that is able to communicate using the NFC interface is still not able to apply keys because the encryption key is kept private by the device owner.
  (C7) To mitigate the problem of eavesdropping that is still possible for any wireless technology, the transferred keys are protected using AE.
- (T8) *Physical attacks targeting IIoT devices.*
  Entities/Assets: (E7); (A1), (A2), (A3)
  (C8) Due to the SC providing tamper resistance, extracting key material is considered infeasible for adversaries.
- (T9) *DoS attack using BYOK interface.*
  Entities/Assets: (E7); (A3)
  (C9) Traditional DoS attacks using the BYOK interface are mitigated by the limited bit rate of NFC and the SC handling all involved cryptographic methods. Thus, the whole computational effort will be handled by the SC.

V. Prototype

A prototypical implementation of an end-to-end encrypted data transfer relying on keys provided through our presented BYOK enhancement was implemented to demonstrate the functionality, feasibility and usability of our approach. The setup consists of a mboile device and three Raspberry PI 3, representing a manufacturing device equipped with our BYOK enhancement, a broker and a subscriber respectively as shown in Fig. 5. Similar to the scenario shown in Fig. 1, the manufacturing device is connected to the smart factory's internal network (blue network cable) while the MQTT broker and the supplier are in an external network (yellow network cable). The used mobile device is a Nexus S smart phone with Android 4.1.2 Jelly Bean installed. The BYOK enhancement comprises the following components:

- The used host controller is an Infineon XMC4500 microcontroller from the Cortex M4 family that offers various connection interfaces such as USB, I2C and Ethernet.
- The SC is connected via I2C to the host controller. In our prototype we used an Infineon SLE78 that is CC EAL5+ (high) certified [36] as SC. This SC includes an NFC interface that is able to power the SC and connected devices such as sensors through the NFC field emitted by active NFC devices.

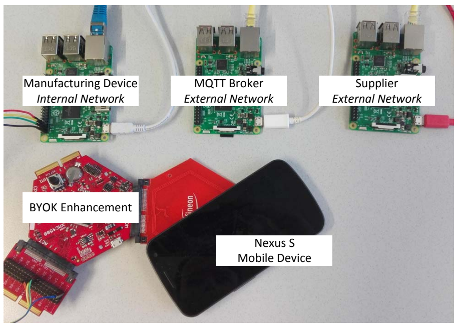


Fig. 5. Prototype setup using a Nexus S mobile device and three Raspberry PI 3, as well as our proposed BYOK enhancement.

VI. Conclusion and Future Work

In this paper we have shown how to apply the BYOK principle to mitigate key related problems arising in the IIoT. This principle, usually applied in cloud computing scenarios, assists in establishing end-to-end encrypted data transfers using IoT protocols such as MQTT. By enabling device owners to change factory deployed keys, this approach helps to increase trust in publishing manufacturing relevant confidential data to the Internet. Using NFC technology to transfer keys is intuitive and offers security advantages compared to other wireless technologies. The proposed BYOK hardware extension allows keys to be deployed using NFC in a secured manner, even if the manufacturing device is without a power supply. We have shown a prototype that highlights the functionality and feasibility of our approach. The presented approach is also shown to be secured against issues that would arise due to including an additional interface into manufacturing devices.

As future work we plan to extend our approach to not only support key material but arbitrary configuration data. As deploying malicious configuration data could lead to physical damage or even threaten human lives, security of transferred configuration data needs to be further improved compared to our current approach.

Acknowledgment

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia.

References

[1] T. Bauernhansl, M. Ten Hompel, and B. Vogel-Heuser, *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung· Technologien· Migration*. Springer-Verlag, 2014.

[2] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, "Smart manufacturing, manufacturing intelligence and demand-dynamic performance," *Computers & Chemical Engineering*, vol. 47, pp. 145–156, 2012.

[3] X. Xu, "From cloud computing to cloud manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 75–86, 2012.

[4] D. Lucke, C. Constantinescu, and E. Westkämper, "Smart Factory - A Step towards the Next Generation of Manufacturing," in *Manufacturing Systems and Technologies for the New Frontier*. Springer, 2008, pp. 115–118.

[5] A. H. Tsang, "Strategic dimensions of maintenance management," *Journal of Quality in Maintenance Engineering*, vol. 8, no. 1, pp. 7–39, 2002.

[6] R. A. Lancioni, M. F. Smith, and T. A. Oliva, "The Role of the Internet in Supply Chain Management," *Industrial Marketing Management*, vol. 29, no. 1, pp. 45–56, 2000.

[7] A. Banks and R. Gupta, "MQTT Version 3.1. 1," *OASIS standard*, 2014.

[8] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian, P. Priller, T. Ebner, T. Ruprechter, and G. Pregartner, "Securing Smart Maintenance Services: Hardware-Security and TLS for MQTT," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE, 2015, pp. 1243–1250.

[9] G. Van Damme, K. Wouters, and B. Preneel, "Practical Experiences with NFC Security on mobile Phones," *Proceedings of the RFIDSec*, vol. 9, p. 27, 2009.

[10] R. Want, "Near Field Communication," *IEEE Pervasive Computing*, vol. 3, no. 10, pp. 4–7, 2011.

[11] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," in *Workshop on RFID security*, 2006, pp. 12–14.

[12] D. López-de Ipiña, J. I. Vazquez, and I. Jamardo, "Touch Computing: Simplifying Human to Environment Interaction through NFC Technology," *1as Jornadas Científicas sobre RFID*, vol. 21, 2007.

[13] H. Mika, H. Mikko, and Y.-o. Arto, "Practical implementations of passive and semi-passive NFC enabled sensors," in *Near Field Communication, 2009. NFC'09. First International Workshop on*. IEEE, 2009, pp. 69–74.

[14] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[15] J. Ondrus and Y. Pigneur, "An Assessment of NFC for Future Mobile Payment Systems," in *Management of Mobile Business, 2007. ICMB 2007. International Conference on the*. IEEE, 2007.

[16] M. Pasquet, J. Reynaud, and C. Rosenberger, "Secure Payment with NFC Mobile Phone in the SmartTouch Project," in *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*. IEEE, 2008, pp. 121–126.

[17] G. Van Damme, K. M. Wouters, H. Karahan, and B. Preneel, "Offline NFC Payments with Electronic Vouchers," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*. ACM, 2009, pp. 25–30.

[18] S. L. Ghìron, S. Sposato, C. M. Medaglia, and A. Moroni, "NFC Ticketing: A Prototype and Usability Test of an NFC-Based Virtual Ticketing Application," in *Near Field Communication, 2009. NFC'09. First International Workshop on*. IEEE, 2009, pp. 45–50.

[19] A. Juntunen, S. Luukkainen, and V. K. Tuunainen, "Deploying NFC Technology for Mobile Ticketing Services Identification of Critical Business Model Issues," in *Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on*. IEEE, 2010, pp. 82–90.

[20] J. Neefs, F. Schrooyen, J. Doggen, and K. Renckens, "Paper Ticketing vs. Electronic Ticketing Based on Off-Line System 'Tapango'," in *Near Field Communication (NFC), 2010 Second International Workshop on*. IEEE, 2010, pp. 3–8.

[21] J. Bravo, D. López-De-Ipiña, C. Fuentes, R. Hervás, R. Peña, M. Vergara, and G. Casero, "Enabling NFC Technology for Supporting Chronic Diseases: A Proposal for Alzheimer Caregivers," in *European Conference on Ambient Intelligence*. Springer, 2008, pp. 109–125.

[22] R. Iglesias, J. Parra, C. Cruces, and N. G. de Segura, "Experiencing NFC-based Touch for Home Healthcare," in *Proceedings of the 2nd international conference on pervasive technologies related to assistive environments*. ACM, 2009, p. 27.

[23] A. Marcus, G. Davidzon, D. Law, N. Verma, R. Fletcher, A. Khan, and L. Sarmenta, "Using NFC-Enabled Mobile Phones for Public Health in Developing Countries," in *Near Field Communication, 2009. NFC'09. First International Workshop on*. IEEE, 2009, pp. 30–35.

[24] E. Uzun, K. Karvonen, and N. Asokan, "Usability Analysis of Secure Pairing Methods," in *International Conference on Financial Cryptography and Data Security*. Springer, 2007, pp. 307–324.

[25] R. Steffen, J. Preissinger, T. Schöllermann, A. Müller, and I. Schnabel, "Near Field Communication (NFC) in an Automotive Environment," in *International Workshop on Near Field Communication*, 2010, pp. 15–20.

[26] L. Chen, G. Pan, and S. Li, "Touch-driven Interaction Between Physical Space and Cyberspace with NFC," in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*. IEEE, 2011, pp. 258–265.

[27] P. Urien, M. Pasquet, and C. Kiennert, "A Breakthrough for Prepaid Payment: End to End Token Exchange and Management Using Secure SSL Channels Created by EAP-TLS Smart Cards," in *Collaboration Technologies and Systems (CTS), 2011 International Conference on*. IEEE, 2011, pp. 476–483.

[28] P. Urien and C. Kiennert, "A New Key Delivering Platform Based on NFC Enabled Android Phone and Dual Interfaces EAP-TLS Contactless Smartcards," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2011, pp. 387–394.

[29] ——, "A New Keying System for RFID Lock Based on SSL Dual Interface NFC Chips and Android Mobiles," in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2012, pp. 42–43.

[30] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.

[31] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption," in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*. IEEE, 1997, pp. 394–403.

[32] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Science & Business Media, 2013.

[33] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104 (Informational), Internet Engineering Task Force, Feb. 1997.

[34] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Moderator-Ravi, "Security as a New Dimension in Embedded System Design," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 753–760.

[35] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper Resistance Mechanisms for Secure Embedded Systems," in *VLSI Design, 2004. Proceedings. 17th International Conference on*. IEEE, 2004, pp. 605–611.

[36] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[37] G. Thomson, "BYOD: enabling the chaos," *Network Security*, vol. 2012, no. 2, pp. 5–8, 2012.

[38] A. M. French, C. Guo, and J. Shim, "Current Status, Issues, and Future of Bring Your Own Device (BYOD)," *Communications of the Association for Information Systems*, vol. 35, no. 10, pp. 191–197, 2014.

[39] H. Zhang, "Bring your own encryption: balancing security with practicality," *Network Security*, vol. 2015, no. 1, pp. 18–20, 2015.

[40] S. Syed and M. Ussenaiah, "The Rise of Bring Your Own Encryption (BYOE) for Secure Data Storage in Cloud Databases ," in *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on*. IEEE, 2015, pp. 1463–1468.

[41] S. McGrath, "The Rise Of Bring Your Own Encryption - InformationWeek," http://www.informationweek.com/interop/the-rise-of-bring-your-own-encryption-/a/d-id/1320796, 9 2015, (Accessed on 12/28/2016).

[42] H. Krawczyk, "The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?))," in *Annual International Cryptology Conference*. Springer, 2001, pp. 310–331.

[43] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat Modeling as a Basis for Security Requirements," in *Symposium on requirements engineering for information security (SREIS)*, vol. 2005. Citeseer, 2005, pp. 1–8.

2017 IEEE International Conference on RFID (RFID)

SecureConfig: NFC and QR-Code based Hybrid Approach for Smart Sensor Configuration

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Christian Lesjak, Holger Bock, Rainer Matischek
Development Center Graz
Infineon Technologies Austria AG
Graz, Austria
{christian.lesjak, holger.bock, rainer.matischek}@infineon.com

***Abstract*—In *smart factories* and *smart homes*, devices such as *smart sensors* are connected to the Internet. Independent of the context in which such a smart sensor is deployed, the possibility to change its configuration parameters in a secure way is essential. Existing solutions do provide only minimal security or do not allow to transfer arbitrary configuration data. In this paper, we present an NFC- and QR-code based configuration interface for smart sensors which improves the security and practicability of the configuration altering process while introducing as little overhead as possible. We present a protocol for configuration as well as a hardware extension including a dedicated security controller (SC) for smart sensors. For customers, no additional hardware other than a commercially available smartphone will be necessary which makes the proposed approach highly applicable for smart factory and smart home contexts alike.**

***Index Terms*—Near Field Communication; Internet of Things; smart sensor; configuration; security controller.**

I. Introduction

For *smart sensors* [1] that are connected to the Internet it is crucial that their configuration and firmware can be updated in a secured and efficient way. Such smart sensors can be deployed in a wide range of fields such as in a *smart factory* or in a *smart home*.

Smart Factory [2]: In smart factories it is essential to perform maintenance operations of sensors involved in the production prcoess. By introducing a secured and easy to use configuration interface, even untrained staff can perform firmware updates or configuration changes. However, it is very important to protect the confidentiality and authenticity of configuration data as employees applying the configuration updates could be potential adversaries. By enabling any employee or external person to perform configuration operations, the flexibility of the already deployed sensors will be increased while the associated maintenance costs will be decreased [3].

Smart Home [4]: Also in a smart home context, configuration and firmware updates for devices need to be performed using a secured and easy to use configuration interface. Devices not only include smart sensors but also other electronic devices such as WiFi routers. Similar to the smart factory use-case, also in a smart home context the configuration data must be secured against various attacks for sustaining the proper functionality of the configured devices. A configuration interface included into smart home devices enables any customer to perform firmware and configuration updates. These updates could, for instance, even be provided by a vendor's helpdesk.

By including such configuration interfaces into smart sensors, also the *Bring Your Own Key (BYOK)* principle [5] can easily be applied in both the smart factory and smart home context. BYOK would allow customers to change vendor supplied cryptographic keys, and thus, give them the certainty that no third party is able to access their data.

The approach presented in this paper not only is able to transfer cryptography keys but also arbitrary configuration data and firmware updates. To transfer data, NFC technology is chosen for three reasons. (i) NFC offers some security advantages compared to other wireless technologies [6]. Also, certain kinds of attacks such as man-in-the-middle are harder to conduct due to the limited communication range of NFC. (ii) The update process can be performed without an internal power source, if the necessary hardware is powered by the NFC field. (iii) NFC is easy and intuitive to use. Humans easily understand the principle of bringing one device near to another to transfer data [7].

If NFC is used to transfer data from a backend to a mobile device and from the mobile device to smart sensors, at least three NFC-enabled devices would be necessary. While smart sensor and mobile device must be equipped with an NFC interface in any case, needing an additional NFC-enabled device such as an NFC reader for the backend is inconvenient at least in a smart home context. Therefore, a combination of NFC and QR-Codes is used in the approach presented in this paper. The presented approach also relies on the functionalities provided by a security controller (SC). We propose to use a SC to protect the confidentiality and authenticity of configuration data that is stored on the SC. To the best knowledge of the authors, no other publication described a combination of these techniques to perform updates for smart sensors. The main contributions of this paper are:

1) The presented configuration approach allows arbitrary configuration data including firmware updates to be transferred in a secured manner.
2) The presented approach therefore is suitable for industrial as well as smart home use-case scenarios.

978-1-5090-4576-1/17/$31.00 © 2017 IEEE

3) A hardware extension and cryptographic methods are presented that are applicable also for legacy sensors.

The remainder of this paper will be structured as follows. In Section II, background information on the used technologies as well as related work will be given. Section III presents the proposed NFC- and QR-Code based configuration interface for smart sensors. This interface will be evaluated with respect to security and the imposed overhead in Section IV. This paper will then be concluded with Section V where possible future work is stated as well.

II. Background and Related Work

A. Near Field Communication (NFC)

NFC is a contactless communication technology that is based on RFID standards. It operates at a radio frequency of 13.56 MHz, up to a range of approximately 10 cm with supported bit rates of 106, 212, 424 and 848 kbits per second (kbps). Also, NFC devices are compatible with many existing RFID devices and tags as the NFC standard comprises various RFID standards [8]. NFC is used in a diverse range of businesses. Today, the most well-known application of NFC is in the mobile payment sector [9]. Coskun et al. [10] note that NFC is also widely used in mobile ticketing applications. Another prominent field for NFC is the Internet of Things (IoT). Atzori et al. [11] state that *NFC [...] together with RFID [...] will link the real world with the digital world.*

B. Quick Response (QR) Code

A QR code is a two-dimensional code that offers various advantages over traditional (linear) barcodes such as a much higher data density or the possibility to read QR codes from all directions. The higher density allows a maximum capacity of 2953 bytes. Although the encryption of a QR code's content is possible, encrypted QR codes can be rarely found [12]. Therefore, Conde-Lagoa et al. [13] suggest to encrypt the content using symmetric cryptography. Soon [14] lists sample applications such as ticketing or identification of all sorts of items.

C. Security Controller (SC)

SCs are used to provide protected processing and storage of data. The key differentiator when compared to traditional processing units is the *tamper resistance* [15] of such SCs which even includes invasive attacks utilizing physical access to the hardware. However, as SCs are not as powerful as general-purpose controllers, splitting the execution environment into a *secured world* and into a *normal world* is suggested for instance by Vasudevan et al. [16]. This splitting principle, also called *security by isolation* or *dual-execution* [17], is realized by implementing SCs as external hardware modules.

D. Configuration via NFC

Configuring devices via an NFC interface is quite a novel topic. Wu et al. [18] discuss the possibility to reprogram computational RFIDs (CRFIDs) over the air. In their approach, the firmware of passive CRFID tags is reprogrammed using the Electronic Product Code (EPC) protocol. Haase et al. [19] present an NFC based configuration solution for sensors and actuators in the home automation context. The authors propose to extend existing hardware with an NFC module that can then be used as a configuration interface for standard smart phones. The authors present a hardware concept and prototype as well as a mobile application for Android smart phones. Serfass and Yoshigoe [20] present a framework for NFC communication in wireless sensor networks. This Android based framework, according to the authors, would allow P2P transport of arbitrary data. In contrast, a more widely used approach is to use NFC to speed up the pairing of wireless devices [21], [22]. A comparison of related work to the approach presented in this paper is given in Table I. As can be seen there, all but one approach do not consider security at all. The approach presented by Haase et al. [19] mentions encryption but only encrypts the data while it is transferred via NFC. The data, however, can be read and changed on the Android smartphone. Also, no solution to transfer the configuration data to the smartphone is given in that work.

TABLE I
Comparison with related work

Related Work	Necessary Hardware	Supported Payload	Security Considerations
[18]	RFID Card Reader, CRFIDs	Firmware Only	None
[19]	NFC-enabled Phone, and Sensor	Arbitrary Data	Encryption used except for initial update; No encryption on mobile device
[20]	At least 2 Android Devices for P2P	Arbitrary Data	None
[21], [22]	RFID tags, 2 NFC devices to pair	Pairing information	None
This Work	NFC-enabled phone and sensor	Arbitrary Data	Discussed in Section III

III. SecureConfig

Configuring smart sensors in a smart factory as well as in a smart home context is desirable. For a solution that is suitable for both contexts, a couple of requirements need to be fulfilled. To be usable in a smart factory context, a central instance that manages all active configurations is needed. In a smart home context, no additional hardware besides a mobile device and sensors should be necessary to make the proposed approach feasible for many users. Therefore, the system architecture shown in Fig. 1 is proposed. It comprises three components.

1) *Backend:* Configurations are created, updated and securely stored at the backend.
2) *Mobile Device:* The mobile device is used to transfer configuration data provided by the backend to the smart sensor. In our prototype we used a smartphone.
3) *Smart Sensor:* The smart sensor receives the provided configuration update.

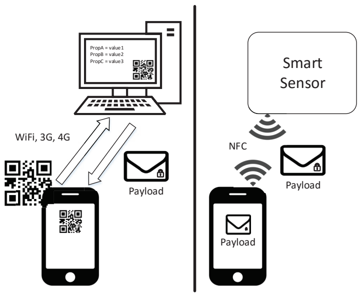


Fig. 1. Hybrid communication approach: On the left hand side, a configuration package is fetched from the backend using a QR code. On the right hand side, the configuration of a smart sensor using NFC is shown.

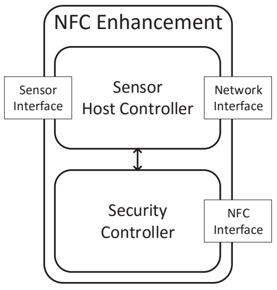


Fig. 2. NFC Enhancement component which can be connected to any conventional sensor via the *Sensor Interface* and thus making it a smart sensor.

In the context of this paper, it is assumed that the data stored on the *backend* is secured by appropriate security mechanisms such as a hardware security module (HSM), and thus, data stored at the backend is efficiently protected from loss or manipulation.

A. NFC Enhancement

To equip any arbitrary sensor with NFC capability and a SC, we propose a hardware component named *NFC Enhancement*. The component is shown in Fig. 2. As can be seen there, it comprises two controllers and various interfaces. The reasons for suggesting a dedicated NFC enhancement module are:

1) By designing a dedicated hardware module with an explicit interface to sensors, currently available (legacy) sensors can easily be transformed into a smart sensor. The NFC enhancement module can easily be offered as a single PCB which is easy to integrate for sensor vendors.
2) By including two controllers, responsibilities can be split perfectly according to the properties of both controllers. The sensor host controller provides interfaces to the sensor and optionally to a network while also offering computational power and memory for any kind of application. The less powerful but energy efficient SC on the other hand offers a secured execution environment and protected storage for configuration data as well as an NFC interface.
3) The NFC interface connected to the SC allows for ad-hoc connectivity instead of opening the configuration interface to a potential network connection. Also, the SC can be powered through the NFC field which allows for configuration updates independent of the sensor's and host controller's power supply.

B. Hybrid Communication Approach

As shown in Fig. 1, different technologies for data transfer are proposed in our approach. To transfer configuration data from the backend to the mobile device, QR codes are used. The configurations stored on the mobile device are then transferred to the smart sensor using NFC. The reasons for using this hybrid approach are:

1) By using QR codes to transfer configurations to the mobile device, no additional hardware (aside from the mobile device) such as an NFC reader is needed by customers. Configurations are imported by simply scanning the QR codes. This makes our approach especially suitable for smart home contexts while not limiting its usefulness in industrial contexts. Configurations could be printed for maintenance workers or displayed in web based configuration interfaces for customers.
2) NFC is suggested to transfer configuration data from the mobile device to the smart sensor. Reasons for using NFC for this data transmission are the additional security resulting from the limited communication range as well as the possibility to also configure sensors that are disconnected from their power supply. This also allows the initial configuration of sensors by the vendor during their assembly where no power supply is available. This property adds additional usefulness to our approach.

As a result of using two different technologies for data transfer, two separate data structures and methodologies need to be applied which are discussed for both variants.

C. Quick Response Code

Due to the size limitations of a QR code's maximum payload, two different modes for transferring configuration data to the mobile device are suggested.

1) The whole configuration payload is stored in the QR code, which allows to store about 2900 bytes of data. We denote this type as *inline* QR code. Inline QR codes do not require the mobile device to have an active network connection, thus, those codes can be distributed, for instance, to a maintenance worker without restrictions.
2) If the configuration data is larger than the size limit of 2900 bytes, only an URL pointing to the backend is

included in the QR code. The mobile device then needs to fetch the configuration data from the backend using a secure channel (TLS). This type is denoted as *URL* QR code. For the download process, the mobile device needs to have a network connection through which the backend can be reached.

The type that is used to transport configuration data however, does not solely depend on the configuration data's size. A second factor is the desired security level, as no active connection to the backend is needed for the *inline* type. Therefore, some of the security measures mentioned in Section III-E can not be applied.

D. Near Field Communication

To transfer configuration data via NFC, the *NFC Data Exchange Format (NDEF)* [23] that uses the NFC Forum reader/writer mode is used. NDEF abstracts the contactless communication and is supported by mobile platforms such as Android [24]. The proposed structure for NDEF packages is shown in Fig. 3. As can be seen there, various security related fields are included in addition to the (encrypted) configuration data.

E. Security Measures

To provide confidentiality, integrity, and authenticity of the transferred configuration data, *authenticated encryption (AE)* [25] is used. As can be seen in the NDEF packet structure shown in Fig. 3, the transferred packet comprises a couple of security related fields as well as the encrypted payload, and a MAC; the later two are calculated by applying AE. The AE method of operation considered as having the best security properties is *encrypt-then-MAC* [26] which is the reason why this approach is used in our work. When using AE it is also important to not use the same key for both encryption and hashing; therefore, a cryptographic key derivation [27] is applied to generate separate encryption and hashing keys from a master key.

In addition to the aforementioned cryptographic principles, additional information regarding the configuration data is included in the NDEF message (see Fig. 3). This information is used by the SC at the smart sensor to decide if a configuration update is rejected or accepted and consequently applied. As the confidentiality, integrity, and authenticity of this information also needs to be protected, all but two fields are included in the encryption process. The two unencrypted fields are:

- Realtime: The time in milliseconds since the mobile device was started.
- Cipher Spec: Specifies the applied cryptographic algorithms for the authenticated encryption.

The fields which are included in the encrypted payload are:

- Version: The configurations version number. A smart sensor will reject configuration updates with a configuration number less or equal to the currently applied configuration.
- Validity: If the transmitted *realtime* is later than the specified *validity*, a configuration update will be rejected.

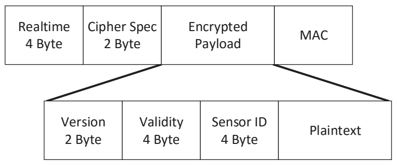


Fig. 3. NDEF packet structure. *Realtime* and *Cipher Specs* are transferred unencrypted. The size of the attached MAC depends on the cipher specs.

- Sensor ID: If the specified sensor ID does not match, the configuration update is rejected.

As there is no time synchronization between the backend and the smart sensor, the process of verifying the configuration's *validity* needs to be discussed in detail. Whenever a configuration is fetched from the backend, the following steps are performed:

1) For each configuration, a validity period Δ needs to be specified at the backend.
2) The mobile device sends a request to the backend, containing the current realtime ϑ.
3) Upon encrypting the configuration data, the included *validity* $\nu = \vartheta + \Delta$ is calculated.
4) The encrypted configuration data is sent to the mobile device.

For our approach to function properly, we assume a secured time source in the mobile device. In the case of an *inline* QR code, no connection to the backend is established; therefore, no validity can be specified for the included configuration data. Due to this, the *inline* mode needs to be considered as less secure than the *URL* mode.

IV. Evaluation

A prototype was realized to evaluate the feasibility, usability, and functionality of the presented approach. This prototype, pictured in Fig. 4, contains the following components:

1) *Sensor:* An air pressure sensor is used in this prototype to demonstrate the configuration update process.
2) *NFC Enhancement:* The NFC enhancement prototype that was realized is based on a concept presented by Lesjak et al. [3] that uses an Infineon XMC4500 microcontroller (Cortex M4 family) as the general purpose controller. This controller offers connection interfaces such as USB and Ethernet, as well as I2C. Via this I2C interface, a common criteria [28] EAL5+ certified SC by Infineon is connected to the XMC4500. This SC provides security features such as secured data storage and code execution by using a self-checking dual CPU concept, integrity checks for data transfers and caches, and encrypted memory and calculations in the CPU. Furthermore, this SC also includes a contactless interface capable of NFC communication. The NFC antenna is integrated into the NFC enhancement module as well.

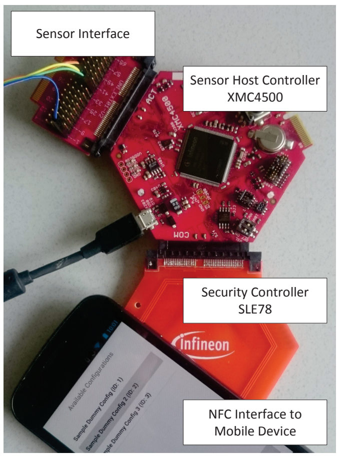

Sensor Interface
Sensor Host Controller XMC4500
Security Controller SLE78
NFC Interface to Mobile Device

Fig. 4. NFC enhancement prototype.

3) *Mobile Device:* A Nexus S smartphone was used as NFC-enabled device in the presented prototype. On this device, Android 4.1.2 Jelly Bean was installed to use the latest NDEF functionality included with API level 14.
4) *Backend:* The backend was realized on a standard Windows PC in this prototype.

This prototype then was used to measure the time necessary for an update process. A configuration update containing 64 byte of data, for example, took roughly 200ms on average which is similar to the time a TLS handshake would need on such hardware.

A. Threat Analysis

To demonstrate the achieved security level, a threat analysis which highlights **Entities (E)**, **Assets (A)**, **Threats (T)**, applied **Countermeasures (C)**, and **Residual Risks (R)** is conducted. Due to the higher security offered by the *URL* QR code, this mode is discussed in this threat analysis. An overview of the threat analysis in *goal structure notation (GSN)* is shown in Fig. 5. The attack possibilities that are analysed are the smart sensor interface as well as the mobile device which is seen as a data channel. The backend is assumed to be properly secured by measures such as an appropriate firewall and a HSM, the SC at the smart sensor is assumed to be certified to the security level EAL5+ according to the common criteria. The assets that need to be protected are **configuration data (A1)** and **sensor functionality (A2)**. Threats can be posted by the **NFC enhancement vendor (E1)**, **customer (E2)**, **mobile device user (E3)** and an **external adversary (E4)**.

Threats resulting from **intentional or unintentional backdoors (T1)**, **weak cryptographic algorithms (T2)** and **bugs in cryptographic algorithms (T3)** by the **vendor (E1)** are investigated in the **common criteria EAL5+ certification process (C1)** for the included SC. The initial encryption keys specified for each SC could be **lost in a security breach (T4)** or even **disclosed in any form (T5)** by the **vendor (E1)**. This can be mitigated by **changing the initial key (C2)** as part of a configuration by the **customer (E2)**. Any malicious **mobile device user (E3)** could try to **manipulate configuration data (T6)**, **try to apply outdated configurations (T7)** or **try to apply configurations to wrong sensors (T8)**. **The presented security measures (C3)**, however, provide efficient mitigation of these threats. If the person responsible to update configurations **(E3) does not apply the configuration at all (T9)**, a potential denial of service attack results if the sensor's functionality is influenced by the missing configuration. There currently is no security measure implemented to **counteract missing updates (R1)**. If the malicious **mobile device user (E3)** or an **adversary (E4)** with physical access to the sensor continuously tries to change a configuration which is rejected by the SC, a possible **DoS attack (T10)** could result. There is currently no security measure implemented against this kind of attack **(R2)**. Attacks that **passively try to eavesdrop (T11)** configuration data are efficiently mitigated by the **implemented security measures (C3)** and the **security features of NFC (C4)**.

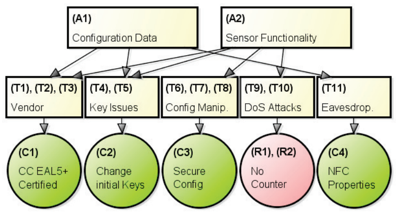


Fig. 5. Overview of threat analysis in GSN.

B. Overhead

The overhead resulting from the implemented security measures can be split into a static and into a variable part. The static overhead, resulting from the information added to the encrypted configuration data and MAC, can easily be calculated by summing up all fields with specified sizes in Fig. 3. The resulting static overhead is $O_{static} = 16$ bytes. The variable overhead depends on the chosen cryptographic algorithms. For this evaluation, *HMAC-SHA256* is assumed as hashing algorithm which adds an additional overhead of $O_{variable} = 32$ bytes. The resulting total overhead in that case would be $O = O_{static} + O_{variable} = 48$ bytes. An overview of the overhead relative to the configuration data size up to 4 kB of data is shown in Fig. 6. As can be seen there, when transferring configuration data of about 300 bytes, less than 15% of the transferred data will be security imposed overhead.

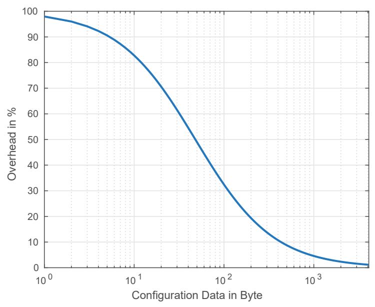


Fig. 6. Overhead in percent relative to transferred configuration data.

V. Conclusion and Future Work

In this paper we present an NFC- and QR-code based hybrid configuration approach for smart sensors which is suitable for smart factory and smart home use cases. To provide the necessary functionality for sensors, an *NFC enhancement* module is presented. To mitigate potential security challenges imposed by such an additional configuration interface, appropriate security measures are included in our approach. It is also shown that by including those security measures, an acceptable amount of overhead is imposed. The feasibility of our approach is demonstrated as a prototype which is presented in this work. As future work we plan to include a password based key exchange protocol such as SPAKE [29] to require user authentication when applying updates. Authenticated users could then read configuration parameters from a smart sensor, directly modify them on their mobile device and update the smart sensor's configuration.

Acknowledgment

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia.

References

[1] G. Meijer, K. Makinwa, and M. Pertijs, *Smart Sensor Systems: Emerging Technologies and Applications*. John Wiley & Sons, 2014.
[2] D. Lucke, C. Constantinescu, and E. Westkämper, "Smart Factory - A Step towards the Next Generation of Manufacturing," in *Manufacturing Systems and Technologies for the New Frontier*. Springer, 2008, pp. 115–118.
[3] C. Lesjak, T. Ruprechter, H. Bock, J. Haid, and E. Brenner, "Facilitating a Secured Status Data Acquisition from Industrial Equipment via NFC," *Journal of Internet Technology and Secured Transactions (JITST)*, 2014.
[4] R. Harper, *Inside the Smart Home*. Springer Science & Business Media, 2006.
[5] H. Zhang, "Bring your own encryption: balancing security with practicality," *Network Security*, vol. 2015, no. 1, pp. 18–20, 2015.
[6] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," in *Workshop on RFID security*, 2006, pp. 12–14.
[7] D. López-de Ipiña, J. I. Vazquez, and I. Jamardo, "Touch Computing: Simplifying Human to Environment Interaction through NFC Technology," *1as Jornadas Científicas sobre RFID*, vol. 21, 2007.
[8] G. Van Damme, K. Wouters, and B. Preneel, "Practical Experiences with NFC Security on mobile Phones," *Proceedings of the RFIDSec*, vol. 9, 2009.
[9] J. Ondrus and Y. Pigneur, "An Assessment of NFC for Future Mobile Payment Systems," in *Management of Mobile Business, 2007. ICMB 2007. International Conference on the*. IEEE, 2007.
[10] V. Coskun, B. Ozdenizci, and K. Ok, "A Survey on Near Field Communication (NFC) Technology," *Wireless personal communications*, vol. 71, no. 3, pp. 2259–2294, 2013.
[11] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
[12] V. Sharma, "A Study of Malicious QR Codes," *International Journal of Computational Intelligence and Information Security*, vol. 3, no. 5, pp. 21–26, 2012.
[13] D. Conde-Lagoa, E. Costa-Montenegro, F. Gonzalez-Castao, and F. Gil-Castieira, "Secure eTickets Based on QR-Codes with User-Encrypted Content," in *2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE)*, 2010.
[14] T. J. Soon, "QR Code," *Synthesis Journal*, vol. 2008, pp. 59–78, 2008.
[15] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper Resistance Mechanisms for Secure Embedded Systems," in *VLSI Design, 2004. Proceedings. 17th International Conference on*. IEEE, 2004, pp. 605–611.
[16] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J. M. McCune, "Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give Me?" in *International Conference on Trust and Trustworthy Computing*. Springer, 2012, pp. 159–178.
[17] M. Sabt, M. Achemlal, and A. Bouabdallah, "The Dual-Execution-Environment Approach: Analysis and Comparative Evaluation," in *IFIP International Information Security Conference*. Springer, 2015, pp. 557–570.
[18] D. Wu, M. J. Hussain, S. Li, and L. Lu, "R2: Over-the-Air Reprogramming on Computational RFIDs," in *RFID (RFID), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–8.
[19] J. Haase, D. Meyer, M. Eckert, and B. Klauer, "Wireless sensor/actuator device configuration by NFC," in *2016 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2016, pp. 1336–1340.
[20] D. Serfass and K. Yoshigoe, "Wireless Sensor Networks Using Android Virtual Devices and Near Field Communication Peer-To-Peer Emulation," in *Southeastcon, 2012 Proceedings of IEEE*. IEEE, 2012, pp. 1–6.
[21] A. Matos, D. Romao, and P. Trezentos, "Secure Hotspot Authentication through a Near Field Communication Side-Channel," in *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2012, pp. 807–814.
[22] M. Jung, J. G. Park, J. H. Kim, and J. Cho, "Interoperability between Medical Devices using Near Field Communication," in *2013 International Conference on Information Science and Applications (ICISA)*. IEEE, 2013, pp. 1–4.
[23] "NFC Data Exchange Format (NDEF)," NFC Forum, Tech. Rep. NDEF 1.0, 07 2006.
[24] A. Lotito and D. Mazzocchi, "OPEN-NPP: an open source library to enable P2P over NFC," in *Near Field Communication (NFC), 2012 4th International Workshop on*. IEEE, 2012, pp. 57–62.
[25] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.
[26] H. Krawczyk, "The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?))," in *Annual International Cryptology Conference*. Springer, 2001, pp. 310–331.
[27] ——, "Cryptographic Extraction and Key Derivation: The HKDF Scheme," in *Annual Cryptology Conference*. Springer, 2010, pp. 631–648.
[28] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.
[29] M. Abdalla and D. Pointcheval, "Simple Password-Based Encrypted Key Exchange Protocols," in *Cryptographers Track at the RSA Conference*. Springer, 2005, pp. 191–208.

2017 Euromicro Conference on Digital System Design

Hardware-Secured Configuration and Two-Layer Attestation Architecture for Smart Sensors

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Sarah Haas, Rainer Matischek, Holger Bock
Development Center Graz
Infineon Technologies Austria AG
Graz, Austria
{sarah.haas, rainer.matischek, holger.bock}@infineon.com

***Abstract*—The necessity to (re-)configure Internet of Things devices such as smart sensors during their entire lifecycle is becoming more important due to recent attacks targeting these devices. Allowing configuration parameters to be changed in any phase of a smart sensor's lifecycle allows security updates or new key material to be applied. Also, the functionality of a smart sensor can be altered by changing its configuration. The challenges that need to be considered when enabling the configuration of arbitrary parameters are the security and usability of the configuration interface, the secured storage of confidential configuration data, and the attestation of successfully applied configuration updates. Therefore, we present an NFC-based configuration approach that relies on dedicated secured hardware to solve these challenges. In addition to a hardware extension for smart sensors, we also present a secured configuration protocol as well as a two-layer configuration attestation process to verify the correct utilization of all transmitted configuration parameters.**

***Index Terms*—Smart Sensor; Configuration; Attestation; Hardware Security.**

I. Introduction

Sensors are seen as one of the major building blocks of the Internet of Things (IoT) [1] where devices can be used to interact with their physical environment only due to embedded sensors. These sensor-equipped devices have led to technologies such as wireless sensor networks (WSN) and high-tech strategies such as Industry 4.0. In these technologies and high-tech strategies, sensing nodes often perform some sort of (pre-)processing in order to optimize properties such as their exactness, energy efficiency, or usefulness. Such sensors are also denoted as *smart sensors* [2].

Due to recent attacks targeting these devices, frequent re-configuration is needed to mitigate certain kinds of attacks [3], [4]. For example, frequent changes of applied encryption keys or parameters such as a used elliptic curve could make attacks harder. Also, security related software updates will be needed to account for new security requirements. In addition to security related updates, also updated functionality of devices can be achieved. Rapidly changing environments, as well as frequently updated requirements regarding their operation, require smart sensors to be configurable. Weyer et al. [5] state that configuring devices will be essential for future Industry 4.0 motivated production systems.

One way to achieve the goal of flexible smart sensors is to make them self-configuring and adaptive [6]. Lee et al. [7] suggest self-configuration and self-adjustment as one of five major building blocks for cyber-physical systems (CPS) used in Industry 4.0 scenarios. However, self-configuration of smart sensors is not considered as mature enough to account for all requirements of industrial scenarios where higher safety and security standards need to be fulfilled. Therefore, manual configuration mechanisms that are reliable while providing a secured update process will be needed for smart sensors.

The European research project *IoSense*[1] addresses the configurability of smart sensors. As envisioned in the IoSense project, the configuration of smart sensors should be possible throughout the complete lifecycle of a sensor. The four phases of a smart sensor's lifecycle and potential example use-cases where a (re)configuration is needed are shown in Fig. 1.

To allow smart sensors to be configured during all four shown lifecycle phases, we propose a Near Field Communication (NFC)-based configuration approach that uses a dedicated hardware-based secure element to provide a protected execution environment for involved security critical code as well as secured storage for confidential configuration data. For non-confidential configuration data, storage will be provided by a general purpose computing environment. Due to these different storage layers and to provide efficient configuration attestation with minimal communication overhead, we also propose a two-layer configuration attestation architecture. Summarized, the contributions of this paper are:

- We present an NFC-based configuration approach suitable for smart sensors used in industrial environments. To account for the enhanced security requirements of industrial scenarios, a hardware architecture using dedicated hardware-based secure elements in combination with suitable cryptographic methods are used to provide data confidentiality, integrity, and authenticity.
- Due to providing unsecured as well as secured storage for configuration data, and to impose an overhead as small as possible, a two-layer configuration attestation architecture is presented in this paper. The configuration protocol is *attestation aware* to support the configuration attestation.

[1]http://www.iosense.eu

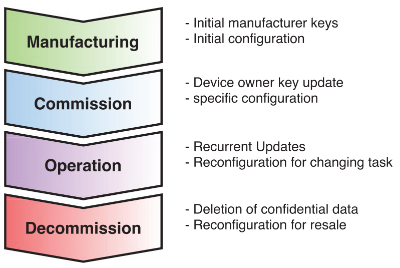

Manufacturing
- Initial manufacturer keys
- Initial configuration
Commission
- Device owner key update
- specific configuration
Operation
- Recurrent Updates
- Reconfiguration for changing task
Decommission
- Deletion of confidential data
- Reconfiguration for resale

Fig. 1. Four phases of a sensor's lifecycle with configuration scenarios.

TABLE I
COMPARISON WITH DEVICE CONFIGURATION RELATED WORK.

Related Work	Energy Efficient	Arbitrary Payload	Secured Protocol	Tamper Resistant	Attes-tation
[8], [9]	✗	✓	✓	✗	✗
[10]	✗	✓	✓	✗	✗
[11]	✓	✗	✗	✗	✗
[12]	✓	✓	✗	✗	✗
[13]	✓	✓	✓	✗	✗
[14]	✓	✗	✓	✓	✗
This work	✓	✓	✓	✓	✓

To the best knowledge of the authors, neither an attestation aware configuration approach nor a two-layer configuration attestation architecture were proposed in other works. The remainder of this paper is structured as follows. In Section II, background on involved technologies as well as related work is discussed. The general configuration and attestation problem, as well as a system model and corresponding assumptions, are discussed in Section III. Section IV demonstrates our proposed hardware-secured configuration approach, Section V presents the corresponding two-layer configuration attestation architecture. The presented approach is then evaluated in Section VI by means of a demonstrator. This paper is then concluded with Section VII.

II. Background and Related Work

A. Device Configuration

Device configuration is an important topic in the IoT due to the large number of devices. Many solutions have been proposed (e.g. [8], [9]) that define interfaces such that devices can be easily configured via the Internet. However, as most of these solutions expose the configuration interface of devices to the Internet without considering protocol *and* physical device security, they are not suitable for industrial scenarios. Steger et al. [10] propose a software update approach for vehicles that relies on 802.11s mesh networks that allow parallel updates of multiple cars. However, this approach is not suitable for industrial smart sensor configuration due to the following two reasons: (i) Due to many sensors being resource constraint devices that operate on battery power, 802.11 based technologies are considered to be too energy consuming for smart sensor configuration purposes. (ii) The authors consider adding hardware security modules (HSM) in their approach but state that such a solution would lead to significant extra costs. This is, of course, infeasible for smart sensors.

Configuration scenarios involving resource constraint devices often use NFC due to the fact that NFC devices operated in a passive mode provide excellent energy efficiency. Wu et al. [11] present an approach to reprogramming computational RFIDs over the air using the electronic product code protocol. Serfass and Yoshigoe [12] propose an Android-based framework for NFC peer-to-peer communication that allows transferring arbitrary data. Similar to that, Haase et al. [13] present an NFC-based configuration framework for sensors and actuators used in home automation contexts. However, due to the home automation focus, the security level provided by that approach is considered as insufficient for industrial scenarios. Ulz et al. [14] present a key update process for industrial devices based on NFC. However, this approach does not allow arbitrary configuration data to be transferred.

None of the presented approaches includes a verification process to ensure the correct application of new configuration data. An overview summarizing the related work regarding device configuration is shown in Table I.

B. Configuration Attestation

The remote attestation of device characteristics such as hardware properties, operating system, or services is a well-covered topic in the IoT [15], for wireless sensor networks [16], and generally for resource constraint devices [17]. Saroiu and Wolman [18] discuss various scenarios that are affected by untrustworthy sensor data. The authors also suggest to include trusted computing hardware such as a trusted platform module (TPM), Intel's trusted execution technology (TXT), or ARM's TrustZone (TXT and TrustZone either use or closely relate to TPM functionality [19]) into sensors to provide trusted data.

In fact, most proposed attestation approaches rely on trusted computing hardware, due to the constraints and assumptions that are often necessary for software-based attestation [20]. However, many approaches attest static parts of a system, such as the BIOS, boot loader, or binaries that should get executed, while we need to attest a configuration that is changing.

Regarding the attestation of changing properties, Kil et al. [21] propose a method for dynamic system properties attestation. In their approach, a *challenger* requests an attestation that is then performed by the *attester*. The dynamic properties that are attested are structures in an application's memory that need to be defined before deployment of the application. The method also needs a BIOS that supports *core root of trust measurements* which makes it infeasible for smart sensors. SCUBA, a secure code update by attestation for sensor networks [22] relies on indisputable code execution which is a software security measure shown to be susceptible to certain attacks [20]. A promising approach is so-called *property-based* attestation [23] that however requires functionality not yet included into the TPM specification. As an alternative, the authors assume a trusted execution environment that is needed in their approach, which we will include in our proposed architecture.

C. Near Field Communication (NFC)

NFC is a contactless communication standard based on several RFID standards. The technology is well-known for its usage in contactless payment solutions, ticketing, and access control systems [24]. NFC operates at 13.56 MHz, typically in ranges of 3-10 cm while supporting bit rates that are multiples of 106 kbps (up to 848 kbps). Due to the frequency spectrum used, NFC is not susceptible to interference from other wireless technologies such as WiFi, Bluetooth, or 801.15.4 based protocols. Due to the limited communication range, NFC provides certain security advantages compared to other wireless technologies [25]. Although the communication range of NFC is limited, attacks that allow eavesdropping in distances of 10 m haven been shown. Therefore, security measures to protect the confidentiality, integrity, and authenticity of transferred data need to be implemented.

D. Authenticated Encryption (AE)

AE comprises *private key cryptography* with *Message Authentication Codes (MAC)* in a secured way such that the confidentiality, integrity, and authenticity of data can be provided [26]. The well-known private key cryptography algorithm *Advanced Encryption Standard (AES)* provides specialized modes of operation such as AES-CCM that are capable of providing AE. AES can be implemented efficiently in hardware with respect to performance as well as size requirements (e.g. Rogawski and Gaj [27]). Therefore, the usage of dedicated hardware to perform security relevant operations in smart sensors is highly practicable [28].

E. Secure Element (SE)

The combination of processing units for secured code execution and secured storage for data and applications is denoted as SE. In contrast to general purpose CPUs, the secured code execution environment mitigates exploits based on flaws such as buffer overflows. In addition to that, the SE also implements appropriate countermeasures to mitigate physical attacks. SEs that are capable of mitigating physical attacks provide so-called *tamper resistance* [29]. The security level provided by SEs is assessed by a common criteria (CC) information technology security evaluation [30] in order to be able to compare the security provided by SEs.

III. Problem Definition and System Model

Before presenting our approach for smart sensor configuration and configuration attestation, we define the problem we face, and define our system model.

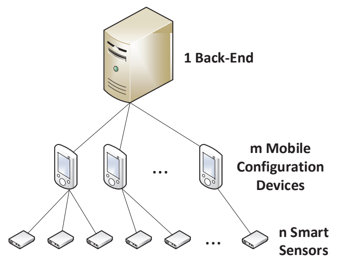


Fig. 2. System model for smart sensor configuration.

A. Problem Definition

When configuring smart sensors, (confidential) configuration data is transferred to a device that can not be fully trusted, even when trusted hardware components such as a trusted platform module (TPM) are included in the smart sensor. The configuration data also needs to be transferred to the device using a communication channel with potential adversaries. Therefore, we need to consider the following three problems:

1) To configure a smart sensor, confidential configuration data needs to be sent using a communication channel that might be accessible to potential adversaries.
2) A malicious sensor device might read configuration data and reveals confidential information to a third party.
3) A malicious sensor device might accept a configuration but does not apply it. Therefore, the correct functionality of the device is compromised.

To summarize these problems, we assume an adversary that is able to access and perform malicious operations on both the communication channel, and the smart sensor.

B. System Model

For our configuration approach we assume the system model shown in Fig.2 that comprises the following three entities:

Smart Sensor: The smart sensor that needs to be configured. There is no limitation on the number of devices; we generally assume *n* smart sensors in our system model.

Mobile Configuration Device: The mobile device used to update configuration data on smart sensors. In our approach there is no limitation regarding the number of configuration devices used, so we assume a number of *m* mobile configuration devices in our system model.

Configuration Back-End: The back-end manages and initializes all configuration changes. This means, changes need to be done done on the back-end from where they are transferred to the smart sensor using the mobile configuration device. In our system model, we assume *one* configuration back-end.

C. Assumptions

Based on our system model, we assume a back-end that operates as a global configuration storage to be trustworthy

231

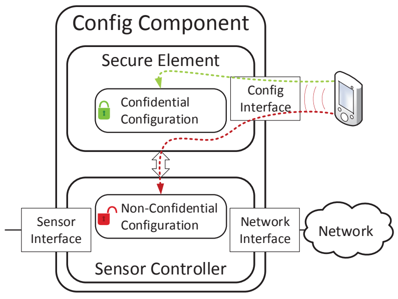


Fig. 3. NFC-based configuration architecture for a smart sensor's confidential as well as non-confidential configuration data.

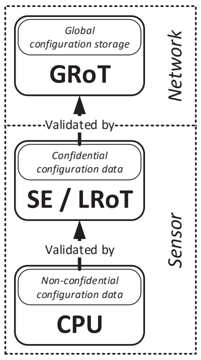

Global configuration storage
GRoT
Network
Validated by
Confidential configuration data
SE / LRoT
Validated by
Sensor
Non-confidential configuration data
CPU

Fig. 4. Necessary two-layer architecture for configuration attestation due to allowing data of two confidentiality levels.

and sufficiently secured against attacks. We further assume that all configuration changes must be initialized and therefore authorized by this back-end. Thus, the back-end has knowledge of all smart sensor configuration versions that are configured and managed by the given back-end.

IV. NFC-based Configuration

Smart sensors need to be configured during their entire lifecycle as shown in Fig. 1. To account for this requirement, we propose to use an NFC-based configuration interface for the following reasons:

- NFC allows ad-hoc connections to be established instead of exposing the configuration interface to potential adversaries located in a network.
- The limited communication range of NFC also offers advantages in limiting the malicious use of this interface due to adversaries having to be in close proximity to the smart sensor in order to use to the configuration interface.
- If the NFC module of the smart sensor is operated in passive mode, no energy is needed for communication. The hardware components involved in the configuration process can even be powered through the NFC field of the communication partner's NFC device, which is needed in certain phases of a smart sensor's lifecycle (e.g. during manufacturing of the smart sensor).

A potential drawback of NFC is that there are no security measures included in the NFC standard to protect the confidentiality, integrity, and authenticity of transferred data. Therefore, we propose to use AE in combination with ticketing information that is used to verify if a configuration should be accepted by a smart sensor or not. To perform all involved cryptographic methods in a secured execution environment, our approach relies on an SE that is combined with a general purpose processor as shown in Fig. 3. This *Config Component* implements *security by isolation* approach (e.g. Vasudevan et al. [31]; *normal* and *secured* world) allows execution and data storage to be split into confidential or critical, and non-confidential or non-critical parts. The responsibility of the SE and sensor controller in our presented approach are:

SE: The SE offers a secured execution environment for critical code such as AE. In addition, the SE also offers secured storage for confidential configuration data that can be stored in a tamper resistant manner. To enable configuration transfer via NFC, the SE also includes an NFC interface. That interface allows the SE to be powered by an NFC field, even if there is no power source attached to the smart sensor. Due to the SE providing the NFC interface, confidential data is directly transferred to the SE and no additional interface for configuration updates or storage needs to be exposed which potentially also mitigates so-called API-level attacks that target these interfaces [32].

Sensor Controller: The sensor controller includes interfaces to the sensor hardware, to the network, and to the SE. Due to the fact that this controller is a general purpose controller, it also provides an execution environment for non-critical code as well as storage for non-confidential configuration data.

Having both, secured and unsecured data storage, our approach is able to handle confidential as well as non-confidential configuration data. Confidential configuration data could include information such as keys used for communication, firmware updates for the SE, or data for local decision making in a smart sensor. Non-confidential data could, for instance, represent settings such as the sampling rate of a smart sensor but also firmware updates for the sensor controller. Due to having two layers of configuration data with different confidentiality requirements, the verification process of the applied configuration update also needs to be done in a two-layer architecture as shown in Fig. 4. There, *LRoT* denotes the *local root of trust* that is used to attest non-confidential configuration data. *GRoT* denotes the *global root of trust* that is then used for confidential configuration attestation.

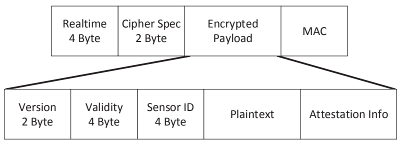


Fig. 5. NDEF packet structure of transferred configuration data.

Independent of the confidentiality level when storing configuration data, security measures need to be implemented when transferring the configuration data using NFC to mitigate eavesdropping, and replay attacks. Therefore, every configuration that is transferred needs to be secured using the security mechanisms shown in the NFC data exchange format (NDEF) packet structure (see Fig.5) used in our approach. The fields included in this NDEF packet are:

- *Realtime:* The realtime of the device that is sending the configuration update. This information is used to decide if a packet will be accepted or rejected by a smart sensor (see field *Validity*). This information is not encrypted as it is added to the NDEF package by the device deploying the configuration due to most smart sensors not having time synchronization.
- *Cipher Specifications:* This field defines the used encryption algorithm and corresponding key lengths to allow the smart sensor to use the appropriate algorithms when decrypting and verifying the package.
- *Encrypted payload and MAC:* These two fields are generated by encrypting the plaintext payload and by calculating the MAC of that same payload.
- *Version:* To mitigate replay attacks, a version number is included in the encrypted payload. For a smart sensor to accept a configuration update, this version number must be larger than the current configuration version.
- *Validity:* The validity defines how long a given configuration package is valid to also mitigate replay attacks. The validity is checked against the included realtime from the configuration deploying device.
- *Sensor ID:* The included sensor ID must match the sensor ID stored in the tamper resistant memory of the SE for the smart sensor to accept the configuration update.
- *Plaintext:* This is the transferred configuration data.
- *Attestation Info:* To allow the LRoT to perform attestation operations, attestation information is also added to the transferred data. The used attestation approaches will be discussed in detail in the next section.

The secured NDEF data structure is used to transfer data from the back-end to the mobile device as well as when transferring data from the mobile configuration device to the smart sensor. Using the same package entails that the transferred data can not be modified in any way on the mobile device. Since we consider the mobile configuration device itself as well as its operator as untrustworthy, only allowing data to be transferred secured by AE mitigates attacks enabled by malicious devices or users.

V. Two-Layer Configuration Attestation

Before presenting our two-layer configuration attestation architecture, we are briefly going to discuss some terminology related to attestation.

A. Attestation Terminology

Usually in attestation there are two roles, a *challenger* and an *attestor* [33]. The challenger is the entity interested in the correctness or trustworthiness of a system. That is, the output of the attestation process. The attestor (often also prover) is the entity that needs to prove its correctness and trustworthiness by measuring and attesting its configuration. An attestation process usually is assisted by some dedicated hardware that supports *trusted computing*. The TPM specification of the trusted computing group (TCG) lists two mechanisms that are of interest when discussing device attestation: *remote attestation* and *sealed storage*. Remote attestation defines how to use a TPM's secured storage, the platform configuration registers (PCR), to implement an attestation process. Sealed storage refers to data (information or code) that is stored encrypted using a key calculated as a function of a TPM's PCR values. That is, the data is only unsealed if the attestor is able to prove its correct state.

B. SE versus TPM

Most attestation solutions require secured hardware to be used at the attestor. This secured hardware component is a TPM in most cases. Sadeghi et al. [34] argue that such secured hardware is too complex and often too expensive for most resource constraint devices such as smart sensors. The authors also state that although software-based attestation solutions have been proposed, at least a basic subset of security features in hardware will be needed. Therefore, we propose to use an SE such as a product from Infineon's SLE78 product family (see [35]) in our approach for the following three reasons:

1) When using an SE such as Infineon's SLE78 that was designed for smart cards, secured hardware can also be included into smart sensors that are constrained in terms of size and available energy.
2) Although TPMs with NFC capability have been proposed [36], no currently available TPM offers an NFC interface. In contrast to that, certain SEs such as from the SLE 78 family offer an NFC interface and the required security properties needed for attestation.
3) The applied attestation approach that will be presented in this paper requires a trusted execution environment which is not included in the TPM specifications. However, security controllers are capable of providing such a tamper resistant execution environment.

C. Two-layer Approach

Our two-layer configuration attestation approach is based on the fact that the configuration solution presented in Section IV supports two different levels of confidentiality for configuration data. As shown in Fig. 4, non-confidential configuration will be attested using the SE included in our proposed configuration component while confidential configuration data will be attested by the trusted back-end.

D. Non-Confidential Configuration Attestation

Non-confidential configuration data that can be changed on the sensor controller can be either application updates (binaries), configuration data (e.g. sampling rate), or both. Since malicious code, as well as malicious configurations, can be harmful, attestation is necessary for both components. Due to the different nature of information, we suggest using two different attestation techniques respectively.

Application (Binaries) are static memory content that is changed less frequent than configuration data. According to Yang et al. [37] the size needed for application binaries can usually be assumed to be two orders of magnitudes larger than the space required for (configuration) data. Therefore, we propose to use basic binary attestation where a hash value is computed over the complete application binaries. Advanced methods such as *pseudo-random memory traversal* [38] can also be implemented to prevent attacks such as memory copy attacks, or pre-computation and replay attacks. The necessary information to attest the correctness of updated binaries are included in the configuration update NDEF package (attestation info, see Fig. 5) and therefore are also updated in the SE (LRoT) whenever new application binaries or non-confidential configuration data are transferred to the SE.

Non-Confidential Configuration Data is also stored in the sensor controller's memory. However, since configuration data is much smaller and attacks such as memory copy attacks are easy to implement, we propose to use *property-based attestation* [23] for configuration data. As stated by the authors, a property-based attestation mechanism requires additional functionality that is currently not included in the TCG's TPM specifications. Therefore, a trusted execution environment is needed to implement the desired functionality. In our approach, the trusted execution environment is provided by the SE. To implement property-based attestation, certificates are needed for each valid configuration property. The problem of certificate revocation can easily be solved in our approach by including certificate information in the configuration update's attestation information field.

The two different attestation techniques are then jointly used to *grant or deny network access* to the sensor controller. We suggest achieving this by using *sealed storage* to protect code such as the whole network stack, or other information from being accessed by an unattested sensor controller. By restricting network access through local attestation instead of using remote attestation, malicious smart sensors can be isolated from the network. Thus, such sensors are hindered from infecting other network devices or start network-based attacks such as denial-of-service attacks, jamming, or deception attacks [39], [40]. The decision on which information to seal in order to protect network access needs to be based on a trade-off between parameters such as security level, and overhead. On the one hand, sealing the network stack would require a sensor controller to prove its correctness only once before copying the network stack to its own memory. However, an adversary then could modify binaries or configuration after having unsealed the network stack. On the other hand, sealing an encryption key and requiring an attestation every time before allowing an encrypted packet to be sent, increases the overhead while still allowing certain kinds of attacks such as DoS attacks. Since we only present an attestation architecture in this paper, we refer to future work for a detailed comparison of different approaches. Independent of the chosen sealing approach, attestation information is stored in an SE that provides tamper resistance, attestation information is efficiently protected from being tampered with. Therefore, adversaries are not able to manipulate stored attestation information.

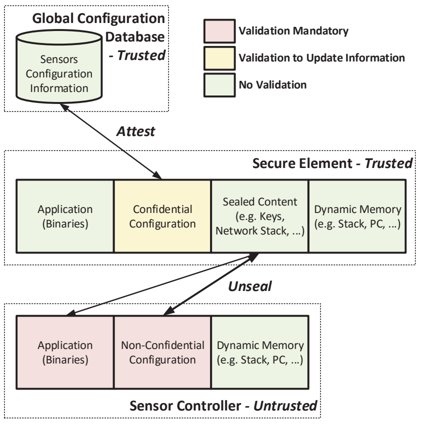

Global Configuration Database - Trusted
Sensors Configuration Information
Validation Mandatory
Validation to Update Information
No Validation
Attest
Secure Element - Trusted
Application (Binaries)
Confidential Configuration
Sealed Content (e.g. Keys, Network Stack, ...)
Dynamic Memory (e.g. Stack, PC, ...)
Unseal
Application (Binaries)
Non-Confidential Configuration
Dynamic Memory (e.g. Stack, PC, ...)
Sensor Controller - Untrusted

Fig. 6. Two-layer configuration attestation in detail.

E. Confidential Configuration Attestation

Confidential configuration data is secured by the applied security measures when transferring the data via NFC and storing that data on an SE that provides tamper resistance. Therefore, the correctness of these configuration parameters is assumed in our approach. The attestation of confidential configuration data to the global configuration database in the back-end (GRoT) is still required to verify the successful application of configuration data. That is, any malicious user that does not apply a configuration update must be detected by the GRoT. Since the second layer needs to attest configuration parameters, *property-based* attestation is used to attest the correctness of confidential configuration data.

VI. Evaluation

To show the technical feasibility of our proposed configuration component, the hardware demonstrator shown in Fig. 7 was realized. This demonstrator comprises two different controllers. An Infineon XMC4500 microcontroller that is

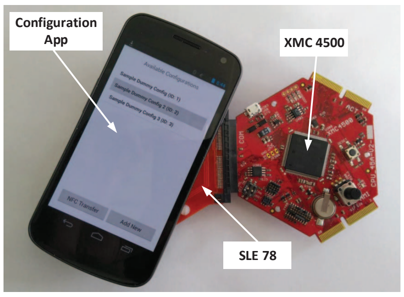


Fig. 7. Configuration component hardware demonstrator and Android device running the NFC configuration application.

used as the sensor controller in our approach and an Infineon SLE 78 [35] that is used as SE. As a mobile configuration device, we used an NFC-enabled Android smartphone. Using this demonstrator, configuration update times (reboot of the system not included) of about 200 ms can be achieved for configurations consisting of 5-10 configuration parameters including the necessary overhead imposed by the implemented security mechanisms.

A. STRIDE Threat Analysis

To highlight the achieved security level, a threat analysis was conducted that demonstrates the lists the threats (*T*) that can be mitigated by countermeasures (*C*) implemented our approach. Further, the threats are categorized by the STRIDE model (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege). Although we do not claim this threat analysis to be exhaustive, we think the listed threats represent the most relevant threats for a smart sensor configuration scenario. Regarding the threat analysis, we make the following assumptions: (i) We assume the global configuration database back-end is sufficiently secured against any kind of attack; therefore, it is considered as trusted entity. (ii) Also the SE used in our approach is considered as trusted entity. (iii) Other than that, we assume all other involved entities as being untrustworthy. **T1** (S, R, I, D, E): Adversary that is trying to perform remote attacks on the configuration interface. **C1**: Mitigated by using NFC which requires an adversary to be in close proximity to the smart sensor. **T2** (R, I): Adversary in close proximity is trying to eavesdrop or manipulate configuration packages. **C2**: Mitigated by using AE. **T3** (S, D): Adversary in close proximity is trying to use a captured configuration package to perform replay attacks. **C3**: Appropriate countermeasures are included in configuration package to mitigate these type of attack. **T4** (S, R, I, D, E): An adversary is able to inject malicious code or manipulated configuration data into the sensor controller. **C4**: When attested, the sensor controller is not able to unseal its network stack stored in the SE. Thus, the malicious smart sensor is blocked from accessing the network. **T5** (S, R, D): Malicious code in the sensor controller performs attacks (e.g. DoS) targeting other network devices or tries to replicate the malicious code to other devices. **C5**: When attested, the sensor controller is not able to unseal its network stack stored in the SE. Thus, the malicious smart sensor is blocked from accessing the network. **T6** (D): A malicious user tries to manipulate the functionality of a smart sensor by not applying a necessary configuration update. **C6**: The global configuration database back-end attests the configuration state of a smart sensor; therefore, not updated devices can easily be detected. **T7** (D): Adversary in close proximity tries to perform a DoS attack by continuously sending malicious configuration packages to the smart sensor. **C7**: The updates are rejected by the SE. Normal operation of the smart sensor is not impacted since the SE is powered through the mobile device's NFC field; therefore, no power required by the smart sensor is consumed. Also, all cryptographic operations to decide if a package needs to be rejected are performed at the SE, which does not impact the normal operation of the sensor controller. **T8** (S, T, R, I, D, E): Adversary with physical access to the smart sensor tries to perform physical and side-channel attacks to reveal confidential data such as key material or cryptographic algorithms. **C8**: The used SE mitigates physical attacks by implementing appropriate countermeasures. The security level is certified by CC.

VII. Conclusion and Future Work

In this paper we present a hardware-secured configuration approach based on NFC that is suitable for both confidential and non-confidential data alike. Our approach comprises (i) a component that can be included into future smart sensors as well as into legacy devices and (ii) a NDEF-based configuration protocol. The protocol includes information to prevent updates from malicious users and mitigates replay attacks. By allowing only NFC for configuration changes, the configuration interface is not exposed to remote attacks from the network. In addition, we also propose a two-layer configuration attestation architecture to attest the correctness of applied configuration updates. This architecture is capable of attesting non-confidential configuration locally using an SE as well as confidential configuration data remotely using a trusted global configuration database. The technical feasibility of our architecture is shown by means of a hardware demonstrator. In addition to that, the security properties are evaluated in a STRIDE threat analysis that highlights the increased security.

As future work we plan to investigate different methods to grant or deny network access for smart sensors regarding their trade-off between provided security level, and overhead.

Acknowledgment

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia.

References

[1] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, “Smart Objects as Building Blocks for the Internet of Things,” *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, 2010.

[2] G. Meijer, K. Makinwa, and M. Pertijs, *Smart Sensor Systems: Emerging Technologies and Applications*. John Wiley & Sons, 2014.

[3] T. Xu, J. B. Wendt, and M. Potkonjak, “Security of IoT Systems: Design Challenges and Opportunities,” in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2014, pp. 417–423.

[4] H. C. Pöhls, V. Angelakis, S. Suppan, K. Fischer, G. Oikonomou, E. Z. Tragos, R. D. Rodriguez, and T. Mouroutis, “RERUM: Building a Reliable IoT upon Privacy- and Security- enabled Smart Objects,” in *Wireless Communications and Networking Conference Workshops (WCNCW), 2014 IEEE*. IEEE, 2014, pp. 122–127.

[5] S. Weyer, M. Schmitt, M. Ohmer, and D. Gorecky, “Towards Industry 4.0 - Standardization as the crucial challenge for highly modular, multi-vendor production systems,” *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 579–584, 2015.

[6] L. Schor, P. Sommer, and R. Wattenhofer, “Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings,” in *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. ACM, 2009, pp. 31–36.

[7] J. Lee, B. Bagheri, and H.-A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[8] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, “Provisioning Software-Defined IoT Cloud Systems,” in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. IEEE, 2014, pp. 288–295.

[9] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen, “Sensor Discovery and Configuration Framework for The Internet of Things Paradigm,” in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 94–99.

[10] M. Steger, C. Boano, M. Karner, J. Hillebrand, W. Rom, and K. Römer, “SecUp: Secure and Efficient Wireless Software Updates for Vehicles,” in *Digital System Design (DSD), 2016 Euromicro Conference on*. IEEE, 2016, pp. 628–636.

[11] D. Wu, M. J. Hussain, S. Li, and L. Lu, “R2: Over-the-Air Reprogramming on Computational RFIDs,” in *RFID (RFID), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[12] D. Serfass and K. Yoshigoe, “Wireless Sensor Networks Using Android Virtual Devices and Near Field Communication Peer-To-Peer Emulation,” in *Southeastcon, 2012 Proceedings of IEEE*. IEEE, 2012, pp. 1–6.

[13] J. Haase, D. Meyer, M. Eckert, and B. Klauer, “Wireless sensor/actuator device configuration by NFC,” in *2016 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2016, pp. 1336–1340.

[14] T. Ulz, T. Pieber, C. Steger, S. Haas, H. Bock, and R. Matischek, “Bring Your Own Key for the Industrial Internet of Things,” in *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2017, pp. 1430–1435.

[15] T. Rauter, A. Höller, J. Iber, and C. Kreiner, “Thingtegrity: A Scalable Trusted Computing Architecture for the Internet of Things,” in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*. Junction Publishing, 2016, pp. 23–34.

[16] H. Tan, W. Hu, and S. Jha, “A TPM-enabled Remote Attestation Protocol (TRAP) in Wireless Sensor Networks,” in *Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. ACM, 2011, pp. 9–16.

[17] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, “TrustLite: A Security Architecture for Tiny Embedded Devices,” in *Proceedings of the Ninth European Conference on Computer Systems*. ACM, 2014, p. 10.

[18] S. Saroiu and A. Wolman, “I am a Sensor, and I Approve This Message,” in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*. ACM, 2010, pp. 37–42.

[19] W. Arthur and D. Challener, *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*. Apress, 2015.

[20] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, “On the Difficulty of Software-Based Attestation of Embedded Devices,” in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 400–409.

[21] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang, “Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence,” in *Dependable Systems & Networks, 2009. DSN’09. IEEE/IFIP International Conference on*. IEEE, 2009, pp. 115–124.

[22] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla, “SCUBA: Secure Code Update By Attestation in Sensor Networks,” in *Proceedings of the 5th ACM workshop on Wireless security*. ACM, 2006, pp. 85–94.

[23] A.-R. Sadeghi and C. Stüble, “Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms,” in *Proceedings of the 2004 workshop on New security paradigms*. ACM, 2004, pp. 67–77.

[24] B. Benyo, A. Vilmos, K. Kovacs, and L. Kutor, “NFC Applications and Business Model of the Ecosystem,” in *Mobile and Wireless Communications Summit, 2007. 16th IST*. IEEE, 2007, pp. 1–5.

[25] E. Haselsteiner and K. Breitfuß, “Security in Near Field Communication (NFC),” in *Workshop on RFID security*, 2006, pp. 12–14.

[26] M. Bellare and C. Namprempre, “Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.

[27] M. Rogawski and K. Gaj, “A High-Speed Unified Hardware Architecture for AES and the SHA-3 Candidate Grøstl,” in *Digital System Design (DSD), 2012 15th Euromicro Conference on*. IEEE, 2012, pp. 568–575.

[28] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Moderator-Ravi, “Security as a New Dimension in Embedded System Design,” in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 753–760.

[29] S. Ravi, A. Raghunathan, and S. Chakradhar, “Tamper Resistance Mechanisms for Secure Embedded Systems,” in *VLSI Design, 2004. Proceedings. 17th International Conference on*. IEEE, 2004, pp. 605–611.

[30] D. Mellado, E. Fernández-Medina, and M. Piattini, “A common criteria based security requirements engineering process for the development of secure information systems,” *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[31] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J. M. McCune, “Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give Me?” in *International Conference on Trust and Trustworthy Computing*. Springer, 2012, pp. 159–178.

[32] M. Bond and R. Anderson, “API-Level Attacks on Embedded Systems,” *Computer*, vol. 34, no. 10, pp. 67–75, 2001.

[33] R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, “Design and Implementation of a TCG-based Integrity Measurement Architecture.” in *USENIX Security Symposium*, vol. 13, 2004, pp. 223–238.

[34] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and Privacy Challenges in Industrial Internet of Things,” in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[35] Infineon Technologies AG, “Infineon Chip Card & Security ICs Portfolio,” 2015, accessed on 04/04/2017. [Online]. Available: http://www.infineon.com/cms/en/product/security-and-smart-card-solutions/security-controllers/sle78/channel.html?channel=5546d462503812bb015066c2d8e91745

[36] M. Hutter and R. Toegl, “A Trusted Platform Module for Near Field Communication,” in *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*. IEEE, 2010, pp. 136–141.

[37] Y. Yang, X. Wang, S. Zhu, and G. Cao, “Distributed Software-based Attestation for Node Compromise Detection in Sensor Networks,” in *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*. IEEE, 2007, pp. 219–230.

[38] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla, “SWATT: SoftWare-based ATTestation for Embedded Devices,” in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*. IEEE, 2004, pp. 272–282.

[39] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli, “False Data Injection Attacks against State Estimation in Wireless Sensor Networks,” in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5967–5972.

[40] C. Kwon, W. Liu, and I. Hwang, “Security Analysis for Cyber-Physical Systems against Stealthy Deception Attacks,” in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 3344–3349.

Towards Trustworthy Data in Networked Control Systems: A Hardware-Based Approach

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Rainer Matischek, Holger Bock
Development Center Graz
Infineon Technologies Austria AG
Graz, Austria
{rainer.matischek, holger.bock}@infineon.com

***Abstract*—The importance of Networked Control Systems (NCS) is steadily increasing due to recent trends such as smart factories. Correct functionality of such NCS needs to be protected as malfunctioning systems could have severe consequences for the controlled process or even threaten human lives. However, with the increase in NCS, also attacks targeting these systems are becoming more frequent. To mitigate attacks that utilize captured sensor data in an NCS, transferred data needs to be protected. While using well-known methods such as Transport Layer Security (TLS) might be suitable to protect the data, resource constraint devices such as sensors often are not powerful enough to perform the necessary cryptographic operations. Also, as we will show in this paper, applying simple encryption in an NCS may enable easy Denial-of-Service (DoS) attacks by attacking single bits of the encrypted data. Therefore, in this paper, we present a hardware-based approach that enables sensors to perform the necessary encryption while being robust against (injected) bit failures.**

***Index Terms*—Networked Control System; Security; Encryption; Forward Error Correction.**

I. Introduction

Networked Control System (NCS) nowadays are gaining popularity due to, among other things, Internet of Things (IoT) technologies where systems such as intelligent traffic control systems comprising of a large number of sensors and actuators are envisioned [1]. Systems that monitor and control a physical process through some computational device are often generally defined as cyber-physical systems (CPS) [2]. These systems also allow the involved devices to be connected to private and even public networks. Inspired by the IoT and CPS, several working groups proposed high-tech strategies such as Industry 4.0 [3] or smart manufacturing [4]. These strategies envision so-called smart factories that connect every device involved in the production process with each other or even with the Internet. All of these trends have one thing in common: devices are interconnected which allows NCS to be implemented efficiently using the corresponding network structures.

A general definition for an NCS is given by Gupta and Chow [5]. The authors state that a traditional feedback control system that is closed via a shared communication channel should be classified as an NCS. They also highlight this as a key characteristic common in many NCS definitions: information in the NCS is exchanged between involved components (sensor, controller, and actuator) using this shared communication channel. However, using a shared communication channel results in several challenges for NCS:

1) **Delays:** Using a shared communication channel may induce unreliable and non-deterministic behaviour into an NCS [6]. If the resulting delays are too large for an NCS with time constraints, the performance of the NCS can be impacted [7]. This could ultimately lead to potential physical damage to the controlled process or even threaten human lives, for example, in traffic NCS.
2) **Packet Loss:** Another property common in shared communication channels is the probability of packet loss. If relevant information such as measured plant output or control input are lost, the stability of the NCS may be compromised [8]. Stabilization problems could lead to compromised NCS performance, severe physical damage of the controlled process, or even threaten lives.
3) **Information Security:** When transferring information such as measured output or control input using a shared communication channel, attacks that could compromise the NCS functionality can easily be conducted [9]. In addition to that, an adversary that has learned the behaviour of an NCS through eavesdropping communication, may be able to manipulate a system in a way such that the attack remains undetected [10]. Therefore, the trustworthiness of transferred information often needs to be improved.

While a lot of current research is dedicated to the impact of network delays and packet loss in NCS, not much research has been done regarding information security as pointed out by Byres and Lowe [11]. One of the limiting factors in NCS related security research is the fact that security measures require additional computational resources and time. For example, using TLS for sensor to controller communication often will be infeasible due to resource constraint sensor hardware. However, the trustworthiness of sensor data is essential in NCS as compromised data can lead to malfunctioning systems. To improve the trustworthiness of data while imposing as little delay as possible, algorithms and/or hardware extensions will be necessary. The approach presented in this paper therefore makes the following contributions: (i) We propose the combination of encryption and error correction to mitigate NCS related attacks. (ii) To impose a minimum of delay, a hardware extension is presented that can be integrated into sensors and actuators. (iii) The presented approach can be applied to the general concept of NCS; no network technology and related security feature such as error correction is assumed.

The remainder of this paper is structured as follows. In Section II technical background information regarding technologies used in our approach as well as related work regarding NCS security solutions is given. Section IV and V show a naïve and an enhanced approach respectively that can be used to mitigate certain kinds of attacks. This paper is then concluded in Section VI where also potential future work is discussed.

II. Background and Related Work

A. Authenticated Encryption (AE)

AE generally combines a symmetric encryption scheme with a message authentication code (MAC) to provide confidentiality, integrity and authenticity of data [12]. A symmetric (private-key) encryption scheme requires the two communicating parties to be in possession of the same shared secret key [13]. A widely used symmetric encryption scheme, the advanced encryption standard (AES) [14], operates as a block cipher, processing plaintext blocks of 16 bytes. For most symmetric block ciphers specialized AE modes exist, such as AES-CCM that can efficiently be implemented in hardware [15] to provide reliable and fast execution of the algorithm. Such implementations are feasible to provide encryption fast enough for 100 Gbit/s ethernets [16].

B. Forward Error Correction (FEC)

FEC is used to detect and correct errors in data transmission resulting from unreliable and noisy communication channels. First applied by Hamming [17], the basic idea is to add redundant information produced by an error correcting code (ECC) before sending data. This redundant information allows to detect or probably even to correct errors without requiring the data to be transmitted again. ECCs, however, are not limited to data transmission as one major field of application is memory [18]. FEC can be implemented efficiently enough to be suitable for applications relying on high speed, such as 100 Gbit/s transport networks [19]. A high performance type of FEC are so-called turbo codes [20] that are used in 3G and 4G networks as well as in space programs [21].

C. Joint Encryption and Error Correction (JEEC)

JEEC was already discussed in research in the 1980s where authors claimed that combining encryption and error correction could lead to efficient implementations that could be done in a cost effective way [22], [23]. These solutions used the data encryption standard (DES) that nowadays is ousted by AES. Mathur et al. [24] present an approach based on AES that provides the same security level as AES. Gligoroski et al. [25] discuss encryption and error correction coding done in a single step for more recent algorithms. As the authors mention, also sequential execution of encryption and error correction codes is a possibility with execution performance being a drawback of that approach.

D. Security Controller (SC)

SCs are processing units that provide a secured execution environment for applications as well as secured storage for data and applications. Compared to a general purpose CPU, attacks based on issues such as buffer overflows are much harder to exploit on an SC. In addition, SCs also provide tamper resistance [26] that mitigates physical attacks by using appropriate countermeasures. To assess the provided security level of an SC the common criteria (CC) information technology security evaluation is used [27]. Because embedded systems are often operated in untrusted environments and thus accessible to adversaries, tamper resistance is of critical importance [28].

E. NCS Security

In order to understand security threats in NCS, a threat model containing potential vulnerabilities and the impact of attacks needs to be defined first, as shown by Cárdenas et al. [29]. The authors also highlight the differences of NCS compared to traditional IT components: (i) frequent security updates may not be possible for NCS and (ii) the interaction of NCS with the physical world that greatly increases the impact of attacks. For an NCS to be considered secured, the following four security properties need to be fulfilled:

- **Confidentiality**: Data is not made available to unauthorized entities.
- **Integrity**: Data is not modified in an undetected manner during its entire life cycle.
- **Availability**: System is available in order to fulfill its intended task.
- **Authenticity**: Data is from the expected sender and not injected by some other entity.

These properties can be compromised by different types of NCS related attacks. Cárdenas et al. [9] highlight five attacking points for CPS (see Fig. 1). Due to the similarities between CPS and NCS, all of these five attacking points also apply for NCS. Attacks of category **A1** directly target the physical process. **A2** attacks are so-called *deception* attacks that are characterized by adversaries inducing false information $\tilde{y} \neq y$. The attacks can be backed by a previous *learning phase* in which the expected behaviour of the plant is learned first [10]. **A3** represents Denial-of-Service (DoS) attacks on the sensor to controller communication channel. Attacks that are characterized by adversaries trying to induce false control commands $\tilde{u} \neq u$ are represented by **A4**. Here, the adversary could either target the controller or the communication channel. **A5** denotes DoS attacks on the controller to actuator channel.

DoS attacks (**A3, A5**) are well covered in research with many authors trying to account for these types of attacks in the controller [30]. Due to the networking nature of NCS,

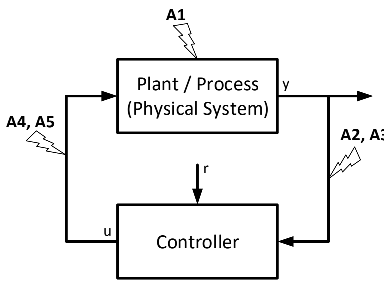


Fig. 1: Potential attacking points in a CPS (adapted from Cárdenas et al. [9]).

TABLE I: Comparison with related work. The properties (C)onfidentiality, (I)ntegrity, (A)vailability, and (Au)thenticity are evaluated.

Related Work	Remark	C	I	A	Au
[32], [30]	Packet loss due to DoS attacks.	✗	✗	✓	✗
[33], [10]	Deception attack detection.	✗	✗	✓	✗
[34]	Encryption and hash algorithms applied; algorithms considered insecure.	✓	✓	✗	✗
[35]	Encryption applied.	✓	✗	✗	✗
Our approach	Suitable combination of algorithms; tamper resistant hardware.	✓	✓	✓	✓

unintentional packet loss is a characteristic that robust control algorithms need to account for [31]. The approaches used to model such unintentional packet loss can then be adapted to account for malicious packet jamming or compromising. Amin et al. [32] use optimal control theory tools to optimize controller performance such that safety specifications are satisfied with high probability while power limitations are considered.

Replay attacks or deception attacks (**A2**) target sensor data in an NCS to learn the expected behaviour of a plant and then use that data to inject false measurements. These false measurements can be used to hide an ongoing attack or to compromise the functionality of an NCS. Mo and Sinopoli [33] discuss the impact of such attacks that can target a system in its steady state. They also propose a method to detect an ongoing replay attack that however decreases the performance of their used controller algorithm. Mo et al. [36] also demonstrate the usage of injected false sensor data to compromise the functionality of a state estimator, thus directly targeting the functionality of an NCS.

Urbina et al. [10] discuss the impact of stealthy deception attacks on control system. They suggest to use a physics-based attack detection model to detect ongoing attacks. The main idea of their approach is to compare current properties of the system with physics-based model of the system under normal behaviour. Pang and Liu [34] propose to use data encryption standard (DES) encryption and MD5 hashes to increase the confidentiality, integrity, and authenticity of transferred packets in an NCS. However, the approach by the authors has three problems: (i) Both used algorithms, DES and MD5 are considered to be insecure nowadays [37], [38]. (ii) Plain hash functions such as MD5 can not be used to efficiently protect message authenticity [39]. (iii) All security measures are implemented in software by the authors. This increases delays as well as allows keys to be extracted by physical attacks [40]. Gupta and Chow [35] analyze additional delay in NCS induced by security algorithms such as DES, 3DES, and AES. In the experiment conducted by the authors only DES encryption is considered as fast enough to not compromise the stability of the NCS. To compensate the overhead for other algorithms, the authors suggest to use 1-D gain schedulers.

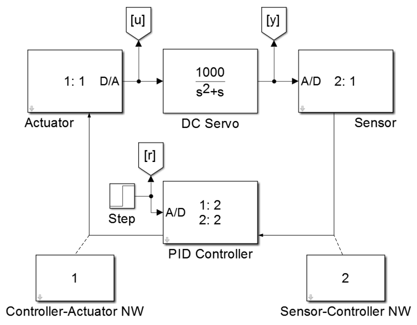


Fig. 2: MATLAB/Simulink model of an NCS for a DC servo. Both involved network connections are additionally outlined.

To increase the trustworthiness of data in an NCS, tradeoffs between measures such as imposed delay, provided security level, or energy efficiency need to be made [41] due to constraint devices. However, these tradeoffs might compromise security. In contrast to that, the approach presented in this paper tries to keep the associated impact of including security such as delay as small as possible. A comparison of our approach with presented work is given in Table I.

III. Evaluation Environment

To demonstrate the impact of different measures and parameters applied to the NCS, we use a MATLAB/Simulink simulation [42]. In addition to that, the TrueTime toolbox [43] is used to simulate network related behaviour, scheduling of software components, and real-time aspects. The process used for evaluation in this paper is described by the transfer function given in (1).

$$G(s) = \frac{1000}{s(s+1)} \qquad (1)$$

The system corresponding to that transfer function is a simple DC servo motor; the measurable system output being the angular position of that DC servo. To control this DC

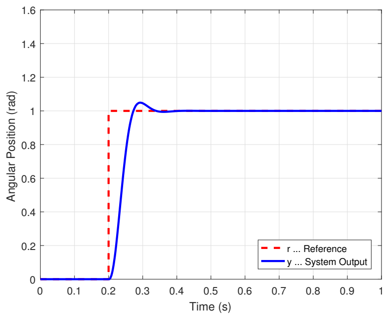


Fig. 3: Step response of the system shown in Fig. 2.

servo, a PD controller is used. Both the plant and the used PD controller can be found in the examples included in the TrueTime toolbox. The NCS shown in Fig. 2 comprises the DC servo plant, the respective actuators and sensors, the PD controller, and the simulated networks necessary for communication between components. As can be seen in Fig. 2, we use two different networks. Network 1 handles communication between the PD controller and the actuator, while the communication between the sensor and PD controller is handled by network 2. The PD controller, therefore is connected to both networks and has the same node ID 2 in all two networks. The actuator and sensor are assigned node ID 1 in their respective network. This setup allows us to simulate the impact of applied measures and network parameters on different parts of the NCS. In our case, we only manipulate the communication between sensor and PD controller. As imposed time delays are not a focus of this publication, the delays imposed by the network technology applied in an NCS are set low enough (10 ms for each transmission) for the simple PD controller to work correctly. The resulting closed-loop step response of our simulated NCS is shown in Fig. 3.

IV. Naïve Approach

The easiest approach to increase confidentiality, integrity, and authenticity of information transferred in an NCS is to use appropriate encryption algorithms. However, as we will show, this approach is also naïve in some kind as it introduces drawbacks regarding the NCS functionality that were to the best knowledge of the authors not discussed in other publications.

A. Usage of AE

In contrast to the discussed related work, we propose to use AE in the presented NCS context as this combination of encryption and MAC is suitable to provide confidentiality, integrity, and authenticity of information. Using encryption only or a combination of encryption and hash algorithms [34], [35] can not be used to provide all three mentioned security properties. If plain encryption is used, it is sufficient for an adversary to change a single bit of each transmitted packet to completely disturb the NCS functionality. As an example we use four sensor measurements shown in Table II as plaintext (PT) and encrypt them in one block using AES. One bit of the corresponding cyphertext (CT) is modified (last bit changed from 1 to 0) which results in a corrupted cyphertext (CT'). If this corrupted cyphertext is decrypted using the same key as for encrypting PT, a corrupted plaintext (PT') results. As can be seen, by just flipping one bit of the cypertext, the plaintext does not correlate to the original sensor measurements in any way and thus, can cause severe problems in an NCS if this corrupt data is not detected.

TABLE II: Sensor data plaintext (PT), cyphertext (CT), and corrupted cyphertext (CT') with resulting plaintext (PT').

	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 4**
PT	`0x00000001`	`0x00000002`	`0x00000003`	`0x00000004`
CT	`0xDE154CCE`	`0x18E65A6E`	`0xBD9A0593`	`0xE1B8250`$\boxed{\mathbf{7}}$
CT'	`0xDE154CCE`	`0x18E65A6E`	`0xBD9A0593`	`0xE1B8250`$\boxed{\mathbf{6}}$
PT'	`0x2D3DB30D`	`0xE89541F5`	`0x9AFD9AED`	`0x03BD8985`

TABLE III: AES modes for AE and the corresponding performance measures from Crypto++ [44].

Algorithm	**MiB/Second**	**Cycles per Byte**	**Table**
AES GCM 2K	102	17.2	2K
AES GCM 64K	108	16.1	64K
AES CCM	61	28.6	-

To detect problems resulting from manipulated cyphertexts and to provide data confidentiality, integrity, and authenticity we propose to use AE. AE can be implemented by combining encryption with a MAC. AES modes that can be used for AE are Counter with CBC-MAC Mode (CCM) as well as Galois/Counter Mode (GCM) [14]. We propose to choose the corresponding mode based on the memory/execution time tradeoff that needs to be made between those two algorithms as shown by Crypto++ Benachmarks [44] in Table III. If execution time is the most relevant factor, AES with GCM should be used for AE.

B. Bit Failures and Block ciphers

If AE based on a block cipher is applied, malicious data packages can be detected and discarded. However, this property is problematic for NCS as a single flipped bit causes a package containing sensor data to be dropped. For example, multi user Ethernet has a typical bit error rate (BER) of about 10^{-9} [45]. The packet error rate (PER) can be calculated according to (2) where N is the packet's size in bits.

$$PER = 1 - (1 - BER)^N \tag{2}$$

For transmitting 1 kB of data (N=8000) this equates to a PER of $\approx 8 \cdot 10^{-6}$. However, if an adversary is able to inject bit

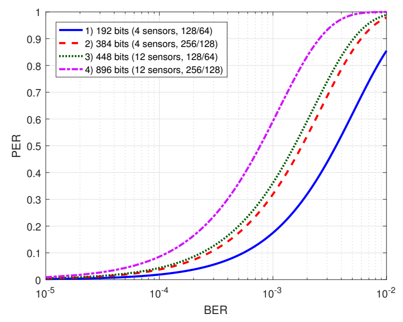


Fig. 4: PER depending on the BER, payload size, and parameters of the applied cryptographic algorithms.

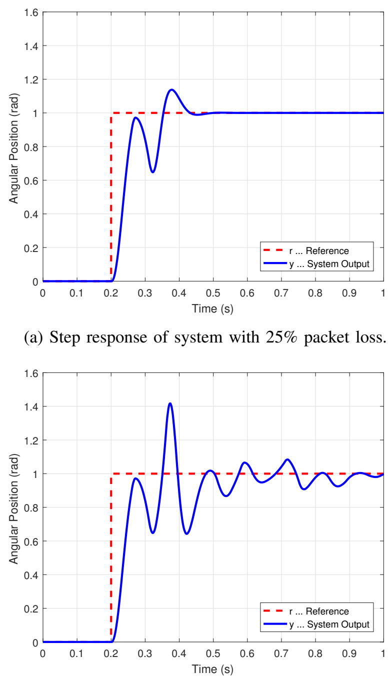
failures in any way, for example, by jamming wireless signals, the PER increases rapidly as can be seen in Fig. 4. In this figure, the PER depending on the BER, the payload size and the parameters of the applied cryptographic functions is shown. We demonstrate four different scenarios with combinations of AES block size, MAC length and number of sensors there. We assumed that each sensor measurement can be represented by a 32 bit number in this example.

1) 192 bits payload: AES block size 128 bit, 4 sensor measurements: 1 AES block, 64 bit MAC
2) 384 bits payload: AES block size 256 bit, 4 sensor measurements, 1 AES block, 128 bit MAC
3) 448 bits payload: AES block size 128 bit, 12 sensor measurements, 3 AES blocks, 64 bit MAC
4) 896 bits payload: AES block size 256 bit, 12 sensor measurements, 3 AES blocks, 128 bit MAC

An adversary that is able to manipulate just one in 1000 transmitted bits, causes a PER between 20% and 60% in our examples (Fig. 4). To highlight the impact of such a high PER, the system presented in Section III is simulated again with 25% packet loss and 50% packet loss (between sensor and controller) respectively. These simulations result in the step responses shown in Fig. 5. For 25% packet loss the given reference input can be achieved by the system (Fig. 5a) although it takes longer to reach the desired reference value when compared to the standard case shown in Fig. 3. When simulating 50% of packet loss between sensors and controller (Fig. 5b) the given reference input is hardly reached by the system. Thus, by trying to prevent deception attacks, simply applying encryption might make DoS attacks a lot easier for adversaries.

C. Stream Ciphers

One potential technology to mitigate the problems related to bit failures in cyphertexts are stream ciphers [46]. Stream ciphers encrypt each plaintext bit separately by combining it in some specified form with a corresponding bit of a keystream.



(a) Step response of system with 25% packet loss.

(b) Step response of system with 50% packet loss.

Fig. 5: Step responses for the system shown in Fig. 2 with different amounts of packet loss in network 2.

Due to this property, stream ciphers are not prone to the previously described problem; flipping one bit in the cyphertext results in one corrupted bit in the plaintext after decryption. However, the most widely used stream cipher Rivest Cipher 4 (RC4) is considered insecure due to various vulnerabilities [47] and was therefore removed from TLS [48]. Other stream ciphers, such as Salsa20 [49] are not yet widely proven to be considered. However, future developments regarding stream ciphers need to be monitored regarding their potential impact on NCS security.

As we have shown, simply applying encryption to prevent deception attacks in NCS is not an applicable approach. Therefore, in the next section, we present an enhanced approach that mitigates the drawbacks of using encryption in NCS.

V. Enhanced Approach

As simply applying encryption in an NCS has drawbacks regarding bit failures, we propose an enhanced approach that combines encryption and error correction to mitigate these drawbacks. Depending on the used network technology in

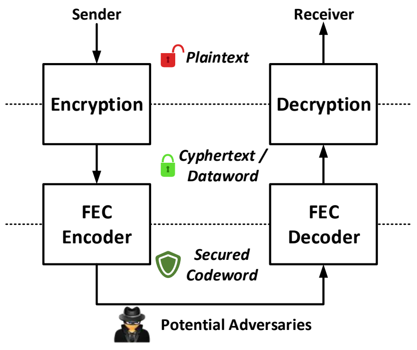


Fig. 6: Sequential JEEC applied in our approach.

an NCS, either encryption, error correction, or both technologies are applied, for instance when using TLS over an Ethernet channel. However, to make NCS security measures independent of the applied network technologies, we propose to include our approach directly into the involved components (sensors, actuators, controller).

A. JEEC

In our approach, we sequentially combine encryption and error correction to reduce the impact of bit failures in cyphertexts. More specifically, we suggest to use a suitable AES mode for AE (GCM or CCM) in combination with turbo codes FEC. We suggest this combination of algorithms for the following reasons:

- AES is a well established symmetric algorithm.
- AES provides modes of operation to apply AE.
- Turbo codes are very fast FEC algorithms that can be implemented efficiently in hardware.
- The performance of turbo codes regarding the achievable BER is close to the Shannon limit [50].

Rao [23] states that the sequential order of encryption and error correction is irrelevant, as long as both are performed. This, however is not true for our proposed approach. If the error correction would be executed before encrypting the complete data package, no advantage compared to simply applying encryption could be achieved. Therefore, we propose to first encrypt the plaintext, followed by performing the FEC encoding before sending data as shown in Fig. 6.

By using this approach, a plaintext is encrypted and the resulting cyphertext is seen as the dataword that is used as input for the FEC encoding. The resulting codeword then resembles the previous cyphertext plus data redundancy that was added by the FEC encoding. On the receiving end, the data first needs to be decoded before decrypting the corresponding cyphertext. This process allows the transmitted data to be protected by AE as well as by FEC in order to provide confidentiality, integrity, and authenticity of data and to make DoS attacks harder compared to simply applying encryption.

B. Analysis of Functionality

In contrast to our sequential approach there are also approaches that combine encryption and error correction in a single step [24], [25]. However, we propose to use separate encryption and error correction algorithms as the provided functionalities are easier to verify and proof for both components respectively. Moreover, this separation of components allows the security relevant parts to be executed on dedicated hardware to increase the provided level of security.

The security properties of AE based on suitable AES modes are demonstrated in literature [12], [14]. AES is the most used symmetric encryption algorithm, and no severe weaknesseses in the algorithm were known at the time this publication was written. The functionality of the proposed turbo codes FEC is measured in the improvement in BER compared to using no FEC. For a fixed "signal to noise" ratio (E_b/N_0), a channel using turbo codes FEC provides a BER that is lower by a factor of 10^4 compared to an unencoded channel [51]. Thus, reducing the impact of (malicious) bit errors when transmitting data in an NCS.

C. Anomaly Detection

In addition to mitigating problems related to bit failures, also anomly detection [52] could be performed using the applied FEC. A very simple approach would be to define a threshold above the expected BER of a communication channel. If the encountered bit errors are then monitored in a certain time window and exceed this specified threshold, an anomaly could be reported. However, more complex mechanisms can be implemented based on this information. We will consider such mechanisms for future work.

D. Hardware Enhancements

Due to the real-time aspects of NCS coupled with often resource constraint devices, we also propose to include dedicated hardware components into sensors, actuators, and controllers in order to provide reliable execution times. In addition to that, dedicated hardware also provides additional security features that we are going to discuss in this section. Fig. 7 illustrates an NCS with included additional hardware components necessary for our presented approach. In this figure, SC denotes a so-called security controller, while EN and DE are FEC encoders and decoders respectively.

To allow security enhancing components to be included easily into sensors, actuators, and controllers, we propose a so-called JEEC enhancement. Due to its included interfaces, the JEEC enhancement shown in Fig. 8 can easily be integrated into NCS components. The JEEC enhancement consists of the following three components:

1) **CPU:** The general purpose CPU offers interfaces to sensors and actuators as well as to the communication channel. All necessary computations such as data preprocessing or the network stacks are handled by this CPU. In addition, the CPU needs to have interfaces to the SC and FEC components.

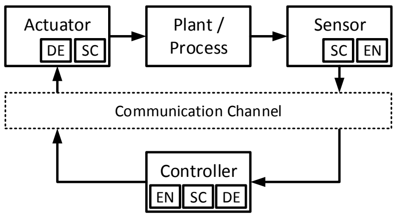


Fig. 7: Block diagram of NCS secured with our presented approach. In this block diagram, EN denotes a FEC encoder while DE denotes a FEC decoder.

2) **SC:** The SC is used to perform cryptographic operations in a secured environment. In addition to that, the SC also provides secured storage for confidential information such as key material. To provide these two functionalities, the SC needs to provide tamper resistance in order to mitigate physical attacks that try to extract or reveal confidential information. In addition, the SC also provides protection against software based attacks. A product line of SCs suitable for smart factories is, for example, offered by Infineon [53].
3) **FEC:** The FEC component is responsible to perform FEC calculations as efficient as possible. Due to constraints in size and/or price of sensors and actuators, the FEC component can also be split into decoder and encoder depending on the specified requirements.

Due to the necessary network functionality of components in an NCS, a network interface needs to be included in any case. Most of the time this also requires the inclusion of a CPU to handle the resulting overhead. Therefore, the proposed JEEC enhancement only requires to add an SC and the FEC component in most cases.

E. Advantages of Approach

The presented approach of using JEEC supported by dedicated hardware components has five advantages compared to current state of the art approaches: (i) AE provides confidentiality, integrity, and authenticity of data in a NCS; therefore, deception attacks can be mitigated. (ii) The combination of AE with FEC helps to mitigate the drawbacks resulting from bit failures in the transferred cyphertext. Despite bit failures, the same step response as shown in Fig. 3 can be achieved. Due to the sequential execution of encryption and error correction, the functionality of both components is not compromised. (iii) The information obtained in the error correction process can be used to perform additional anomaly detection. (iv) The presented JEEC enhancement can easily be included in any NCS component and thus increase the security of transferred data in an NCS. Due to using dedicated hardware components, constant runtime of cryptographic algorithms can be provided. (v) The tamper resistance provided by the SC can be used to protect confidential data if, for example, sensors are deployed where they are accessible by potential adversaries.

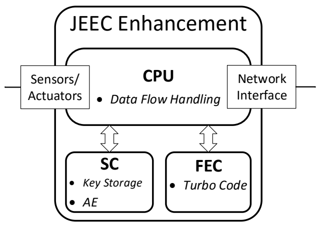


Fig. 8: JEEC enhancement for components in an NCS.

VI. Conclusion and Future Work

In this paper we have shown an encryption only approach to mitigate deception attacks in NCS. We highlight drawbacks of simply applying encryption and show the potential impact of (injected) bit failures in a NCS. To counteract these drawbacks, we propose to use JEEC to protect data confidentiality, integrity, and authenticity while also limiting the impact of adversaries that are able to artificially increase the BER in the used communication channel. We also presented a JEEC enhancement that can easily be integrated into NCS components while providing increased security and keeping delays as low as possible. As future work we plan to further investigate stream ciphers and JEEC algorithms that are able to perform encryption and error correction in a single step which might provide additional advantages.

Acknowledgment

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
[2] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, “Cyber-Physical Systems: The Next Computing Revolution,” in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 731–736.
[3] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, *Recommendations for Implementing the strategic initiative INDUSTRIE 4.0: securing the future of German manufacturing industry; final report of the Industrie 4.0 working group*. Forschungsunion, 2013.
[4] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, “Smart manufacturing, manufacturing intelligence and demand-dynamic performance,” *Computers & Chemical Engineering*, vol. 47, pp. 145–156, 2012.
[5] R. A. Gupta and M.-Y. Chow, “Networked Control System: Overview and Research Trends,” *IEEE Transactions on Industrial Electronics*, vol. 57, no. 7, pp. 2527–2535, 2010.
[6] S. Soucek and T. Sauter, “Quality of Service Concerns in IP-Based Control Systems,” *IEEE transactions on Industrial Electronics*, vol. 51, no. 6, pp. 1249–1258, 2004.
[7] G. Xie and L. Wang, “Stabilization of Networked Control Systems with Time-Varying Network-Induced Delay,” in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4. IEEE, 2004, pp. 3551–3556.
[8] J. Xiong and J. Lam, “Stabilization of linear systems over networks with bounded packet loss,” *Automatica*, vol. 43, no. 1, pp. 80–87, 2007.

[9] A. A. Cárdenas, S. Amin, and S. Sastry, "Secure Control: Towards Survivable Cyber-Physical Systems," in *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on*. IEEE, 2008, pp. 495–500.

[10] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the Impact of Stealthy Attacks on Industrial Control Systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1092–1105.

[11] E. Byres and J. Lowe, "The Myths and Facts behind Cyber Security Risks for Industrial Control Systems," in *Proceedings of the VDE Kongress*, vol. 116, 2004, pp. 213–218.

[12] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.

[13] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption," in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*. IEEE, 1997, pp. 394–403.

[14] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Science & Business Media, 2013.

[15] E. López-Trejo, F. Rodríguez-Henríquez, and A. Díaz-Pérez, "An FPGA Implementation of CCM Mode Using AES," in *International Conference on Information Security and Cryptology*. Springer, 2005, pp. 322–334.

[16] L. Henzen and W. Fichtner, "FPGA Parallel-Pipelined AES-GCM Core for 100G Ethernet Applications," in *ESSCIRC, 2010 Proceedings of the*. IEEE, 2010, pp. 202–205.

[17] R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[18] C.-L. Chen and M. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review," *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 124–134, 1984.

[19] F. Chang, K. Onohara, and T. Mizuochi, "Forward Error Correction for 100 G Transport Networks," *IEEE Communications Magazine*, vol. 48, no. 3, pp. S48–S55, 2010.

[20] C. Berrou and A. Glavieux, "Turbo Codes," *Encyclopedia of Telecommunications*, 2003.

[21] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones, and F. Pollara, "The Development of Turbo and LDPC Codes for Deep-Space Applications," *Proceedings of the IEEE*, vol. 95, no. 11, pp. 2142–2156, 2007.

[22] S. C. Kak, "Joint Encryption and Error-Correction Coding," in *1983 IEEE Symposium on Security and Privacy*, 1983.

[23] T. R. N. Rao, "Joint Encryption and Error Correction Schemes," in *ACM SIGARCH Computer Architecture News*, vol. 12, no. 3. ACM, 1984, pp. 240–241.

[24] C. N. Mathur, K. Narayan, and K. Subbalakshmi, "High Diffusion Cipher: Encryption and Error Correction in a Single Cryptographic Primitive," in *International Conference on Applied Cryptography and Network Security*. Springer, 2006, pp. 309–324.

[25] D. Gligoroski, S. J. Knapskog, and S. Andova, "Cryptcoding-Encryption and Error-Correction Coding in a Single Step," in *Security and Management*. Citeseer, 2006, pp. 145–151.

[26] R. Anderson and M. Kuhn, "Tamper Resistance - a Cautionary Note," in *Proceedings of the second Usenix workshop on electronic commerce*, vol. 2, 1996, pp. 1–11.

[27] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[28] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper Resistance Mechanisms for Secure Embedded Systems," in *VLSI Design, 2004. Proceedings. 17th International Conference on*. IEEE, 2004, pp. 605–611.

[29] A. A. Cárdenas, S. Amin, and S. Sastry, "Research Challenges for the Security of Control Systems," in *HotSec*, 2008.

[30] A. Teixeira, H. Sandberg, and K. H. Johansson, "Networked Control Systems under Cyber Attacks with Applications to Power Networks," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 3690–3696.

[31] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry, "Foundations of Control and Estimation Over Lossy Networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 163–187, 2007.

[32] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and Secure Networked Control Systems under Denial-of-Service Attacks," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2009, pp. 31–45.

[33] Y. Mo and B. Sinopoli, "Secure Control Against Replay Attacks," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE, 2009, pp. 911–918.

[34] Z.-H. Pang and G.-P. Liu, "Design and Implementation of Secure Networked Predictive Control Systems Under Deception Attacks," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1334–1342, 2012.

[35] R. A. Gupta and M.-Y. Chow, "Performance Assessment and Compensation for Secure Networked Control Systems," in *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*. IEEE, 2008, pp. 2929–2934.

[36] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli, "False Data Injection Attacks against State Estimation in Wireless Sensor Networks," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5967–5972.

[37] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. Springer Science & Business Media, 2012.

[38] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2005, pp. 19–35.

[39] M. Bellare, R. Canetti, and H. Krawczyk, "Message Authentication using Hash Functions: The HMAC Construction," *RSA Laboratories CryptoBytes*, vol. 2, no. 1, pp. 12–15, 1996.

[40] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Moderator-Ravi, "Security as a New Dimension in Embedded System Design," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 753–760.

[41] W. Zeng and M.-Y. Chow, "Optimal Tradeoff Between Performance and Security in Networked Control Systems Based on Coevolutionary Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 7, pp. 3016–3025, 2012.

[42] B. Shahian and M. Hassul, *Computer-Aided Control System Design Using MATLAB*. Prentice Hall Professional Technical Reference, 1992.

[43] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Arzén, "How Does Control Timing Affect Performance? Analysis and Simulation of Timing using Jitterbug and TrueTime," *IEEE control systems*, vol. 23, no. 3, pp. 16–30, 2003.

[44] W. Dai, "Speed Comparison of Popular Crypto Algorithms," https://www.cryptopp.com/benchmarks.html, 3 2009, (Accessed on 01/27/2017).

[45] V. J. Hernandez, A. J. Mendez, C. V. Bennett, R. M. Gagliardi, and W. J. Lennon, "Bit-Error-Rate Analysis of a 16-User Gigabit Ethernet Optical-CDMA (O-CDMA) Technology Demonstrator Using Wavelength/Time Codes," *IEEE Photonics Technology Letters*, vol. 17, no. 12, pp. 2784–2786, 2005.

[46] T. W. Cusick, C. Ding, and A. R. Renvall, *Stream Ciphers and Number Theory*. Elsevier, 2004, vol. 66.

[47] A. Klein, "Attacks on the RC4 Stream Cipher," *Designs, Codes and Cryptography*, vol. 48, no. 3, pp. 269–286, 2008.

[48] A. Popov, "Prohibiting RC4 Cipher Suites," Internet Requests for Comments, RFC Editor, RFC 7465, February 2015. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7465.txt

[49] D. J. Bernstein, "The Salsa20 Family of Stream Ciphers," in *New stream cipher designs*. Springer, 2008, pp. 84–97.

[50] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. 1," in *Communications, 1993. ICC'93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2. IEEE, 1993, pp. 1064–1070.

[51] M. A. Jordan and R. A. Nichols, "The Effects of Channel Characteristics on Turbo Code Performance," in *Military Communications Conference, 1996. MILCOM'96, Conference Proceedings, IEEE*, vol. 1. IEEE, 1996, pp. 17–21.

[52] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks Against Process Control Systems: Risk Assessment, Detection, and Response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*. ACM, 2011, pp. 355–366.

[53] J. Haid, "Hardware-based solutions secure machine identities in smart factories," *Boards & Solutions*, pp. 10–13, 2016.

Sneakernet on Wheels: Trustworthy NFC-based Robot to Machine Communication

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Sarah Haas, Rainer Matischek
Development Center Graz
Infineon Technologies Austria AG
Graz, Austria
{sarah.haas, rainer.matischek}@infineon.com

***Abstract*—Wireless communication technologies such as WiFi, ZigBee, or Bluetooth often suffer from interference due to many devices using the same, unregulated frequency spectrum. Also, wireless coverage can be insufficient in certain areas of a building. At the same time, eavesdropping a wireless communication outside a building might be easy due to the extended communication range of particular technologies. These issues affect mobile robots and especially industrial mobile robots since the production process relies on dependable and trustworthy communication. Therefore, we present an alternative communication approach that uses Near Field Communication (NFC) to transfer confidential data such as production-relevant information or configuration updates. Due to NFC lacking security mechanisms, we propose a secured communication framework that is supported by dedicated hardware-based secure elements. To show the feasibility of our approach, an Industry 4.0 inspired production process that uses our communication approach is evaluated in simulation.**

***Index Terms*—Near Field Communication; Industrial Robots; Industry 4.0; Configuration.**

I. Introduction

Having a trustworthy and dependable communication channel is essential in many scenarios, especially in industrial settings where the production process can be influenced negatively due to malfunctioning communication between involved devices. This applies in particular to so-called *smart factories* as envisioned in high-tech initiatives such as *Industry 4.0* [1]. Such smart factories are characterized by rapidly changing product demands, varying utilization of different production machinery, and usage of industrial autonomous mobile robots (IAMRs). Since IAMRs are involved in the production process, they need to transfer information between them and the involved production machinery. Although wired network technologies are usually preferred in industrial settings, wireless technologies are required in such Industry 4.0 setting due to the IAMRs not being stationary devices. Therefore, industrial wireless technologies are gaining popularity [2] although they generally suffer from the following three problems:

1) **Interference:** The 2.4 GHz frequency band that is used by communication technologies such as WiFi, Bluetooth or ZigBee is crowded due to all these technologies using the same spectrum. In addition, also devices such as cordless telephones, baby phones or other remote controlled accessories could potentially operate in the 2.4 GHz range [3]. The alternative 5 GHz range for WiFi is also already used by other devices such as cordless phones, radar, and digital satellites [4]. Due to many devices operating in the same frequency range, interference will occur and affect wireless communication.
2) **Insufficient Coverage:** Due to certain objects in buildings that dampen or even shield wireless communication (e.g. walls or large production machines) it is costly to provide good wireless coverage for every part of a certain area. For IAMRs this is a problem due to the non-deterministic behavior when navigating on a factory floor. For instance, avoiding a moving obstacle (e.g. humans) might require the IAMR to navigate to a certain part of the factory floor without sufficient wireless coverage.
3) **Eavesdropping:** The communication range of wireless technologies such as WiFi (and particularly sub-GHz ISM-band protocols) ranges up to several hundred meters. Due to this fact, eavesdropping ongoing communication could be possible outside an enclosed factory environment. This fact allows potential adversaries to eavesdrop and attack wireless communication without physical access to the smart factory, even if the communication is sufficiently secured against remote attacks.

In order to mitigate these problems, we propose to use Near Field Communication (NFC) in industrial settings due to the following three reasons. (i) NFC operates at a different frequency range than the most commonly used wireless technologies, thus reducing the risk of interference with other devices. (ii) NFC only supports peer-to-peer communication. Therefore, wireless coverage for a certain area is not required. Instead, each communication partner needs to be equipped with NFC capable devices. (iii) Due to the limited communication range of NFC, eavesdropping becomes more complicated for potential adversaries compared to other wireless communication technologies. To account for the previously discussed Industry 4.0 settings, NFC devices need to be mounted on any production machinery and IAMR that wants to communicate with other involved partners. We present a hardware extension that can be integrated into new equipment as well as retrofit to existing legacy devices. In combination with security mechanisms that we are going to present in this paper, this NFC extension is capable of providing a dependable and trustworthy communication mechanism that does not suffer from the previously mentioned problems of other wireless communication technologies. Because a production machine A can not directly communicate with a second production machine B due to the limited communication range of NFC, the machines will rely on the IAMRs moving between them to transport information from A to B. This concept of communicating originates from the early days of IT, where network connections were not that common. In a so-called *sneakernet* [5], data was transported from A to B using mediums such as floppy disks or USB sticks. In our case, this sneakernet concept will therefore be introduced to (robotic) wheels.

Contributions. Briefly, the contributions of this paper are: (i) We propose to use NFC as communication technology for industrial contexts that involve IAMRs to mitigate drawbacks of other wireless technologies. (ii) To provide a secured and reliable connection that can be used in new equipment as well as for legacy hardware, we present a hardware extension and the software components necessary for our approach. (iii) The feasibility of our presented approach is then shown in a simulation of an Industry 4.0 inspired use case.

Outline. The remainder of this paper is structured as follows. In Section II background information on the involved technologies as well as related work is discussed. The NFC-based communication approach for IAMRs is then presented in Section III. Section IV discusses and evaluates the feasibility of that approach for Industry 4.0 inspired settings. Future work and a conclusion are then given in Section V.

II. Background and Related Work

A. Industrial Robot Wireless Communication

Robot wireless communication has evolved from early technologies such as infrared towards radio frequency (RF) technologies such as Bluetooth and WiFi [6], [7]. Due to the emergence of Wireless Sensor Networks (WSNs) and the Internet of Things (IoT) in general, the mitigation of interference effects is a focus in research [8], [9], [10]. Many of the presented approaches try to minimize the effects of interference by modifying the lower layer protocols (e.g. MAC layer protocols). Although more robust solutions were proposed in research, in current practice WiFi is still seen as the de-facto standard in industrial communication due to factors such as low cost, ease of integration, and compatibility with almost any system. Therefore, special variants of wireless technologies suited for industrial use have been proposed [11].

The topic of robot wireless communication is also discussed concerning robotic inspired use cases such as the RoboCup that is seen as a testbed for future robotic solutions. Rooker and Birk [12] show that using wireless communication poses certain constraints that need to be considered in the respective robotic use case. Liu et al. [13] compare different communication technologies regarding their dependability and delay. The authors also note that wireless communication is especially critical in industrial settings. Santos et al. [14], [15] present measures on how to efficiently use a shared wireless communication channel in RoboCup competitions. In contrast to that, Birk et al. [16] propose to use cable-based communication for scenarios where reliable communication is of utmost importance such as for rescue robots. However, to the best knowledge of the authors, no satisfactory solution suited for robot to machine communication has been presented yet.

B. Near Field Communication (NFC)

NFC operates at an RF of $13.56\,\mathrm{MHz}$, typically at a range of $3\,\mathrm{cm}$-$10\,\mathrm{cm}$ and supports bit rates of 106, 212, 424, and $848\,\mathrm{kbps}$. The technology is based on several RFID standards and operates in a so-called contactless communication mode. The most common and well-known fields of application for NFC are mobile payment and access control systems [17]. NFC supports the following three standardized modes of operation: (i) *Card Emulation Mode:* The NFC device emulates a (smart) card; no RF field is generated by the device (passive mode). (ii) *Reader/Writer Mode:* The NFC device generates an RF field that is used to communicate with a passive device. The passive device also can be powered through the RF field emitted by the active device. (iii) *Peer to Peer Mode:* In this mode, a master/slave principle is used. The communication's initiator is defined as master. Independent on the chosen mode of operation, the device pairing principle of NFC is fundamentally different compared to other wireless technologies such as WiFi or Bluetooth. NFC devices are paired by bringing the two communicating devices in close proximity of each other [18]. Other than the so-called *security by proximity* principle, NFC provides no security mechanisms at the link layer; therefore, security needs to be provided by the application layer.

C. Authenticated Encryption (AE)

To provide data confidentiality, integrity, and authenticity AE comprises *symmetric cryptography* and *Message Authentication Codes (MAC)* [19]. Symmetric encryption (or private key encryption) requires both communicating partners to be in possession of the same shared secret that is then used for encryption and decryption of data. The most commonly used symmetric cryptographic algorithm is the *Advanced Encryption Standard (AES)* [20]. AES provides various modes of operation that provide different characteristics regarding execution speed or size of the implementation. Some of these modes such as AES-CCM or AES-GCM support the calculation of AE.

D. One-Time Ticket (OTT)

OTTs are similar to one-time passwords [21] in that they are used to authorize an entity to access a certain service exactly once. An OTT is issued to a certain entity and might be valid only for a given time. If the ticket holder tries to use the ticket after it has expired, access to the service is rejected. The concept of using tickets to access services is applied in widely used protocols such as *Kerberos* [22].

E. Security Controller (SC)

SCs are dedicated hardware-based secure elements that are capable of providing a secured execution environment for security-critical code as well as secured data and application storage. These functionalities can be offered by SCs due to their *tamper resistance* [23]. An SC that provides tamper resistance uses appropriate countermeasures to mitigate *physical attacks*. These kind of attacks are different to *remote attacks* as physical attacks are performed by adversaries who have physical access to the system under attack. Physical attacks are not a focus of research in robotics yet; however, the necessity to have some instance that provides reliable execution of software components in mobile robots was already proposed by Tomatis et al. [24]. Although the authors implemented their SC in software, its correct functionality is validated by a dedicated processor to improve the safety and security of the presented mobile robot platform.

III. NFC-based Communication

To enable production machinery as well as IAMRs to communicate using NFC technology, these devices need to be equipped with appropriate NFC-capable hardware. Additionally, a secured communication protocol needs to be applied to provide a trustworthy and dependable data channel. Therefore, we present NFC enhancement hardware for production machinery and IAMRs as well as a communication protocol fitted to the presented hardware. The feasibility of our approach will be evaluated int the context of the *RoboCup Logistics League's (RCLL)*. Since the RCLL's goal is to provide a factory automation testbed that resembles an Industry 4.0 motivated scenario including IAMRs [25], we consider this league as an ideal setting to evaluate our proposed approach.

A. NFC Enhancement Components

NFC communication supports different communication modes; however, all of these communication modes require an *active (master)* and a *passive (slave)* device in order to establish a connection and transfer data. In an Industry 4.0 inspired use case that involves production machinery and IAMRs, we propose to implement the IAMRs as active devices while the production machinery will be implemented as passive device. This allocation of roles would allow the IAMRs to communicate with production machinery independent of the machines current (power) state. Therefore, an approaching IAMR could, for example, turn on or activate the respective production machine, without the passive machine having to poll or wait for incoming connections. Both, the active and passive NFC enhancements are shown in Fig. 1 where the proposed hardware components are applied to an Industry 4.0 inspired simulation that was adapted from the RCLL's official simulation environment [26].

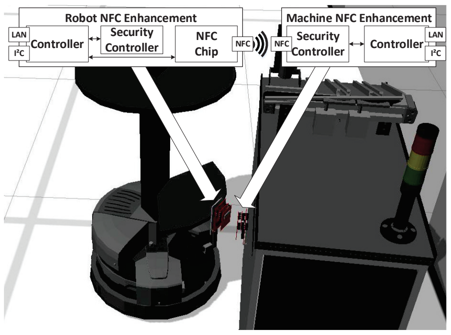


Fig. 1: Concept of NFC-based robot to machine communication applied to RCLL game simulation in Gazebo.

B. Robot (Active NFC Device)

The *Robot NFC Enhancement* component that is the active NFC device is shown on the left-hand side of Fig. 1 and comprises the following three components:

1) The *NFC Chip* provides the necessary interface to initiate and execute NFC communication. Due to the component being active, the NFC chip always needs to be powered by a power source provided by the IAMR.
2) The *SC* executes security related code that is required for the proposed communication protocol. In addition, the SC also provides secured storage for confidential information such as key material. SCs that are suitable for industrial use cases are offered, for example, in Infineon's Optiga family [27].
3) The *Controller* operates as an interface to the IAMR and thus, provides interfaces to connect the NFC enhancement to existing robotic hardware.

C. Machine (Passive NFC Device)

The *Machine NFC Enhancement* component that acts as passive NFC device is shown on the right-hand side of Fig. 1 and comprises the following two components:

1) The *SC* provides an NFC interface as well as secured execution of security relevant code. To be independent of machine states, the SC should be powered by the NFC field of the active device. SCs that provide this feature can be found, for example, in Infineon's SLE78 family.
2) The *Controller* acts as a gateway between SC and existing hardware and thus, provides appropriate interfaces such as Ethernet or I^2C.

D. Communication Protocol

In addition to the NFC enhancement components discussed in Section III-A we also propose a communication protocol

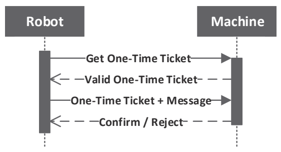


Fig. 2: Sequence diagram of communication handshake.

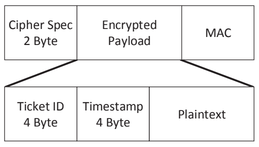


Fig. 3: NDEF packet structure.

that provides the necessary security measures entailed by the transfer of confidential data in industrial scenarios. To protect that data, we identify the following types of attacks that need to be mitigated by our approach:

Eavesdropping: Although NFC has a limited communication range, data *confidentiality* needs to be protected such that no unauthorized party has access to transferred data.

Manipulated Packets: Packet manipulation by potential adversaries must be detected in order to protect data *integrity*.

Authorized Communication Partners: Unauthorized senders must be detected in order to reject data sent by such communication partners. Thus, data *authenticity* is protected.

Replay Attacks: Captured and re-sent data that is unmodified must be identified and rejected in order to mitigate replay attacks and thus, protect the system's *functionality*.

To mitigate all of the mentioned attack types, we propose to apply AE in combination with OTTs. AE is used to provide data confidentiality, integrity, and authenticity. In addition, we use OTTs to detect and mitigate replay attacks. The master requests the OTT from the slave after initiating the communication. OTTs are directly generated at the slave when they are requested. Upon reception of that ticket, the master then is allowed to send a single message to the slave using this ticket. The sequence of this simple handshake and data sending is shown in Fig. 2.

The OTT used in our approach is composed of two components: (i) a random number, and (ii) the issued timestamp of a given ticket. In contrast to other approaches such as Kerberos, where multiple tickets can be issued and used at the same time, our approach only allows one ticket to be valid at any time. That is, if an OTT is requested, the ticket issuer (machine) stores the corresponding *Ticket ID* that comprises a random number and the issue *timestamp*. If a new ticket is requested without the old one being used, the old OTT automatically becomes invalid since it is overwritten. To request an OTT, the requester also needs to specify the timestamp of the OTT request. Both request and OTT are then only valid for a specified amount of time to mitigate replay attacks. Due to the ticket information being confidential, it also needs to be sent encrypted. The NDEF packet structure we use for the whole communication process is shown in Fig. 3. The fields included in this NDEF message are:

Cipher Spec: Specifies the algorithm and used key length for AE. This information is transmitted unencrypted.

MAC: The MAC calculated for the entire message; transmitted unencrypted.

Ticket ID: The OTT's ID (random number) that can be generated using a true random number generator provided by the SC. The ticket ID is transmitted encrypted.

Timestamp: The timestamp of either the ticket request or the ticket issuing. The timestamp is transmitted encrypted.

Plaintext: The information of the transferred message. The plaintext is transmitted encrypted.

IV. Evaluation

To evaluate our presented approach, we discuss two measures that are essential for determining the feasibility for industrial use-cases: security and communication performance.

A. Security Analysis

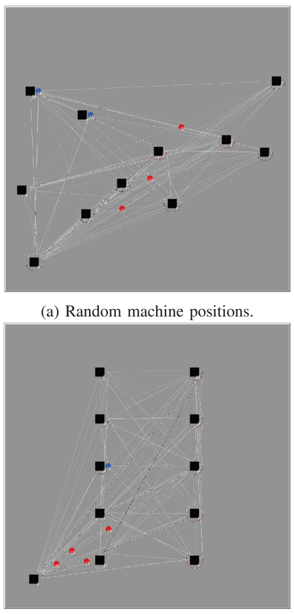
Using the NFC enhancement components discussed in Section III-A in combination with the protocol presented in Section III-D the following security-related properties can be provided by our approach:

CIA: Data **C**onfidentiality, **I**ntegrity, and **A**uthenticity are provided by the applied AE that is executed in a secured environment on the SC. The used key material that is also confidential is protected by the tamper resistance provided by the SC. Thus, eavesdropping, packet manipulation, and unauthorized access can be mitigated by our presented approach.

Replay Attacks: By using OTTs, also replay attacks are mitigated since a captured package cannot be re-sent by an attacker to provoke an unwanted machine state. Without this measure, an attacker could, for example, capture a message that configures a machine such that a certain product is produced, and re-send this message at a later time.

B. Communication Performance

We analyze and discuss communication performance-based on two use-cases that are prevalent in industrial scenarios. (i) Robot to machine communication to configure a machine for the respective production process. In network terms, this is a *unicast message*. (ii) To send information such as firmware updates or global configuration changes to all machines, a *multicast/broadcast* is required.


(a) Random machine positions.

(b) Fixed machine positions.

Fig. 4: Simulation setup for (a) randomized and (b) fixed machine positions. The black squares are simulated machines, red circles are moving IAMRs, blue circles represent IAMRs interacting with machines, and white lines represent the IAMRs' trajectories.

C. Unicast Message

To analyze the feasibility of robot to machine communication, we compare the connection timings of a point-to-point wireless TLS connection between two Raspberry PI 3 and our presented approach when sending a message of $256\,\mathrm{bytes}$. In an ideal case where only the two involved devices are in the WiFi network, we measured an average message time including the TLS handshake of about $100\,\mathrm{ms}$. Compared to our approach, the handshake also needs on average $100\,\mathrm{ms}$. That means our approach is able to perform equally as fast as TLS for small amounts of data.

D. Multicast/Broadcast Message

In addition to unicast messages, we also analyze multicast/broadcast messages in the form of a configuration update (e.g. firmware) that should be transported to all machines. Since other technologies such as WiFi or Ethernet offer a faster distribution time than our NFC-based approach, sending urgent broadcast information such as emergency stops is infeasible using our presented approach and needs to be done using other technologies. However, we believe that non-urgent configuration updates can be applied efficiently using our approach.

In this evaluation, we investigate the difference between using a *dedicated update robot* and using a wireless sensor network (WSN) inspired algorithm to deliver broadcast messages *without* having a dedicated update robot. The WSN algorithm we apply is the so-called *Trickle algorithm* [28] where a node sends an update until the same update information received from another node. As evaluation setting, we simulate an RCLL inspired factory floor consisting of 10 production machines and a varying number of IAMRs as shown in Fig. 4 where we consider two cases: (a) machine positions are randomized for each simulation run and (b) machine positions are fixed. We ran 1000 distinct simulations for both scenarios with the number of IAMRs ranging from 1 to 10. The results of that simulation are shown in Fig. 5 where the average time required for a broadcast to reach all machines is plotted. As shown in Fig. 5a, having more than 4 IAMRs would outperform having a dedicated update robot while also being more energy and cost efficient due to not requiring the otherwise necessary additional IAMR. When running the same simulation setting with fixed machine positions where an optimized update schedule for the dedicated update robot can easily be defined (see Fig. 4b), at least 6 IAMRs are necessary to outperform the dedicated update robot (see Fig. 5b).

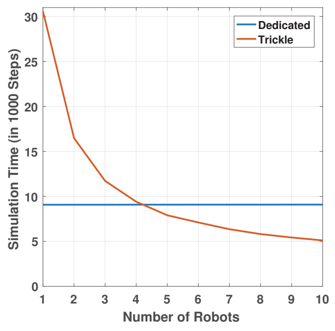
V. Conclusion and Future Work

In this paper we propose to consider NFC as an alternative to other wireless technologies in industrial contexts and RoboCup competitions. To account for the security and performance requirements of industrial data transfer, we present NFC enhancement components that can be used to equip existing as well as new devices with NFC functionality. In addition to that, we also propose a secured communication protocol that relies on AE and OTTs to provide data confidentiality, integrity, and authenticity while also mitigating replay attacks. We show the feasibility of our presented approach in terms of a security analysis as well as performance evaluations for two messaging scenarios. As future work, we plan to also evaluate WSN routing protocols regarding their efficiency if combined with our NFC-based communication approach.

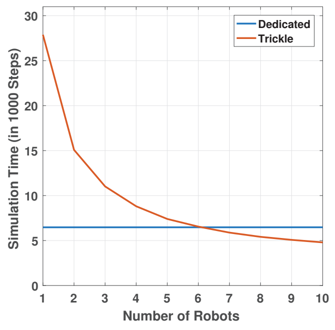
Acknowledgment

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia.

IoSense is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2016 and April 2019. More information: https://iktderzukunft.at/en/.



(a) Random machine positions.



(b) Fixed machine positions.

Fig. 5: Simulation results for (a) random machine positions and (b) fixed machine positions for 1 to 10 IAMRs.

References

[1] T. Bauernhansl, M. Ten Hompel, and B. Vogel-Heuser, *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung· Technologien· Migration*. Springer-Verlag, 2014.

[2] A. Willig, K. Matheus, and A. Wolisz, "Wireless Technology in Industrial Networks," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1130–1151, 2005.

[3] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan, "Understanding and Mitigating the Impact of RF Interference on 802.11 Networks," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 385–396, 2007.

[4] A. L. Brandão, J. Sydor, W. Brett, J. Scott, P. Joe, and D. Hung, "5GHz RLAN Interference on Active Meteorological Radars ," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 2. IEEE, 2005, pp. 1328–1332.

[5] B. Edwards, "Say Goodbye to Sneakernet Chores with LAN Inventory," *LAN Times*, vol. 12, no. 21, p. 85, 1995.

[6] S. Arumugam, R. K. Kalle, and A. R. Prasad, "Wireless Robotics: Opportunities and Challenges," *Wireless personal communications*, vol. 70, no. 3, pp. 1033–1058, 2013.

[7] Z. Wang, M. Zhou, and N. Ansari, "Ad-hoc Robot Wireless Communication," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 4. IEEE, 2003, pp. 4045–4050.

[8] T. M. Chiwewe, C. F. Mbuya, and G. P. Hancke, "Using Cognitive Radio for Interference-Resistant Industrial Wireless Sensor Networks: An Overview," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 6, pp. 1466–1481, 2015.

[9] V. C. Gungor and G. P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *IEEE Transactions on industrial electronics*, vol. 56, no. 10, pp. 4258–4265, 2009.

[10] X. M. Zhang, Y. Zhang, F. Yan, and A. V. Vasilakos, "Interference-Based Topology Control Algorithm for Delay-Constrained Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 4, pp. 742–754, 2015.

[11] A. Frotzscher, U. Wetzker, M. Bauer, M. Rentschler, M. Beyer, S. Elspass, and H. Klessig, "Requirements and current solutions of wireless communication in industrial automation," in *Communications Workshops (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 67–72.

[12] M. N. Rooker and A. Birk, "Multi-robot exploration under the constraints of wireless networking," *Control Engineering Practice*, vol. 15, no. 4, pp. 435–445, 2007.

[13] Y. Liu, M. Mazurkiewicz, and M. Kwitek, "A Study Towards Reliability- and Delay-Critical Wireless Communication for RoboCup Robotic Soccer Application," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*. IEEE, 2007, pp. 633–636.

[14] F. Santos, L. Almeida, L. S. Lopes, J. L. Azevedo, and M. B. Cunha, "Communicating among Robots in the RoboCup Middle-Size League," in *Robot Soccer World Cup*. Springer, 2009, pp. 320–331.

[15] F. Santos, L. Almeida, P. Pedreiras, L. S. Lopes, and T. Facchinetti, "An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among Mobile Autonomous Agents," in *Proc. of the Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems, WACERTS*, vol. 2004. Citeseer, 2004, pp. 657–665.

[16] A. Birk and C. Condea, "Mobile Robot Communication Without the Drawbacks of Wireless Networking," in *Robot Soccer World Cup*. Springer, 2005, pp. 585–592.

[17] V. Coskun, B. Ozdenizci, and K. Ok, "A Survey on Near Field Communication (NFC) Technology," *Wireless personal communications*, vol. 71, no. 3, pp. 2259–2294, 2013.

[18] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," in *Workshop on RFID security*, 2006, pp. 12–14.

[19] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.

[20] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Science & Business Media, 2013.

[21] N. Haller, "The S/KEY One-Time Password System," in *In Proceedings of the Internet Society Symposium on Network and Distributed Systems*, 1994.

[22] B. C. Neuman and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications magazine*, vol. 32, no. 9, pp. 33–38, 1994.

[23] R. Anderson and M. Kuhn, "Tamper Resistance - a Cautionary Note," in *Proceedings of the second Usenix workshop on electronic commerce*, vol. 2, 1996, pp. 1–11.

[24] N. Tomatis, G. Terrien, R. Piguet, D. Burnier, S. Bouabdallah, K. O. Arras, and R. Siegwart, "Designing a Secure and Robust Mobile Interacting Robot for the Long Term," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 3. IEEE, 2003, pp. 4246–4251.

[25] T. Niemueller, D. Ewert, S. Reuter, A. Ferrein, S. Jeschke, and G. Lakemeyer, "RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Testbed," in *Automation, Communication and Cybernetics in Science and Engineering 2015/2016*. Springer, 2016, pp. 605–618.

[26] F. Zwilling, T. Niemueller, and G. Lakemeyer, "Simulation for the RoboCup Logistics League with Real-World Environment Agency and Multi-level Abstraction," in *Robot Soccer World Cup*. Springer, 2014, pp. 220–232.

[27] J. Haid, "Hardware-based solutions secure machine identities in smart factories," *Boards & Solutions*, pp. 10–13, 2016.

[28] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, 2004.

IEEE JOURNAL OF RADIO FREQUENCY IDENTIFICATION, VOL. 1, NO. 1, MARCH 2017 75

Secured and Easy-to-Use NFC-Based Device Configuration for the Internet of Things

Thomas Ulz, *Member, IEEE*, Thomas Pieber, *Member, IEEE*, Andrea Höller, *Member, IEEE*, Sarah Haas, *Member, IEEE*, and Christian Steger, *Member, IEEE*

***Abstract*—Public awareness regarding security aspects in the Internet of Things (IoT) is currently rising due to regular media presence of various IoT-related security breaches. One of the major weaknesses of IoT devices is the absence of appropriate mechanisms for firmware and configuration updates. In addition, improved security concepts often result in poor usability which discourages users from relying on these concepts. Therefore, in this paper, we present an easy-to-use NFC-based configuration approach for IoT devices that is secured by appropriate security measures in software and hardware. Since industrial usage of such a configuration approach entails different requirements than home usage, we present and compare three different configuration processes. The applicability of our approach is demonstrated by two prototypical implementations, as well as a detailed security analysis. We also show that the imposed overhead due to the implemented security measures is negligible for most configuration updates.**

***Index Terms*—Near field communication, Internet of Things, security, configuration.**

I. Introduction

SECURITY aspects of the Internet of Things (IoT) and the lack thereof are a major issue due to the high number of potentially vulnerable devices. Although IoT devices are often resource constraint, they are still an enticing target for attackers since these devices are often used in botnets [1], [2]. In addition to that, each device in the IoT is equipped with some sort of sensor. This fact also increases the risk of attacks since adversaries may be interested in the provided sensor data, especially of Industrial IoT (IIoT) devices. Various studies show that between 10% and 40% of all scanned IoT devices are vulnerable to attacks because of issues such as using standard settings as well as username and passwords [3], [4] or due to exposing their configuration interface to the Internet [5]. Therefore, we consider the secured and easy-to-use configuration of IoT devices as a major gap in current research.

Regarding the configuration of IoT devices, we consider two application domains that entail different requirements in terms of security, hardware requirements, and usability.

(i) Industrial: Industrial usage of IoT devices requires high levels of security since malicious devices might interrupt a production process, reveal confidential information, or even cause physical damage and threaten human lives [6]. So-called smart factories [7] utilize a large number of IIoT devices for sensing the production process. Maintenance that involves configuration updates due to updated production- or security-requirements is essential in such an environment. By introducing a secured and easy-to-use configuration interface, even untrained staff can perform firmware updates or configuration changes. However, it is essential to protect the confidentiality and authenticity of configuration data as employees applying the configuration updates could be potential adversaries. Since in industrial settings the security aspect is of utmost importance, other factors such as the necessity for additional hardware components that increase the security can be seen as negligible.

(ii) Personal: Configuration approaches for IoT devices used in home automation or smart home [8] contexts need to provide good usability and low cost. However, also in a smart home context, configuration and firmware updates for devices need to be performed using a secured configuration interface. Similar to industrial use-cases, also in a smart home context the configuration data must be secured against various attacks for sustaining the proper functionality of the configured devices.

Independent of the domain in which IoT devices are used, configuration updates need to be performed in every phase of the device's lifecycle. Fig. 1 shows a typical IoT device configuration lifecycle that involves three major configuration phases: *initial configuration*, *reconfiguration*, and *deletion* of configuration data if an IoT device is sold or discarded. While the initial configuration might be performed in a controlled environment by the device manufacturer, all other reconfigurations of the IoT device will be performed in the potential presence of adversaries. Based on these observations, we extend and adopt the NFC-based configuration approach [9] presented at the IEEE International Conference on RFID. In addition to the configuration approach presented in that paper, we present different implementations that are tailored to the needs of certain application domains.

Manuscript received June 30, 2017; accepted August 21, 2017. Date of publication August 29, 2017; date of current version November 30, 2017. This work was supported by the Electronic Component Systems for European Leadership Joint Undertaking through the European Union's Horizon 2020 research and innovation programme and Germany, The Netherlands, Spain, Austria, Belgium, Slovakia, under Grant 692480. *(Corresponding author: Thomas Ulz.)*

T. Ulz, T. Pieber, and C. Steger are with the Institute for Technical Informatics, Graz University of Technology, 8010 Graz, Austria (e-mail: thomas.ulz@tugraz.at; thomas.pieber@tugraz.at; steger@tugraz.at).

A. Höller and S. Haas are with Infineon Technologies Austria AG, 8020 Graz, Austria (e-mail: andrea.hoeller@infineon.com; sarah.haas@infineon.com).

Digital Object Identifier 10.1109/JRFID.2017.2745510

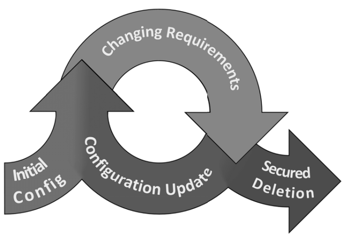


Fig. 1. Necessary configuration phases during an IoT device's lifecycle.

Contributions: For a configuration interface that is suitable for a wide range of IoT devices, we identify the following requirements:

Req1 Configuration changes for IoT devices should be possible in a secured and easy-to-use manner.

Req2 The configuration interface must be protected against remote attacks and misuse.

Req3 Implemented security features should provide high usability such that users acceptance is improved.

Req4 Potential hardware extensions must be suitable for legacy devices as well as for newly developed devices.

Req5 Device configuration must be possible in every phase of an IoT device's lifecycle.

Req6 The configuration approach should be suitable for industrial as well as home usage.

Req7 There should be no or minimal additional hardware required to perform configuration updates.

In this paper, we present an NFC-based configuration approach is capable of fulfilling these seven requirements. The presented approach provides data confidentiality, integrity, and authenticity while being intuitive to use. To account for different application domains such as industrial usage or home usage, we present different implementations of our approach that optimizes usability, and security for the respective domain. The implemented security features comprise a secured configuration protocol as well as a hardware extension that includes tamper resistant hardware to further increase provided security. This hardware extension is applicable for legacy and new IoT devices and enables device configuration in every phase of an IoT device's lifecycle. To highlight the applicability of our configuration approach, we also present a novel smart factory inspired use case which we used as a demonstrator for our prototype.

Outline: The remainder of this paper is organized as follows: In Section II we give background information on methods and technologies included in our approach and discuss related work. Section III defines our system model and lists corresponding assumptions. We then present our NFC-based configuration approach in Section IV and compare three different realizations of that approach. The security features implemented in software and hardware are then presented in Section V. In Section VI we show a prototypical implementation of our approach that was also demonstrated using a smart factory environment. The evaluation of our approach that includes a security analysis is discussed in Section VII. Future work and a conclusion are given Section VIII.

II. Background and Related Work

A. Near Field Communication (NFC)

NFC is a contactless communication standard based on RFID technology that operates at a radio frequency of 13.56 MHz [10], [11]. The typical communication range of NFC is approximately 10 cm while supporting bit rates that are multiples of 106 kbps (up to 848 kbps). Although the communication range of NFC is limited, a range of approximately 10 m for active and 1 m for passive devices should be considered as a rule of thumb for possible eavesdropping [12]. In addition to eavesdropping, also other types of attacks such as man-in-the-middle, denial-of-service or replay attacks can be applied to unsecured NFC communication [13]. Despite these potential issues, NFC is used in various domains due to its intuitive device coupling mechanism that is easy to understand for humans [14]. The mobile payment sector [15] and mobile ticketing applications [16] are the most prominent applications of NFC; however, NFC is also seen as a future building block for the IoT to link the real world with the digital world [17].

B. Symmetric Cryptography

Symmetric Cryptography requires the same cryptographic key to be used for data encryption and decryption. Due to this, the used key is considered as shared secret between communicating parties and thus, needs to be kept private. The most widely used symmetric cryptographic algorithm is the Advanced Encryption Standard (AES) [18]. Algorithms for symmetric cryptography such as AES are capable of providing data confidentiality. In order to also provide data integrity and authenticity, symmetric cryptography needs to be combined with other security measures, such as Message Authentication Codes (MAC).

Authenticated Encryption (AE) combines symmetric cryptography with MACs in a secured way such that data integrity and authenticity can be provided in addition to data confidentiality [19]. AES provides specialized modes of operation such as AES-CCM or AES-GCM that are capable of providing AE.

C. Tamper Resistant Hardware

Cryptographic algorithms such as AES can be implemented efficiently in hardware with respect to performance, power consumption, and size requirements [25]. However, such hardware components might leak information that can be used to reveal used keys or other information [26]. In addition to these so-called side-channel attacks, also invasive physical attacks can be used to reveal confidential information [27]. Tamper resistant hardware [28] such as security controllers (SCs) can be used to provide protected execution environments as well as secured data storage that mitigate side-channel and physical attacks. However, since SCs are not as powerful as general purpose controllers or dedicated hardware components, splitting the execution environment into a secured world and a

TABLE I
COMPARISON WITH RELATED WORK. THE CHARACTERISTICS REGARDING ARBITRARY PAYLOAD, PROVIDED SECURITY, SUITABILITY FOR PERSONAL USE, AND SUITABILITY FOR INDUSTRIAL USE ARE EVALUATED

Related Work	Remarks	Arbitrary Payload	Provided Security	Personal Use	Industrial Use
[20], [21]	Device pairing information is exchanged; no security is provided.	✗	✗	✗	✗
[22]	Reprogramming CRFID firmware over the air. No security provided.	✗	✗	✗	✗
[23]	NFC peer-to-peer framework that allows arbitrary data to be transmitted.	✓	✗	✓	✗
[24]	Arbitrary configuration data possible; initial configuration not secured.	✓	✗ / ✓	✓	✗
[9]	Arbitrary configuration data possible; hardware and software security.	✓	✓	✓	✗ / ✓
This work.	Arbitrary configuration data possible; hardware and software security; multiple configuration mechanisms supported.	✓	✓	✓	✓

normal world is suggested [29]. This splitting principle by implementing SCs as external hardware modules that can then be combined with general purpose CPUs.

D. NFC-Based Device Configuration

Although NFC is considered as an ideal technology for device pairing [12], using it for IoT device configuration is not that common. Most device pairing solutions (e.g., [20] and [21]) have in common that only pairing information can be transmitted and that no security measures are integrated. Wu *et al.* [22] present an approach for reprogramming computational RFID (CRFID) tags over the air. The authors propose to use the Electronic Product Code (EPC) protocol to update the firmware of passive CRFID tags. The drawbacks of the presented approach that only complete firmware images can be flashed as well as missing security features. Serfass and Yoshigoe [23] present a framework for NFC communication in wireless sensor networks. This framework allows arbitrary data to be transferred using NFC but does not provide security measures. Haase *et al.* [24] propose to NFC-enabled mobile phones for NFC-based sensor and actuator configuration in smart home contexts. The authors also discuss security measures. However, the initial device configuration is unencrypted, and no key update mechanism is provided. Ulz *et al.* [9] present a QR and NFC-based hybrid configuration approach that implements security measures in hardware and software. This approach is further extended in this paper such that different update mechanisms are supported to suit personal and industrial usage scenarios. Also, an automated key derivation process is included. The discussed related work is compared with the approach presented in this paper in Table I.

III. SYSTEM MODEL AND ASSUMPTIONS

When designing a configuration interface for IoT devices, we are faced with the following three problems:

1) Configuration data for IoT devices might contain confidential information, especially when considering IIoT devices. This configuration data needs to be transferred using an untrusted channel including potential adversaries that eavesdrop or manipulate the transferred data.

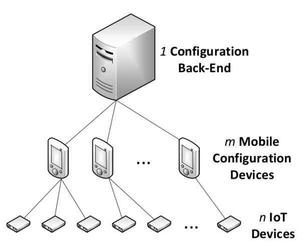


Fig. 2. System model we assume for IoT device configuration.

2) IoT and IIoT devices might be operated in unsupervised environments, thus configuration data needs to be stored at the device such that confidential information cannot be extracted, even if adversaries have unlimited physical access to the device under attack.
3) The configuration interface might be subjected to misuse, both unintentional and intentional.

To mitigate these problems, the configuration approach we are presenting in this paper is based on the system model shown in Fig. 2 that comprises the following three entities:

IoT Device: The device that needs to be configured. There is no limitation on the number of devices; we generally assume *n* IoT devices in our system model.

Configuration Device: The mobile device used to transfer configuration data to the IoT device. We also do not limit the number of configuration devices in our model; therefore, we assume *m* such configuration devices.

Configuration Back-End: The back-end is responsible for administrating all configurations that are done using our presented approach. This means that the back-end needs to keep track of all configuration changes. Therefore, we assume *one* configuration back-end in our system model.

Based on our system model, we assume the configuration back-end that operates as a global configuration storage to be trustworthy and sufficiently secured against any kind of attack. We further assume that all configuration changes must be initiated and authorized by this back-end. Thus, the back-end has knowledge of device configurations from all devices administrated by that back-end. Regarding configuration data,

```
temp_threshold: 0x5D
sampling_rate: 0x01
wifi_key: 0x778D9FE1325BAB9811
```

Fig. 3. Example configuration that contains non-confidential information as well as confidential information.

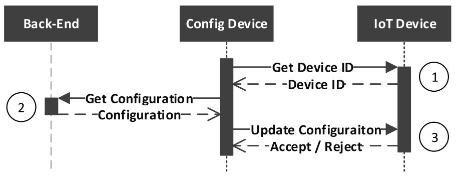


Fig. 4. Sequence of NFC communication for IoT device configuration.

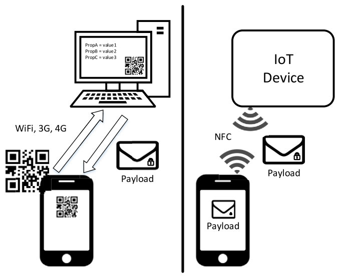


Fig. 5. Hybrid configuration approach: On the left hand side, configuration data is fetched from the back-end using a QR code. On the right hand side, the configuration is transferred to the IoT device using NFC.

no assumption concerning the content is made. That is, we assume configuration data to contain non-confidential information such as temperature thresholds or sampling rates, as well as confidential information such as keys or WiFi passwords. An exemplary configuration is shown in Fig. 3.

IV. Configuration Mechanisms

Depending on the domain in which IoT devices are used, different requirements regarding a configuration interface can be defined. For example, protecting confidential information is of utmost importance for devices used in industrial settings. For IoT devices used by private persons, a configuration approach should be as easy-to-use as possible and should not require any costly additional hardware. Therefore, we present three different configuration mechanisms supported by our approach. Each of these mechanisms provides different advantages and disadvantages that we are going to discuss. All three mechanisms implement the security measures that are discussed in Section V. However, for simplicity, we only discuss the principle process of each configuration mechanism in this section.

A. NFC-Based Configuration

In the NFC-based configuration mechanisms, NFC and a wireless data connection are used during device configuration according to the protocol shown in Fig. 4. The protocol comprises the following three steps: ① The configuration device queries the IoT device for its identifier using NFC. ② Using this device ID, a configuration is fetched from the configuration back-end. In this case, we assume that the device is managed by that back-end and that a new configuration that needs to be applied is available. ③ If a configuration for that given device is available; it is transferred to the IoT device again using NFC.

Advantages/Disadvantages:

+ This mechanism is easy to use. The device to configure is automatically identified, and the corresponding configuration is fetched. Due to the fact that users only need to bring the mobile configuration device with a working data connection close to the IoT device, this approach is very well suited for remote support.
– Active data connection is required to fetch configuration data which might not be possible in industrial settings. Also, initial data such as a symmetric key needs to be synchronized between IoT device and configuration back-end (e.g., by manufacturer).

B. NFC and QR Code-Based Configuration

As second configuration mechanism, we propose a QR-code and NFC-based hybrid configuration approach [9]. The principle of that approach is shown in Fig. 5. As can be seen there, QR-codes are used to transfer configuration data from the configuration back-end to the mobile configuration device, while NFC is used to transfer the configuration data from the mobile configuration device to the IoT device. Due to the limited maximum payload of a QR code [30] and to support different usage scenarios, we propose the following two different modes of operation:

1) The complete configuration payload is stored in the QR code, which allows a maximum payload of roughly 2900 bytes of data. Therefore, we denote this type as *inline* QR code. Inline QR codes do not require the mobile configuration device to have an active network connection. Thus, these QR codes can be distributed and used where no working network is available.
2) If the configuration data is larger than the maximum payload of 2900 bytes, only an URL pointing to the configuration stored at the back-end is included in the QR code. The mobile configuration device then needs to fetch the configuration data from the back-end, as in the first mechanism. We denote this type of QR code as *URL* QR code.

Advantages/Disadvantages:

+ This mechanism is easy to use. In addition to that, if the inline mode is used, no active network connection is required when configuring devices. The inline mode allows using this approach in situations where no working network connection is available. QR codes can also easily be distributed by paper, for instance, by including the

TABLE II
COMPARISON OF PRESENTED CONFIGURATION MECHANISMS

	NFC-based	**Hybrid Inline**	**Hybrid URL**	**Location Aware**
Device ID read	✓	✗	✗	✓
Works offline	✗	✓	✓/✗	✗
Configuration size unlimited	✓	✗	✓	✓
Suited for remote support	✓	✓	✓	✗
Complex back-end upkeep	✗	✗	✗	✓
Suited for personal use	✓	✓	✓	✗
Suited for industrial use	✓	✓	✓	✓
Camera required	✗	✓	✓	✗
Wireless interface required	✓	✗	✓	✓
Infrastructure required	✗	✗	✗	✓

initial configuration of a device inside the packaging the device is sold in. Also, as shown in Fig. 5, configurations can be directly downloaded from the monitor where a configuration is edited.

– The mobile configuration device needs to have a working camera in order to scan QR codes. Also, this mechanism potentially favors potential user errors since the user needs to be aware which configurations need to be downloaded beforehand when using the inline mode.

C. Location-Aware Configuration

The third implemented configuration mechanism does not require the device to identify itself. Instead, localization mechanisms are used to determine the mobile configuration device's position and the closest administrated IoT device. As soon as the configuration process is initiated by the user, the corresponding configuration is fetched from the configuration back-end as in the NFC-based approach. However, instead of requesting a configuration based on the IoT device's ID, the estimated coordinates of the mobile configuration device are sent to the configuration back-end. The back-end then replies with the most probable device configuration that is then sent to the IoT device using NFC.

Advantages/Disadvantages:

+ This mechanism is easy to use. The device to configure is automatically identified, and the corresponding configuration is fetched. Due to using localization to identify the corresponding IoT device, NFC communication between IoT device and mobile communication device are reduced to a minimum.

– Active data connection is required to fetch configuration data which might not be possible in industrial settings. Also, the configuration back-end needs to be configured such that the location of each administrated device is known to the back-end. While outdoor localization using GPS might be accurate and easy, indoor localization is an ongoing research topic [31]. Also, indoor localization requires an infrastructure that is used to calculate the mobile configuration device's position.

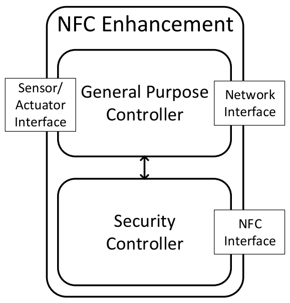


Fig. 6. NFC Enhancement that can be integrated into any IoT device.

In our prototypical implementation, we used a Received Signal Strength Indicator (RSSI) based trilateration algorithm [32] relying on wireless access points. The localization quality using such an approach strongly depends on factors such as fading, obstacles, or the temperature [33]. However, in our setting, we were able to achieve accuracies of less than 1 m which will be sufficient for most settings.

D. Comparison

Since all three previously presented mechanisms have different advantages and disadvantages, we compare them in Table II regarding their suitability for different usage scenarios of IoT devices. As can be seen there, no algorithm is suited best for all scenarios; thus, the applied mechanism needs to be chosen based on the context in which IoT devices need to be configured.

V. SECURITY MECHANISMS

A. NFC Enhancement

In order to allow new as well as retrofit IoT devices to be equipped with the proposed NFC configuration interface, we present a hardware extension suitable for these two types of devices. This so-called *NFC Enhancement* provides a number of interfaces for different purposes:

Sensor/Actuator Interface: This interface is used to connect sensors and actuators that are used by the IoT device with the NFC enhancement component.

Network Interface: This interface is used to connect the IoT device with a network. The IoT device's core functionality is accessible through this interface.

NFC Interface: This interface is used for device configuration. The NFC interface will also be used to harvest energy during the configuration process such that no additional power source is necessary for device configuration.

A concept of the NFC enhancement component containing all three interfaces is shown in Fig. 6. The component includes two controllers, a *general purpose controller* and a *SC*. Due to including two controllers, responsibilities can be split perfectly according to the capabilities of both controllers. On the one hand, the general purpose controller provides interfaces

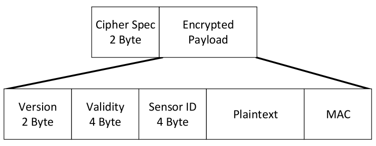


Fig. 7. NDEF packet structure used to protect configuration data.

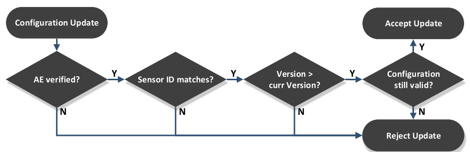


Fig. 8. Decision process of configuration update rejection or acceptance.

to sensors, actuators, and to the network which requires computational power. In addition, computational expensive data aggregation, manipulation, and processing can be done by the general purpose controller. The less powerful SC, on the other hand, offers a secured execution environment and protected storage. Cryptographic operations are executed by the SC, and confidential information is stored in the SC's protected storage. The SC also offers an NFC interface that is used for IoT device configuration. This NFC interface is also capable of harvesting energy from an NFC field such that the SC does not require any additional power source. Due to this, IoT devices can be configured at any time, for instance, during the manufacturing process without attaching any power source. In addition, SCs in our approach are considered as trusted entity, since their correct functionality is evaluated based on a common criteria [34] certification process. That is, all security critical operations performed by the SC in our approach can be considered as being properly secured and correct.

B. Data Transfer Protocol

Configuration data in our approach needs to be transferred using different untrusted channels (NFC, QR, WiFi, . . .). Therefore, security measures need to be applied to provide data confidentiality, integrity, and authenticity. In addition to that, information regarding configuration data is necessary such that the IoT device's SC is capable of deciding if a configuration should be rejected or accepted and thus applied. For NFC data transfer we implement a protocol based on the NFC Data Exchange Format (NDEF) [35]. Due to the low overhead of NDEF, we use the same data structure when transferring data using QR-codes or a network connection. Although NDEF provides some security measures such as signatures [36], we did not rely on these measures since they are insufficient and shown to be vulnerable to attacks [37]. Instead, confidential information in our approach is protected by applying AE in the MAC-then-Encrypt mode that is also used in the Transport Layer Security (TLS) protocol [38]. The complete structure of NDEF packets in our approach including additional security related fields ins shown in Fig. 7. All but the **Cipher Spec** field is protected by AE in our approach. This field specifies the applied cryptographic algorithm and key size used for AE. This information needs to be transmitted unencrypted since it is required for decryption. All other fields are contained in the encrypted payload.

Version: A version number identifying the specific configuration version. The IoT device will reject configuration updates with a configuration number less or equal to the currently applied configuration.

Validity: If the current realtime of the IoT device is later in time than the specified validity, a configuration update will be rejected. For this check, we assume there is a secured time source for the IoT device.

Device ID: If the specified device ID does not match the actual device's ID, the configuration is not indented for the respective device and thus rejected.

MAC: The MAC corresponding to the transmitted plaintext. It is calculated by a so-called one-way function [39] and is part of the AE process.

Using the additional information together with AE, the IoT device either rejects or accepts the configuration update. The flowchart in Fig. 8 summarizes the decision process.

VI. Prototype

To evaluate the presented approach with respect to feasibility, usability, and functionality we implemented a prototype comprising the presented security measures in hardware and software. This prototype, shown in Fig. 9 consists of the following components:

Sensor/Actuator: An air pressure sensor without any actuator is used to represent the IoT device's functionality.

General Purpose Controller: An Infineon XMC4500 microcontroller from the Cortex M4 family was used as a general purpose controller. This microcontroller provides connection interfaces such as USB, I^2C, and Ethernet.

SC: As SC, an Infineon SLE78 that is CC EAL5+ certified was used. The SC is connected to the general purpose controller via I^2C. The SLE78 SC provides security features such as secured data storage and code execution while also including a contactless interface for NFC communication.

Mobile Configuration Device: We used an off-the-shelf Nexus S mobile phone as NFC-enabled mobile configuration device. The device is running Android 4.1.2 Jelly Bean to use API level 14 and above that supports the latest NDEF functionality of Android.

Configuration Back-End: The configuration back-end that is not pictured in Fig. 9 was realized on a standard Windows PC in this prototype. The required functionality is written in NodeJS such that any computer that is capable of running JavaScript can run the back-end.

A. Smart Factory Prototype

In addition to the prototype shown in Fig. 9 we also present a prototypical smart factory use case in which we evaluated our presented approach. The evaluation is done in an

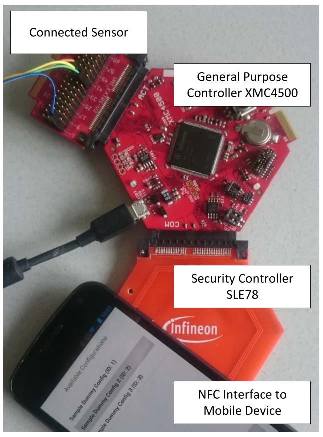

Connected Sensor
General Purpose Controller XMC4500
Security Controller SLE78
NFC Interface to Mobile Device

Fig. 9. NFC Enhancement prototype comprising of an Infineon XMC4500 microcontroller used as general purpose controller and an Infineon SLE78 SC.

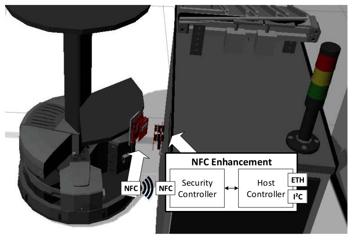


Fig. 10. Simulated smart factory prototype in RCLL environment.

Industrie 4.0 [40] inspired smart factory setting that is simulated in the RoboCup Logistics League (RCLL) [41]. The league is intended as a testbed for smart factory inspired robotic solutions where a number of autonomous mobile robots need to transport semi-finished and finished individualized products between production machines. In this process, robots also need to configure machines such that the desired products are manufactured. The configuration in this context is done using wireless communication. The main issues with wireless communication in industrial settings are real-time capability and reliability [42]. Reliability of wireless communication is most often compromised by interference of various technologies operating in the same frequency spectrum [43]. Especially the 2.4 GHz and 5 GHz spectra are used by many technologies such as WiFi, Bluetooth, or wireless phones. Hence, when using these technologies interference is a common problem.

We applied the NFC-based configuration approach presented in this paper to the RCLL context and used an existing simulation environment [44] that we extended such that NFC related characteristics could be simulated [45]. Using this simulation environment, we investigated if the achieved production capacity of a system is negatively influenced if our NFC-based approach is used instead of WiFi. Fig. 10 shows the prototype during a simulation run. Since the point-to-point performance of our approach is comparable to WiFi (see Section VII), no drawback in terms of production capacity could be observed. However, due to the limited communication range of NFC, interference caused by robots simultaneously configuring machines was reduced, and thus, the reliability of machine configuration could be improved.

VII. Evaluation

In addition to the presented prototype that demonstrates the feasibility and usability of our presented approach, we also evaluated the provided security level, the overhead, and the performance of our approach.

A. Security Analysis

To demonstrate the security improvements achieved by the implemented security measures in our approach, we present a comprehensive security analysis. In this analysis, we list involved **E**ntities, **A**ssets that are threatened and need to be protected, the **T**hreats, **C**ountermeasures applied to mitigate these threats, and **R**esidual Risks for threats that cannot be mitigated. We also list **As**sumptions that are made in the context of this analysis. An overview of the security analysis in Goal Structure Notation (GSN) is shown in Fig. 11. In this notation, the threats for each asset are highlighted. In addition, for each threat existing countermeasures or residual risks are shown. The assets that are protected by our secured configuration approach are: **(A1) Configuration Data:** Since configuration data may contain confidential information such as keys, the confidentiality, integrity, and authenticity of this information needs to be protected. **(A2) Device Functionality:** Correct functionality of IoT devices must not be compromised due to the inclusion of our proposed configuration interface. That is, any attack targeting this interface must not disturb proper operation of the device.

Threats for these assets can be posted by the following entities: **(E1) IoT Device Manufacturer:** The manufacturer of the device. Manufacturing includes all components such as sensors, actuators, and NFC enhancement. **(E2) IoT Device Owner:** Any user that is in possession of the IoT device and thus, allowed to make configuration changes. **(E3) Person Applying Configuration Updates:** Any person that is trying to apply configuration updates at the IoT device. This could be a different person than the device owner, especially in industrial settings. **(E4) Adversary:** Any adversary that can access the IoT device's configuration interface, either remotely or physically.

Before investigating potential threats, certain assumptions are made in order to restrict the scope of this threat analysis: **(As1) Configuration Back-End:** The configuration back-end that maintains all current configurations is assumed to be properly secured against any type of attack. **(As2) SC Certification:** The SC used in the NFC Enhancement component is assumed to be certified to a CC security level of

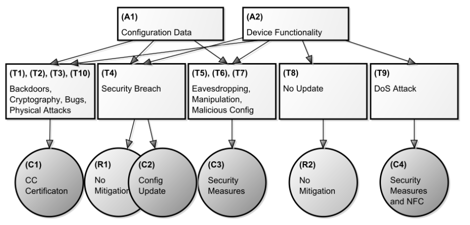

(A1) Configuration Data
(A2) Device Functionality
(T1), (T2), (T3), (T10) Backdoors, Cryptography, Bugs, Physical Attacks
(T4) Security Breach
(T5), (T6), (T7) Eavesdropping, Manipulation, Malicious Config
(T8) No Update
(T9) DoS Attack
(C1) CC Certificaton
(R1) No Mitigation
(C2) Config Update
(C3) Security Measures
(R2) No Mitigation
(C4) Security Measures and NFC

Fig. 11. Security analysis in GSN.

at least EAL5+ and thus, is capable of mitigating physical attacks.

Considering all identified assets, entities, and corresponding assumptions our approach is now analyzed regarding potential threats. For each threat, one or multiple countermeasures and potential residual risks will be given. **(T1) Backdoors:** There might be intentional or unintentional backdoors included in the configuration interfaces hardware or software. **(C1) CC Certification:** The CC certification process investigates and mitigates this type of threat. **(T2) Weak Cryptographic Algorithms:** The algorithms used in the configuration approach might be susceptible to attacks due to weaknesses in the used algorithm or due to using too short keys. **(C1) CC Certification:** The CC certification process investigates and mitigates this type of threat. **(T3) Bugs:** Security related functionality implemented by the manufacturer might include weaknesses or even bugs. **(C1) CC Certification:** The CC certification process investigates and mitigates this type of threat. **(T4) Security Breach:** Initial keys stored by the manufacturer could be lost in a security breach or disclosed in any other form, intentional or unintentional. **(C2) Easy Configuration:** Changing configuration parameters such as the initial key can be easily performed by users. **(R1) No Update:** If the initial key is not updated, this threat cannot be mitigated. **(T5) Eavesdropping Configuration Data:** An adversary might try to eavesdrop configuration data and thus, learn confidential information. **(C3) Security Measures:** The security measures presented in this publication provide effective mitigation of this threat. **(T6) Manipulate Configuration Data:** An adversary might try to manipulate transferred configuration data, either while being transferred from configuration back-end to mobile configuration device, or while being transferred from mobile configuration device to IoT device. **(C3) Security Measures:** The security measures presented in this publication provide effective mitigation of this threat. **(T7) Malicious Configuration:** An adversary might try to apply outdated configuration data or configuration data that is intended for a different device. **(C3) Security Measures:** The security measures presented in this publication provide effective mitigation of this threat. **(T8) No Update:** A malicious user does not apply any necessary update. Thus, he basically performs a denial-of-service (DoS) attack targeting the IoT device's correct functionality. **(R2) No Countermeasure:** Our approach cannot provide any countermeasure against users that do not apply intended updates. **(T9) DoS Attack:** An adversary might try to perform DoS attacks targeting the configuration interface. **(C4) Security Measures and Proximity:** The security measures presented in this publication provide effective mitigation of this threat. In addition, DoS attacks targeting the configuration interface can only be performed by adversaries in close proximity to the IoT device. **(T10) Physical Attacks:** An adversary that has physical access to the IoT device might try to reveal confidential information by performing physical attacks on the device. **(C1) CC Certification:** The CC certification process investigates and mitigates this type of threat.

The list of discussed threats as well as the respective countermeasures and residual risks is not exhaustive by any means, but it reflects the threats that we consider as most crucial for the presented NFC-based configuration approach. Of the eleven identified threats, nine can be effectively mitigated while residual risks remain only for two threats. This highlights the improved level of security provided by the presented configuration approach.

B. Overhead and Performance

The implemented security measures in the NDEF protocol (see Section V-B) entail an overhead of transferred data. This overhead can be split into a static part (O_{static}) and a variable part ($O_{variable}$). The static overhead resulting from the additionally included information (cipher spec, version, validity, and sensor ID) can easily be calculated by summing up the field sizes specified in Fig. 7.

$$O_{static} = 2B + 2B + 4B + 4B = 12B \tag{1}$$

The variable overhead depends on the selected cryptographic algorithms and the corresponding key sizes. For this evaluation, we assume a MAC length of 32 B. In addition to that, also the padding required by block ciphers needs to be accounted for. For this evaluation, we assume AES that has a block size of 16 B which entails an overhead due to the padding of 0 B - 15 B. The total overhead O is then calculated by summing up all incidental overheads.

$$O = O_{static} + O_{dynamic} \tag{2}$$

An overview of the resulting overhead relative to the transferred configuration data size up to 4 kB is shown in Fig. 12. The sawtooth pattern results from the varying padding overhead that oscillates in the range of 0 B - 15 B. For typical configuration sizes of 300 B, less than 15 % of the transferred data will result from security imposed overhead.

To evaluate the performance of our approach, we measured the time that was required to transfer a configuration package with a typical size of 300 B. The complete data transfer including key agreement, encryption and decryption, and configuration acceptance/rejection decision process requires roughly 350 ms. Compared to that, transmitting the same amount of data using a secured TLS channel over a direct WiFi connection between two Raspberry PIs takes roughly 200 ms. However, it has to be considered that the processing power of a Raspberry PI is by far larger than the used components in our prototype and that a direct WiFi connection was used

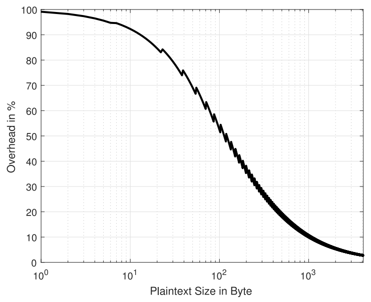


Fig. 12. Percentage of overhead relative to transferred configuration data.

between the devices. Therefore, the timing difference between these two approaches can be assumed as negligible.

VIII. Future Work and Conclusion

In this paper, we present a secured NFC-based configuration approach that is suitable for personal and industrial IoT devices alike. To account for the different requirements in these two domains, we present different configuration mechanisms that provide different advantages and disadvantages. In order to provide data confidentiality, integrity, and authenticity we present security measures in hardware and software. The NFC enhancement component we present, can be used for new and retrofit IoT devices. The NDEF based protocol we present is secured by applying authenticated encryption in combination with additional information that is used to validate configuration data. The feasibility and usability of our approach are demonstrated by two prototypes, while the provided security, the resulting overhead, and the performance are also evaluated. As future work, we plan to further extend our approach such that the correct change of configuration data can be attested in our system.

References

[1] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, Feb. 2017.

[2] D. Peraković, M. Periša, and I. Cvitić, "Analysis of the IoT impact on volume of DDoS attacks," in *Proc. 33rd Symp. New Technol. Postal Telecommun. Traffic (PosTel)*, 2015, pp. 295–304.

[3] A. Cui and S. J. Stolfo, "A quantitative analysis of the insecurity of embedded network devices: Results of a wide-area scan," in *Proc. ACM 26th Annu. Comput. Security Appl. Conf.*, Austin, TX, USA, 2010, pp. 97–106.

[4] M. Patton *et al.*, "Uninvited connections: A study of vulnerable devices on the Internet of Things (IoT)," in *Proc. IEEE Joint Intell. Security Informat. Conf. (JISIC)*, The Hague, The Netherlands, 2014, pp. 232–235.

[5] Y. M. P. Pa *et al.*, "IoTPOT: Analysing the rise of IoT compromises," in *Proc. 9th USENIX Conf. Offensive Technol.*, Washington, DC, USA, 2015, p. 9.

[6] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial Internet of Things," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6.

[7] D. Lucke, C. Constantinescu, and E. Westkämper, "Smart factory—A step towards the next generation of manufacturing," in *Proc. 41st CIRP Conf. Manuf. Syst.*, Tokyo, Japan, 2008, pp. 115–118.

[8] R. Harper, *Inside the Smart Home*. London, U.K.: Springer, 2006.

[9] T. Ulz *et al.*, "SECURECONFIG: NFC and QR-code based hybrid approach for smart sensor configuration," in *Proc. IEEE Int. Conf. RFID (RFID)*, Phoenix, AZ, USA, 2017, pp. 1–6.

[10] *Information Technology—Telecommunications and Information Exchange Between Systems—Near Field Communication—Interface and Protocol (NFCIP-1)*, ISO/IEC Standard 18092, 2004.

[11] *ISO/IEC 21481 Information Technology—Telecommunications and Information Exchange Between Systems—Near Field Communication—Interface and Protocol (NFCIP-2)*, ISO/IEC Standard 21481, 2005.

[12] E. Haselsteiner and K. Breitfuß, "Security in near field communication (NFC)," in *Proc. Workshop RFID Security*, 2006, pp. 12–14.

[13] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger, "NFC devices: Security and privacy," in *Proc. IEEE 3rd Int. Conf. Availability Rel. Security (ARES)*, Barcelona, Spain, 2008, pp. 642–647.

[14] D. López-de-Ipiña, J. I. Vazquez, and I. Jamardo, "Touch computing: Simplifying human to environment interaction through NFC technology," in *Proc. 1as Jornadas Científicas Sobre RFID*, vol. 21. 2007, pp. 1–12.

[15] J. Ondrus and Y. Pigneur, "An assessment of NFC for future mobile payment systems," in *Proc. IEEE Int. Conf. Manag. Mobile Bus. (ICMB)*, Toronto, ON, Canada, 2007, p. 43.

[16] V. Coskun, B. Ozdenizci, and K. Ok, "A survey on near field communication (NFC) technology," *Wireless Pers. Commun.*, vol. 71, no. 3, pp. 2259–2294, 2013.

[17] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.

[18] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*. Heidelberg, Germany: Springer, 2013.

[19] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2000, pp. 531–545.

[20] A. Matos, D. Romao, and P. Trezentos, "Secure hotspot authentication through a near field communication side-channel," in *Proc. IEEE 8th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, Barcelona, Spain, 2012, pp. 807–814.

[21] M. Jung, J. G. Park, J. H. Kim, and J. Cho, "Interoperability between medical devices using near field communication," in *Proc. Int. Conf. Inf. Science Appl. (ICISA)*, Suwon, South Korea, 2013, pp. 1–4.

[22] D. Wu, M. J. Hussain, S. Li, and L. Lu, "R2: Over-the-air reprogramming on computational RFIDs," in *Proc. IEEE Int. Conf. RFID (RFID)*, Orlando, FL, USA, 2016, pp. 1–8.

[23] D. Serfass and K. Yoshigoe, "Wireless sensor networks using android virtual devices and near field communication peer-to-peer emulation," in *Proc. IEEE Southeastcon*, Jacksonville, FL, USA, 2012, pp. 1–6.

[24] J. Haase, D. Meyer, M. Eckert, and B. Klauer, "Wireless sensor/actuator device configuration by NFC," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, 2016, pp. 1336–1340.

[25] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications," in *Proc. Int. Conf. Inf. Technol. Coding Comput. (ITCC)*, vol. 2. Las Vegas, NV, USA, 2004, pp. 583–587.

[26] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2005, pp. 157–171.

[27] A. Becher, Z. Benenson, and M. Dornseif, "Tampering with motes: Real-world physical attacks on wireless sensor networks," in *Proc. Int. Conf. Security Pervasive Comput.*, 2006, pp. 104–118.

[28] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems," in *Proc. IEEE 17th Int. Conf. VLSI Design*, Mumbai, India, 2004, pp. 605–611.

[29] M. Sabt, M. Achemlal, and A. Bouabdallah, "The dual-execution-environment approach: Analysis and comparative evaluation," in *Proc. IFIP Int. Inf. Security Conf.*, 2015, pp. 557–570.

[30] T. J. Soon, "QR code," *Synthesis J.*, vol. 2008, pp. 59–78, 2008.

[31] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proc. ACM 16th Annu. Int. Conf. Mobile Comput. Netw.*, Chicago, IL, USA, 2010, pp. 173–184.

[32] N. K. Sharma, "A weighted center of mass based trilateration approach for locating wireless devices in indoor environment," in *Proc. 4th ACM Int. Workshop Mobility Manag. Wireless Access*, Torremolinos, Spain, 2006, pp. 112–115.

[33] N. Baccour *et al.*, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Trans. Sensor Netw.*, vol. 8, no. 4, p. 34, 2012.

[34] D. Mellado, E. Fernández-Medina, and M. Piattini, “A common criteria based security requirements engineering process for the development of secure information systems,” *Comput. Stand. Interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[35] “NFC data exchange format (NDEF),” NFC Forum, Wakefield, MA, USA, Tech. Rep. NDEF 1.0, Jul. 2006.

[36] M. Roland and J. Langer, “Digital signature records for the NFC data exchange format,” in *Proc. IEEE 2nd Int. Workshop Near Field Commun. (NFC)*, 2010, pp. 71–76.

[37] C. Mulliner, “Vulnerability analysis and attacks on NFC-enabled mobile phones,” in *Proc. IEEE Int. Conf. Availability Rel. Security (ARES)*, Fukuoka, Japan, 2009, pp. 695–700.

[38] H. Krawczyk, “The order of encryption and authentication for protecting communications (or: How secure is SSL?)” in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 310–331.

[39] M. Naor and M. Yung, “Universal one-way hash functions and their cryptographic applications,” in *Proc. ACM 21st Annu. ACM Symp. Theory Comput.*, Seattle, WA, USA, 1989, pp. 33–43.

[40] T. Bauernhansl, M. T. Hompel, and B. Vogel-Heuser, *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung Technologien Migration*. Wiesbaden, Germany: Springer-Verlag, 2014.

[41] T. Niemueller *et al.*, “RoboCup logistics league sponsored by festo: A competitive factory automation testbed,” in *Automation, Communication and Cybernetics in Science and Engineering 2015/2016*. Cham, Switzerland: Springer, 2016, pp. 605–618.

[42] A. Frotzscher *et al.*, “Requirements and current solutions of wireless communication in industrial automation,” in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Sydney, NSW, Australia, 2014, pp. 67–72.

[43] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan, “Understanding and mitigating the impact of RF interference on 802.11 networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 385–396, 2007.

[44] F. Zwilling, T. Niemueller, and G. Lakemeyer, “Simulation for the robocup logistics league with real-world environment agency and multi-level abstraction,” in *Robot Soccer World Cup*. Cham, Switzerland: Springer, 2014, pp. 220–232.

[45] T. Pieber, T. Ulz, and C. Steger, “SystemC test case generation with the gazebo simulator,” in *Proc. IEEE 7th Int. Conf. Simulat. Model. Methodol. Technol. Appl. (SIMULTECH)*, 2017, pp. 65–72.

Thomas Ulz received the first M.Sc. degree in information and computer engineering and the second M.Sc. degree in computer science from the Graz University of Technology in 2016, focussing on Security, Embedded Systems, Robotics, and Machine Learning, where he is currently pursuing the Ph.D. degree in information and computer engineering with the Institute for Technical Informatics. His research currently focuses on security aspects of Internet of Things devices.

Thomas Pieber received the M.Sc. degree in information and computer engineering from the Graz University of Technology in 2016, focussing on Embedded Systems, Mobile Computing, and Robotics, where he is currently pursuing the Ph.D. degree in information and computer engineering with the Institute for Technical Informatics. His research currently focuses on energy efficiency of Internet of Things devices.

Andrea Höller received the M.Sc. degree in information and computer engineering, focusing on System-on-Chip Design and Information Security and the Ph.D. degree with a thesis on software-based fault-tolerance for resilient embedded systems from the Graz University of Technology, in 2013 and 2016, respectively. From 2013 to 2016, she has conducted research on dependability for cyber-physical systems with the Institute for Technical Informatics. In 2016, she joined Infineon Technologies Austria AG, where she is currently working on the future of secured authentication and encryption for cyber-physical systems and the Internet of Things.

Sarah Haas received the first M.Sc. degree in information and computer engineering and the second M.Sc. degree in computer science from the Graz University of Technology, in 2016, focussing on Robotics, Machine Learning, Embedded Systems, and Big Data Analysis, where she is currently pursuing the Ph.D. degree in information and computer engineering. She is with Infineon Technologies Austria AG. Her research currently focuses on security of industrial mobile robots.

Christian Steger received the Dipl.-Ing. (M.Sc.) degree in 1990, and the Dr.Techn. (Ph.D.) degree in electrical engineering from the Graz University of Technology, Austria, in 1995. He graduated from the Export, International Management and Marketing course with the Karl-Franzens-University of Graz in 1993 and completed the Entrepreneurship Development Program at MIT Sloan School of Management, Boston, in 2010. He is a Strategy Board Member of the Virtual Vehicle Competence Center (ViF, COMET K2) in Graz, Austria. Since 1992, he has been an Assistant Professor with the Institute for Technical Informatics, Graz University of Technology, where he heads the HW/SW codesign group with the Institute for Technical Informatics.

Chapter 10

Cyber-Physical System and Internet of Things Security: An Overview

Thomas Ulz
Graz University of Technology, Austria

Sarah Haas
Infineon Austria AG, Austria

Christian Steger
Graz University of Technology, Austria

ABSTRACT

An increase of distributed denial-of-service (DDoS) attacks launched by botnets such as Mirai has raised public awareness regarding potential security weaknesses in the Internet of Things (IoT). Devices are an attractive target for attackers because of their large number and due to most devices being online 24/7. In addition, many traditional security mechanisms are not applicable for resource constraint IoT devices. The importance of security for cyber-physical systems (CPS) is even higher, as most systems process confidential data or control a physical process that could be harmed by attackers. While industrial IoT is a hot topic in research, not much focus is put on ensuring information security. Therefore, this paper intends to give an overview of current research regarding the security of data in industrial CPS. In contrast to other surveys, this work will provide an overview of the big CPS security picture and not focus on special aspects.

INTRODUCTION

In recent years, customers' demands for personalized products increased rapidly (Adomavicius & Tuzhilin, 2005). To account for these customer requests, traditional mass production facilities need to be altered such that personalized products can be manufactured in a cost-effective way. One possible

DOI: 10.4018/978-1-5225-2845-6.ch010

way to achieve this goal is to make factories smart by enabling the interconnection of all devices involved in the manufacturing process. The term Smart Factory was introduced by Zuelke (2010) when he described his vision of a factory-of-things. According to Zuelke, in such a factory-of-things, smart objects could interact with each other using Internet of Things (IoT) and cyber-physical systems (CPSs) concepts (Weiser, 1991) (Mattern & Floerkemeier, 2010), (Lee, 2008). Recent high-tech initiatives such as Germany's Industry 4.0 further extend the vision of smart factories beyond providing cost effective personalized products. In these initiatives, smart factories utilize self-organizing multi-agent systems that operate without human assistance. In addition, also big data analysis will play a major role in future smart factories in order to optimize production processes.

To account for the envisioned functionalities of a smart factory, devices ranging from battery operated sensors up to big data servers need to be interconnected. Due to the diversity of devices which might be resource constraint, standard web protocols such as HTTP often cannot be applied, thus making Web of Things (WoT) concepts infeasible. Instead, lightweight protocols and concepts from the IoT can be applied. IoT concepts in industrial contexts offer advantages but also critical disadvantages. One advantage is the possibility to control and reconfigure machines such that personalized products can be manufactured (Gibson, Rosen, Stucker et al., 2010). However, connecting production machinery to the Internet also results in issues that do not arise in traditional production facilities. Machinery that is accessible through the Internet implicates security and safety issues; security breaches in industrial contexts may lead to the loss of highly confidential data or may even threaten employees' lives (Cheng, Zhang, & Chen, 2016).

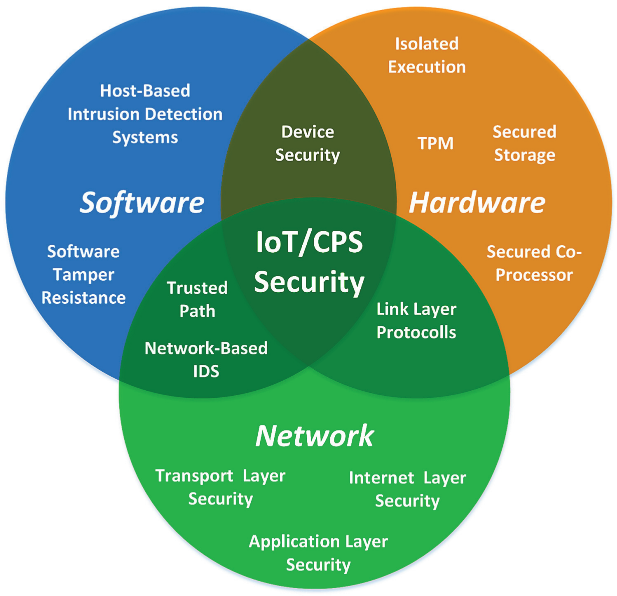
Security, however, is often not considered in industrial IoT research as current main topics in research are enabling technologies and production strategies. Therefore, the intention of this work is to present an overview of current security related research on industrial IoT and CPSs. In contrast to other works, the authors intend to give a broad overview of security aspects, not focusing on single special topics. This broad overview, however, is given in a compact form to present the big picture of IoT and CPS security. An overview of all topics discussed in this work is presented in the big picture shown in Figure 1.

This work is structured as follows. As background information, attack taxonomies are given and different types of attacks are discerned in Section Attack Taxonomies. Also in this Section, challenges of CPS security and differences compared to traditional IT systems are discussed. This section also lists current research trends regarding CPS security. To highlight the importance of IoT and CPS security, recent attacks targeting IoT devices and CPSs are presented in Section Attacks. The subsequent sections discuss security enhancing technologies on different layers as shown in Figure 1. Section Network Security lists network related security problems and solutions. Issues and fixes related to device security are discussed in Section Device Security where hardware and software related topics are discussed. This work is then concluded with Section Conclusion where also current hot research topics are briefly discussed.

ATTACK TAXONOMIES

Attacks in information security often are associated with the resulting "CIA" (Confidentiality, Integrity, Availability) triad security attributes that are broken by the respective attack. The three security attributes, as defined in principle by Saltzer and Schroeder (1975), are:

Figure 1. Big picture of IoT and CPS related security measures as discussed in this work
Source: Big Picture.



- **Confidentiality:** The property of information that is protected from unauthorized persons, entities or processes.
- **Integrity:** The property of information that is protected from being modified in an authorized, undetected manner during its entire life-cycle.
- **Availability:** Describes the property of information being available when it is needed such that the information system can serve its purpose.

Besides these three most commonly referred to security attributes, there are also many other attributes such authenticity, possession, or non-repudiation.

Attacks on cyber-physical systems can be further divided into two categories (Ravi, Raghunathan, Kocher, & Hattangady, 2004). Attacks corresponding to the first category, logical attacks can be conducted using existing communication interfaces. Logical attacks typically target software weaknesses and can be done remotely. The second category of attacks, physical and side-channel attacks, usually requires an attacker to have physical access to the hardware. An attacker that is able to physically access the targeted hardware is then able to attack both, the software and hardware weaknesses of a system.

Logical Attacks

Logical attacks can target either a single device or a whole network. Hansman and Hunt (2005) give a categorization of possible attacks on network and computer systems. Most of these attacks can also occur in CPSs or IoT systems.

- **Viruses and Worms:** Malicious software components executed at the targeted system. The malicious code is often spread and even updated via a network. Viruses and worms might compromise data confidentiality and integrity as well as the availability of systems.
- **Exploits:** By using weaknesses in a software or hardware implementation, attackers are able to perform various operations such as injecting malicious code or revealing secret data. Exploits include, for example, buffer overflows or various code injections (e.g. SQL-injection, cross site scripting). These attacks also target the confidentiality, integrity and availability of a system.
- **Denial of Service (DoS):** In DoS attacks, the aim is to make the service provided by the targeted system unavailable. This can be achieved in various ways such as flooding a server with a high amount of requests. This kind of attack targets the availability of a system.
- **Network Attacks:** In this type of attack, network related vulnerabilities are used to attack a system. The goal is often to redirect traffic to a malicious system. Examples of attacks include man-in-the-middle attacks or DNS spoofing. The targeted CIA attributes are confidentiality, integrity and availability.
- **Password Attacks:** Attackers try to reveal users' password in order to gain unauthorized access to a system. THIS can be done, for example, via brute force attacks or dictionary attacks. All three CIA attributes might be compromised by such attacks.

The above listed attacks are general attacks on network and computer systems. Pasqualetti et al. (2013) discuss logical attacks that are mainly relevant for CPSs. In contrast to the previously mentioned attacks, these attacks often target information regarding the physical process attached to a CPS.

- **Deception Attacks:** In deception attacks, wrong data (such as sensor data or control data) is injected into the CPS. On the one hand, an attacker might inject data that is false and unrelated to the system; however, such attacks might easily be detected. On the other hand, an attacker might first try to learn a system's behavior and then inject data based on the learned standard behavior (stealthy deception attack) which is harder to detect. This type of attack targets the integrity and as a possible consequence also the availability of a CPS.
- **Replay Attacks:** In replay attacks, an attacker first captures data produced by a CPS which also can be encrypted. After capturing that data, the attacker then injects this data into the system at a later point in time. In contrast to deception attacks, an attacker does not necessarily need to have any knowledge about the sent data. This type of attack also targets the integrity and availability of a CPS.

Physical and Side-Channel Attacks

Physical and side-channel attacks always compromise the confidentiality, integrity, and availability of CPSs due to the information revealed by them. These attacks can be categorized by two criteria: an attacker's behavior and the attack's degree of invasiveness. First, an attacker's behavior can be used to distinguish attacks (Kocher, Lee, McGraw, Raghunathan, & Moderator-Ravi, 2004):

- **Active Attack:** An attacker actively tries to induce faults into the hardware, for example, by injecting power spikes. In unprotected devices, this may lead to failures in the executed software which then might reveal keys or weaknesses of the implementation.
- **Passive Attack:** An attacker passively observes physical properties of the hardware, for example, a CPU's power consumption. These physical properties might reveal details about the implementation or even confidential data such as keys.

The second type of categorization can be done depending on an attack's degree of invasiveness (Kocher, Lee, McGraw, Raghunathan, & Moderator-Ravi, 2004):

- **Invasive Attack:** In invasive attacks, there is no limit regarding the actions an attacker might take. Possible actions include removing the packaging, probing internal bus lines or even permanent changes to the circuits of a hardware element.
- **Semi-Invasive Attack:** In semi-invasive attacks the attacker does not change the attacked hardware. Although semi-invasive attacks often include the de-packaging of hardware, no physical contact with the internal components is made. Desired faults are injected by, for example, using radiation or light to attack the hardware.
- **Non-Invasive Attack:** In non-invasive attacks the attacker observes properties of the hardware without damaging or changing it. Such properties include side-channels (Le, Canovas, & Clédiere, 2008) such as the power consumption or the timing of a certain part of software.

An overview of potential attacks, categorized by these two criteria can be seen in Table 1. All of these attacks can be applied to CPSs.

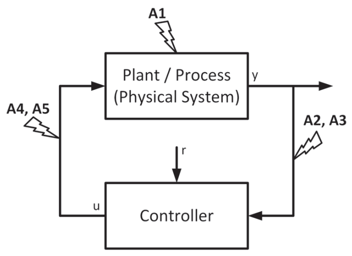
CPSs are, by definition, seen as an embedded system or controller that is attached to a physical process. The physical process can be monitored using sensors or actively influenced by a CPS using actuators. A wide range of systems can be classified as CPS, such as smart grids, process control systems, (autonomous) robotic systems, (autonomous) car systems, medical devices, and many more. Such systems provide various potential points of attack inside a CPS as depicted in Figure 2. There, *y* is seen as the

Table 1. Physical attacks categorized according to Section Attack Taxonomies

	Active	Passive
Invasive	Circuit Changes, Forcing, …	Probing, …
Semi-Invasive	Light Attacks, Radiation Attacks, …	Inspecting the Hardware, EM Attacks, …
Non-Invasive	Spike Attacks, Low Voltages, …	Side-Channel Attacks (Power, Timing, …)

Attacks extracted from (Weingart, 2000) and (Anderson, Bond, Clulow, & Skorobogatov, 2006).

Figure 2. Potential points of attacks inside a CPS
Adapted from (Cardenas, Amin, & Sastry, Secure Control: Towards Survivable Cyber-Physical Systems, 2008). Source: Attack Points.



output of a process, for example, sensor measurements and *u* are the control commands sent to the physical process. The potential attacks on such CPSs and their implications can be categorized into five groups (adapted from Cardenas, Amin, & Sastry, Secure Control: Towards Survivable Cyber-Physical Systems, 2008). A1 are attacks targeted directly at the physical process itself. The aim of such attacks could be actuators or even a physical attack against the plant. A2 are so called deception attacks. In these kinds of attacks the adversary induces false information $\tilde{y} \neq y$ by attacking, for instance, a sensor. Possible information that can be forged includes measurements or the time associated to measurements. A3 represent DoS attacks where an attacker prevents the controller from receiving the physical process' output. A4 represent attacks where an adversary attacks the controller and induces false control commands $\tilde{u} \neq u$. These manipulated control commands could harm and destroy the physical process. A5 is similar to A3; an adversary attacks the data transportation from the controller to the physical process. Because of its nature, this attack is also a DoS attack.

Challenges and Research Trends

The challenges regarding IoT security are manifold (Jing, Vasilakos, Wan, Lu, & Qiu, 2014); therefore, the authors identify four major challenges. (i) A high number of insecure devices is supposed to be already connected to the Internet. Many devices were shown to be vulnerable to simple intrusion attacks by a large scale study (Cui & Stolfo, 2010). The study results show that about 13 percent of all discovered devices are configured with factory default passwords; the carna botnet revealed 1.2 million devices with weak passwords or no password set at all (Le Malécot & Inoue, 2014). (ii) There is believe that current security measures such as public-key infrastructures will not scale to the large number of IoT devices (Roman, Najera, & Lopez, 2011). (iii) As most IoT devices are highly constraint devices, finding a single weak link to attack could be an easy task for attackers. Therefore, efficient security algorithms need to be developed to mitigate attacks. (iv) Being in control of a single device could already lead to failures of many services. Thus, each involved component needs to be secured.

In the context of CPSs, even more security challenges arise compared to traditional ICT systems (Cardenas et al., 2009). For example, a challenge could arise through the necessity for security related software updates that often require reboots of the updated system or additional redundant systems to prevent reboots. Reboots are critical as the physical process also needs to be stopped in order to avoid potential problems. Restarting a physical process such as a power plant will take magnitudes longer than restarting, for example, a personal computer (PC). Regarding the CIA attributes a shift in priorities between CPSs and ICT systems can be found. Protecting data confidentiality is crucial for systems processing private data while for CPSs that interact with a physical process their availability is in most cases more important than data confidentiality. Another challenge is the need for real-time availability of CPSs. Many traditional IT systems such as web services only need to provide availability of their service with no requirements regarding real-time aspects. However, the major difference between CPSs and traditional IT systems is that CPSs are connected to a physical process. Attacks might target the physical process itself or intend to damage the process which even might threaten human lives. In traditional it systems, attacks mostly target the processed information.

Lun et al. (Lun, D'Innocenzo, Malavolta, & Di Benedetto, 2016) describe current trends and hot topics in research related to CPS security. We expand this list by current research trends regarding the challenges mentioned by us:

- Countermeasures against special attacks targeting CPSs (deception, false data injection, etc.) (Kim & Poor, 2011; Lo & Ansari, 2013)
- Prevention, detection and mitigation of attacks (Chaojun, Jirutitijaroen, & Motani, 2015), (Huang, Li, Campbell, & Han, 2011)
- Ensuring integrity of data in case of attacks (Kwon, Liu, & Hwang, 2014), (Vuković & Dán, 2014)
- Security measures for resource constraint devices such as sensors (Mishra, Shoukry, Karamchandani, Diggavi, & Tabuada, 2015; Mo, Weerakkody, & Sinopoli, 2015; Höller, Druml, Kreiner, Steger, & Felicijan, 2014)
- Security concepts for specific CPS application fields (e.g. Power Grid, Autonomous Vehicles, etc.) (Xue, Wang, & Roy, 2014; Zhu & Basar, 2015)
- Security measures for controllers (Dadras, Gerdes, & Sharma, 2015; Urbina et al., 2016)

Lun et al. also state that focus regarding CPS application is almost entirely on power grids. The research interest in the field of communication aspects is very low which is surprising as communication is an essential topic for all networks. This work shows that many topics are addressed currently but many more need to be approached to provide solutions for real world applications of CPSs.

ATTACKS

Cyber-attacks targeting CPSs became the focus of public attention in recent years. The probably best known cyber-attack that focused on physically destroying a target was Stuxnet. Stuxnet's only goal, contrary to traditional worms, was to harm a target instead of stealing, manipulating or erasing information. However, Stuxnet was not the first attack that harmed a physical process. Some other and earlier attacks are listed in Table 2 (collected and adapted from Miller & Rowe, 2012).

Table 2. Attacks

Attack	Reported Description and Sources
Siberian Pipeline Explosion (1982)	The first known cyber-attack targeting critical infrastructure. A trojan planted in a control system caused the explosion of a Siberian pipeline (Daniela, 2011).
Chevron Emergency Alert System (1992)	Chevron's alert system was disabled by a fired employee. The undetected attack threatened people in 22 states in the USA and parts of Canada (Denning, Cyberterrorism: The Logic Bomb versus the Truck Bomb, 2000).
Worcester, MA Airport (1997)	An attacker successfully disabled a telephone computer that serviced Worcester Airport. The outage affected services such as the aviation control tower, the airport fire department or the airport security and thus threatened human lives (Denning, Cyberterrorism: The Logic Bomb versus the Truck Bomb, 2000).
Gazprom (1999)	Attackers supported by a disgruntled employee gained access to the central switchboard that controls the gas flow in pipelines. The attackers reportedly used a trojan horse to gain access (Denning, Cyberterrorism: Testimony Before the Special Oversight Panel on Terrorism Committee on Armed Services US House of Representatives, 2000).
Davis-Besse Nuclear Power Plant (2003)	The Davis-Besse nuclear power plant in Ohio, USA was infected by a worm that disabled the plant's safety parameter display system and the plant process computer for several hours (Beggs, 2006).
CSX Corporation (2003)	Train signaling systems in Florida, USA were shut down by a fast spreading worm. There are no major incidents caused by this attack; however still many lives were threatened by it (Nicholson, Webber, Dyer, Patel, & Janicke, 2012).
Stuxnet (2010)	Stuxnet attacked Iranian nuclear facilities exploiting zero-day vulnerabilities. The worm tried to destroy centrifuges by frequently switching between high and low speeds which ultimately led to the failure of these centrifuges (Langner, 2011).
Night Dragon (2011)	Five global energy and oil companies were attacked by a combination of social engineering, trojans and using Windows exploits. The attacks are said to have been ongoing for about two years. Although no damage has been detected, data such as operational blueprints were stolen (Nicholson, Webber, Dyer, Patel, & Janicke, 2012).
Flame (2012)	Flame, a piece of malware was found on computers operating in Iran, Lebanon, Syria, Sudan and other places in the Middle East and North Africa. The malware was used to extract documents but also opened a backdoor that allowed adding any new functionality that could be used to harm the systems under attack (Lee D., 2012).
HAVEX (2014)	The HAVEX malware primarily targeted the energy sector, collecting data from attacked systems and leaving backdoors to control systems. Through these backdoors, the connected physical process could be controlled in a malicious way and therefore could also be manipulated or destroyed by attackers (Hentunen & Tikkanen, 2014).
Black Energy (2015)	Initially known as a botnet (Lee, Jeong, Park, Kim, & Noh, 2008), Black Energy changed its purpose in 2015. Ukrainian power plants infected were infected with a trojan through a backdoor opened by Black Energy. The trojan then tries to destroy the system by deleting certain files relevant for booting the system (ICS-CERT, 2016).

Mainly collected and adapted from (Miller & Rowe, 2012).

New Attack Dimension

All attacks listed in Table 2 reportedly successfully manipulated or destroyed a physical process. Recently, attacks have not tried to harm a physical process or device, but have tried to capture devices in order to use them in botnets (Dagon, Gu, Lee, & Lee, 2007). Because of their large number, IoT devices are a favored target to be used in botnets (Pa, et al., 2015). In addition, IoT devices are online 24/7, which makes them even better suited to be used in botnets. In 2014 both, Sony's and Microsoft's gaming platforms were attacked by a large number of infected IoT devices (Somani, Gaur, & Sanghi, 2015). The number of infected devices is rising since, culminating in a recent attack that reached traffic peaks of 620 gigabits per second (Gallagher, 2016). In this attack, an IoT botnet called Mirai was involved in attacking DNS services. The Mirai botnet comprises of devices such as WiFi routers and IP cameras.

According to a study, the number of DDoS attacks in 2016 increased by 71 percent when compared to 2015 (Daws, 2016). The attacks originated from countries shown in Figure 3. As the number of IoT devices will continue to grow, also the number of associated attacks will increase.

NETWORK SECURITY

In the context of initiatives such as Industry 4.0 (Referat, 2013), more CPSs are going to be connected to the Internet. These CPSs process confidential data and control production relevant processes. Therefore, securing the data transfer between these devices is of high importance. According to the Internet protocol suite, there are four layers: Link Layer, Internet Layer, Transport Layer and Application Layer (see Figure 4). All four of these layers are capable of providing different security measures that are going to be discussed in this section. In most cases, security measures on multiple levels are needed as sufficient security cannot be provided by one layer due to information not being available. For instance, information such as IP addresses that might be required to detect certain kinds of attacks are not available at the link layer.

Link Layer

On the link layer, there are a couple of protocols that are used in the IoT. The protocols that are of most interest when discussing security are wireless protocols, as this type of communication offers by far more weaknesses than wired communication. For example, eavesdropping in a wireless network can be as easy as positioning a malicious device in the communication range of the attacked devices. However,

Figure 3. Top origins of DDoS attacks
(Daws, 2016). Source: DDoS Origins.

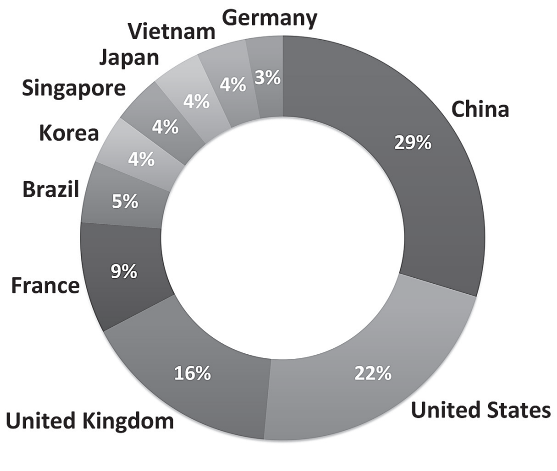
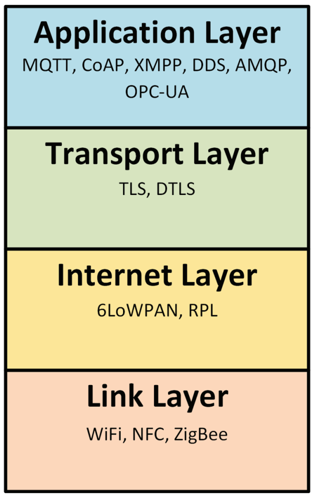


Figure 4. TCP/IP protocol architecture layers with protocols discussed in this work
Source: TCP.



also wired communication technologies can be attacked if communication is not properly secured. The technologies that are seen as most promising (Zorzi, Gluhak, Lange, & Bassi, 2010) for the IoT are Wireless LAN, Near Field Communication and 802.15.4 based technologies such ZigBee. Therefore, these three technologies are analyzed regarding their security vulnerabilities in industrial usage (Plósz, et al., 2014).

- **Wireless LAN (Wi-Fi):** Wi-Fi is a wireless communication technology that has its origin in personal computers. Wi-Fi operates in frequency bands of 2.4 GHz or 5 GHz with a communication range of approximately 100 meters. This rather high range allows adversaries to attack Wi-Fi networks without being, for instance, in the same building. Wi-Fi standards include authentication and encryption mechanisms such as WEP and WPA. The security of Wi-Fi communication therefore relies on the confidentiality of these keys. If, for instance, an adversary is in possession of a Wi-Fi network's WPA key, ongoing communication can be read by the adversary. Therefore, key cracking key cracking by eavesdropping Wi-Fi communication is one of the biggest threats against this technology.
- **Near Field Communication (NFC):** NFC is a wireless communication technology that is based on RFID standards. NFC has a typical communication range of 10 cm and operates at a radio frequency of 13.56 MHz. Because of NFC's limited communication range, attacks need to be conducted in close proximity to the NFC devices. Although communication is limited to a couple centimeters, eavesdropping might be possible in a range of up to 10 m (Haselsteiner & Breitfuß, 2006). Currently there is no dedicated NFC standard for authentication and access control.

Therefore, unauthorized access to NFC devices is seen as the most critical issue with NFC. To mitigate the problem of unauthorized access, application layer security must be implemented.

- **802.15.4/ZigBee:** Zigbee is intended to be used in low power wireless networks. ZigBee operates in the ISM (industrial, scientific and medical) radio bands and allows for communication up to a range of approximately 20 m. The 802.15.4 standard allows higher layers to provide security; therefore, ZigBee implements security features such as authentication, encryption, and key establishment. ZigBee also defines a so-called Trust-Center that is a special node responsible for storing network keys (Lennvall, Svensson, & Hekland, 2008). The biggest weakness of ZigBee is rogue nodes that might not be detected. Also, in many installations master keys are factory installed (Baronti et al., 2007). If these keys are extracted by an adversary with physical access to a device, security of the attacked network is severely threatened.

Plósz et al. (2014) list potential attacks for each of the three technologies, Wi-Fi, NFC, and ZigBee. All attacks are then assessed according to their likelihood and impact. This assessment yields a final rank highlighting the risk of each attack. Attacks with a major or even critical risk are listed in Table 3. The threats listed are of different nature, mostly depending on the wireless technology's architecture and communication range.

Improving security at the link layer usually is a complex task, as the overhead imposed by security at this layer is significant compared to the transmitted payload. Because most CPSs are resource constraint devices, a large number of 802.11 based networks operate without any cryptographic protection (Hurley, 2003). To mitigate this issue, Karlof et al. (Karlof, Sastry, & Wagner, TinySec: A Link Layer Security Architecture for Wireless Sensor Networks, 2004) present a link layer security architecture tailored for resource constraint devices. The authors have chosen to implement security measures at the link layer,

Table 3. Link Layer protocols and possible attacks with major or critical risks

	Threat	Highest Risk
Wi-Fi	WEP Shared Key Cracking	Confidentiality
	WPA-PSK Cracking	Confidentiality
	Application Login Theft	Confidentiality, Authenticity
	Intercepting TCP, SSH, SSL	Confidentiality, Integrity, Authenticity
	Evil Twin Access Point	Confidentiality, Availability, Integrity, Authenticity
	Device Cloning	Confidentiality, Integrity, Authenticity
NFC	Clone or Modify Portable Reader Device	Confidentiality, Authenticity
	Wormhole / Relay Attack	Authenticity
	Rogue Node	Confidentiality, Availability, Integrity, Authenticity
	Unauthorized Access to Node	Confidentiality, Authenticity
ZigBee	Rogue Node	Confidentiality, Integrity, Authenticity
	Device Cloning / Firmware Replacement	Confidentiality, Integrity, Authenticity
	Security Parameter Extraction by Physical Access	Confidentiality, Authenticity
	Plaintext Key Capture	Confidentiality, Integrity, Authenticity

Plósz et al., 2014.

because CPSs often communicate in a many-to-one pattern. In this pattern, many sensors and actuators communicate their data to a central base station, which makes traditional end-to-end security such as SSH, TLS or IPSec infeasible. The approach uses authenticated encryption to secure the transported payload at the link layer.

Internet Layer

Security measures implemented at the Internet layer increased in popularity when the Internet Engineering Task Force (IETF) started the IP Security Working Group. The goal of this working group was to design cryptographic security for IPv6 that could also be ported to IPv4 (Oppliger, 1998). The result, IPSec, is widely known and supported nowadays as it is capable of providing data confidentiality and integrity by mitigating network attacks. IPSec is very popular and integrated in IPv6; therefore, the focus for the rest of this section will be on approaches that are capable to detect attacks at the Internet layer.

IETF introduced IPv6 for Low power Wireless Personal Area Networks (6LoWPAN) for resource constraint IoT devices. 6LoWPAN enables these devices to be connected to the Internet by compressing standard IPv6 headers. Kasinathan et al. (2013) present an approach that includes an Intrusion Detection System (IDS) and DoS detection into 6LoWPAN. A network based IDS analyzes the 6LoWPAN traffic to detect intrusion attempts and to raise alerts in case of detected attacks. The IDS approach helps to detect and mitigate network and DoS attacks and thus, increases confidentiality, integrity and availability of a CPS. The presented network based approach requires the inclusion of IDS Probe nodes that are allowed to analyze all packets, irrelevant of the actual recipient. In case of an attack, the IDS then alerts the DoS protection manager that then further collects data to verify the potentially ongoing DoS attack. The authors claim that their distributed hybrid approach is capable of detecting DoS attacks reliably.

The Routing Protocol for Low-Power and Lossy Networks (RPL) is a standardized routing protocol for IoT devices that use 6LoWPAN. Attacks against routing protocols were successfully applied against wireless sensor networks (WSNs) as well (Karlof & Wagner, 2003). Wallgren et al. (Wallgren, Raza, & Voigt, 2013) propose to place IoT IDSs at the root nodes of RPL routing trees, thus, giving the IDS a global view. This allows the routing protocol, for instance, to exclude malicious nodes from the routing tree in order to prevent network attacks. By excluding malicious nodes, confidentiality, integrity, and availability of a CPS can be increased. In their approach, ICMPv6 messages protected by IPSec with ESP are used to detect anomalies in the network. Wallgren et al. also show that the inclusion of an IDS introduces only a small overhead in power consumption of about 10 percent.

Transport Layer

Transport Layer Security (TLS) is considered to be of utmost importance in IoT applications (Garcia-Morchon, Kumar, Struik, Keoh, & Hummen, 2013). Although considered by some as an application layer protocol, the authors put TLS into the transport layer as its name suggests. Similar to the Internet layer, conventional protocols of the transport layer cannot directly be applied to IoT devices because of resource limitations. Especially in low-power lossy networks, protocols such as conventional TLS cannot be applied. TLS is a stream oriented protocol building on TCP that suffers from frequent packet loss in the form of delays. As an alternative to TLS, the Datagram Transport Layer Security (DTLS) protocol that is using UDP was introduced. DTLS provides the same protection mechanisms as TLS and does not influence the underlying packet transport. Thus, DTLS is able to provide confidentiality and integrity of transferred data by mitigating network attacks. A gateway from the Internet (TLS) to a lossy IoT network (DTLS) is proposed by Brachmann et al. (2012) in order to be able to provide end-to-end security between networks. The DTLS protocol is often used to implement IoT related security mechanisms such as a two-way authentication (Kothmayr, Schmitt, Hu, Brünig, & Carle, 2013) that uses X.509 certificates and an Elliptic Curve Diffie Hellman (EC-DH) key agreement process. By using this approach, both communication partners are authenticated and an encrypted communication is enabled. Kothmayr et al. (2013) show that securing a connection using DTLS imposes minimal overhead for the involved devices and also conclude that the usage of dedicated security hardware such as Trusted Platform Modules (TPM) will further decrease the overhead in power consumption and delay. TPMs and other security related hardware concepts will be discussed in Section Device Security.

Application Layer

Application layer protocols are an essential part when discussing communication in the context of IoT. There is a wide range of protocols that also might be suitable for an industrial context. IoT application layer protocols such as MQTT, CoAP, or DDS typically provide a very low protocol overhead that enables these protocols to efficiently transport the huge amount of data created by IoT devices. To meet the security requirements of industrial IoT applications, these existing protocols need to be adapted. Therefore, existing IoT application layer protocols will be evaluated regarding their built-in security features and the available security extensions for the protocols.

- **MQTT:** MQTT is based on the publish/subscribe principle and uses a client-server architecture (Standard, 2014). Clients publish messages to a specific topic or subscribe to a topic using a so-called broker. The broker is a server which manages the distribution of messages in the network. MQTT Messages are sent using TCP to enable reliable message delivery. Before clients can send and receive messages, they have to connect to the broker. The CONNECT message, sent by the client, contains optional fields for username and password that can be used for authentication. These fields are the only built-in security features MQTT provides. Due to these scarce security mechanisms, OASIS highly recommends the use of TLS (Dierks, 2008) to secure messages from attackers. Unfortunately, TLS suffers from attacks such as BEAST or CRIME (Sarkar & Fitzgerald, 2013). To overcome the issues with TLS and to provide reliable security for MQTT, Singh et al. (Singh, Rajan, Shivraj, & Balamuralidhar, 2015) propose the use of Key/Cipher text Policy-Attribute Based Encryption (KP/CP-ABE) that relies on lightweight Elliptic Curve Cryptography (ECC). ABE using lightweight ECC supports broadcast encryption. Encrypting a broadcast message enables many clients to decrypt a message. ABE are of the types Ciphertext Policy based ABE (CP-ABE) and Key Policy based ABE (KP-ABE). CP-ABE provides private key generation over a set of attributes and uses an access tree to encrypt the data. KP-ABE generates a user's private keys based on an access tree depending on the user's privileges. In KP-ABE the data is encrypted over a set of attributes. The combination of CP/KP-ABE provides data confidentiality and also provides access control. Another approach to secure MQTT was made by Niruntasukrat et al. (Niruntasukrat, et al., 2016) who proposed an authorization mechanism based on the OAuth 1.0 authentication algorithm. With this algorithm, a device can generate an access token to be allowed to subscribe to a specific resource. This authorization only approach is designed for highly constraint devices which cannot carry TLS or perform cryptographic functions.

- **CoAP:** CoAP relies on the request/response principle between endpoints using a client/server architecture (Shelby, Hartke, & Bormann, 2014). Clients can request specific resources using URIs with HTTP media types such as GET. Requests and responses are sent using UDP to keep the protocols footprint small. Although CoAP uses UDP for transport it also offers modes for guaranteed message delivery. CoAP itself does not provide any security features. To secure the sent messages anyway, the RFC 7252 requires the implementation of DTLS (Modadugu & Rescorla, 2004) but allows the NoSec mode to be used where DTLS can be disabled. DTLS, similar to TLS, results in a huge overhead compared to CoAP's overhead. Therefore, Raza et al. (2013) propose Lithe, a lightweight DTLS integration for CoAP. The integration is, among others, done by using the principle of 6LoWPAN header compression mechanisms (Hui & Thubert, 2011). The header compression for DTLS reduces the overhead for the complete handshake headers by about 33%. There are also other approaches such as proposed by Capossele et al. (2015) or Ukil et al. (2014) that try to secure CoAP by manipulating DTLS to reduce the packet overhead and number of messages.
- **XMPP:** XMPP is based on a client/server architecture (Saint-Andre, Smith, Tronçon, & Troncon, 2009) and uses XML to structure the data sent between clients and servers. All clients in a specific domain are connected to one server. Servers can connect to other servers to enable inter domain communication. The communication between client and server can be secured using TLS; the communication between servers however does not necessarily need to be secured. Therefore, the RFC 6120 (Saint-Andre, 2011) recommends end-to-end encryption between clients in different networks to provide data security. One approach to secure XMPP was done by Celesti et al. (2013), who proposed SE Clever. SE Clever is the secure extension of an existing middleware for cloud computing. The security extensions enable XMPP to (i) sign the sent XML files with private key of the sender, (ii) attach content encrypted with receiver's public key to the message body, (iii) attach a session key for symmetric encryption, and (iv) attach signed timestamps. These extensions enable a secure XMPP middleware without establishing TLS connections.
- **DDS:** DDS is a protocol for real-time, high-performance data exchange between clients (Pardo-Castellote, 2003) that relies on the publish/subscribe principle but does not require a broker to distribute messages. DDS clients simply publish data to topics and other clients subscribe to the topics. DDS' architecture is similar to the one of bus systems where every client is connected to the bus. Data is transported using TCP, UDP or any other transport specification. DDS does not provide any security features; therefore, TLS or DTLS should be used to protect the data from manipulation or theft.
- **AMQP:** AMQP is a message-oriented protocol based on publish/subscribe and point-to-point communication (Vinoski, 2006). AMQP uses a broker to distribute messages. The broker provides an exchange service and a message queue service. The exchange service is used to send data to a specific receiver where the data is stored in a queue the receiver can read from. The exchange service uses point-to-point communication with the broker as a forwarding device. The message queue service copies the same message to each client that has subscribed to the message topic. The message queue service uses the publish/subscribe principle for data distribution; the messages are sent using TCP but AMQP can be extended to also use UDP. AMQP does not provide any security features; therefore, Vinoski (2006) recommends the use of TLS to provide data security. Besides TLS no other security extensions for AMQP were proposed yet.
- **OPC-UA:** OPC-UA is based on a client/server architecture using the request/response principle (Mahnke, Leitner, & Damm, 2009). Each client needs an OPC-UA client implementation that

uses the OPC-UA communication stack to create request messages. The client's communication stack communicates with the server's communication stack by sending the request messages. The server's communication stack forwards the request to the server implementation. The server implementation provides the response which is sent to the client by using the server's and client's communication stacks. Furthermore, subscriptions and notifications can be sent between client and server using a publish/subscribe principle (Cavalieri, Cutuli, & Monteleone, 2010). OPC-UA provides two different communication modes for message exchange. The first mode, UA Web Services, uses web services secured with HTTPS to communicate. The second mode is named UA Native and sends data in plain text using TCP. Besides HTTPS, OPC-UA provides a huge amount of built-in security features. The security features include:

- **Session Encryption:** Transmitted messages are encrypted with 128 bit or 256 bit keys.
- **Message Signing:** Messages are signed to prevent data manipulation.
- **Sequenced Packets:** Sequencing eliminates the possibility of replay attacks.
- **Authentication:** OpenSSL certificates are used to authenticate systems or applications.
- **User Control:** Login credentials must be provided by users to access applications.

Because OPC-UA already provides extended security features, no proposed security extensions exist for this protocol. Due to these security features, OPC-UA generates a huge overhead compared to other protocols.

DEVICE SECURITY

When connecting CPSs to the Internet, securing the device itself is as important as securing the communication between devices. Communicating over an unsecured channel might threaten the confidentiality and integrity of transferred data. Leaving weaknesses at a device itself, however, might lead to bigger issues such as the device being overtaken. Such an overtaken device could then forward confidential data to adversaries' servers, use the device in botnet related attacks or even manipulate the device's intended

Table 4. Security analysis of existing IoT application protocols

	Built-In Security	Extended Security	Provides
MQTT	User/Password Authentication	TLS, KP/CP-ABE (Singh, Rajan, Shivraj, & Balamuralidhar, 2015), Authorization (Niruntasukrat, et al., 2016)	Confidentiality, Integrity
CoAP	None	DTLS, Lithe (Raza, Shafagh, Hewage, Hummen, & Voigt, 2013)	Confidentiality, Integrity
XMPP	None	TLS/SASL, SE Clever (Celesti, Fazio, & Villari, 2013)	Confidentiality, Integrity
DDS	None	TLS/DTLS	Confidentiality, Integrity
AMQP	None	TLS	Confidentiality, Integrity
OPC-UA	Sequencing, Encryption, Authentication, Signing, User Control	None	Confidentiality, Integrity

behavior. The security of a whole network is threatened if an adversary possesses a single device belonging to it. The adversary might apply any type of physical attack to reveal confidential data or even keys stored on the device under attack. These keys could then be used to connect malicious devices to the network without anyone noticing. To counteract all kinds of attacks at the device level, so-called tamper resistance needs to be achieved in software as well as in hardware. A system's tamper resistance can be split into four different steps (Ravi, Raghunathan, & Chakradhar, 2004):

- **Attack Prevention:** Attack Prevention techniques should complicate attacks that target CPSs and thus make the attacks infeasible. Possible techniques include packaging, special hardware design, and software design.
- **Attack Detection:** Attack Detection should detect potential attacks as soon as possible to minimize the effect of them. Possible techniques include, for example, a run-time detection of malicious memory accesses.
- **Attack Recovery:** Attack Recovery is essential in the case of a detected attack to take appropriate countermeasures and to check that the system returns to a normal operation state. Possible techniques include, for example, locking the system or rebooting the system.
- **Tamper Evidence:** Tamper Evidence is responsible for keeping track of past attacks that can be used for inspection later. Tamper evidence be protected from being reversed. Thus, techniques such as seals or wires that have to be cut can be used.

Software

Tamper resistance is a security feature that often is associated with hardware components. However, also software measures can and need to be taken to provide tamper resistance of executed code (Lie, et al., 2000). Horne et al. (Horne, Matheson, Sheehan, & Tarjan, 2001) propose a self-checking code mechanism that can be integrated into existing code segments to provide tamper resistance. Aucsmith et al. (1996) present an approach for tamper resistant software that uses so-called Integrity Verification Kernels to check if software is operating as intended. Integrity verification kernels are self-modifying, self-decrypting, self-checking and installation unique code segments that communicate with other kernels to create an interlocking trust model. Software tamper resistance is able to mitigate physical and side-channel attacks that passively inspect a device and try to reveal data from information such as timings. Thus, these approaches are able to provide data confidentiality. The authors also list design principles for tamper resistant software (Aucsmith, 1996):

- **Secret Dispersion:** Secret Dispersion is used to evenly spread confidential information throughout the whole system. For instance, if a key is distributed in the whole memory instead of being stored in a single location, an attacker is hindered from revealing the whole secret by randomly guessing and observing the correct position in memory.
- **Obfuscating and Interleaving:** This principle converts a program into a state that is harder to understand for humans without changing the functionality of the obfuscated code. Obfuscated code is used to hide its logic and purpose to prevent tampering and reverse engineering.
- **Installation of Unique Code:** Installation of unique code is used to mitigate class attacks (Ouyang, Le, Liu, Ford, & Makedon, 2008) by checking that each code has a unique component. Uniqueness can be added to software by different unique code sequences or encryption keys.

- **Interlocking Trust:** This is the principle of code components relying on other code segments to effectively perform their tasks. Not only are code segments responsible for their own functionality, but also for maintaining and verifying the integrity of other components. Thus, each software component is monitored by another component of the system which forms an interlocking trust relationship between components.

Although software tamper resistance can increase a system's security, it has two major drawbacks compared to tamper resistant hardware. First, most CPSs are constrained in their processing capabilities which limits the feasibility of adding security features in software. Second, software tamper resistance has been shown to be prone to many attacks (Oorschot, Somayaji, & Wurster, 2005), (Wurster, van Oorschot and Paul, & Somayaji, 2005). Therefore, software tamper resistance cannot be relied on to provide a device's security without other security measures.

IDSs are another measure to increase CPS security by potentially detecting viruses, worms, DoS attacks, network attacks, or password attacks. Thus, increasing the confidentiality, integrity, and availability of CPSs. Mitchell and Chen (2014) state the importance of IDSs for CPSs as an unnoticed adversary could set up an attack that is more harmful than attacks that are immediately recognized. The authors further categorize CPS IDSs by their detection technique and the used audit material. The detection technique defines how such IDSs need to be trained and how misbehaving code is detected.

- **Knowledge Based Approach:** These approaches identify runtime features based on specific patterns of misbehavior (Whitman & Mattord, 2011). Because knowledge based approaches only react to known bad code segments, the false positive rate of such approaches is usually low.
- **Behavior Based Approach:** These IDSs approaches identify runtime features that differ from the ordinary (Whitman & Mattord, 2011). Depending on what is defined as ordinary, these IDSs need to be trained live or on supervised data. The advantage of such approaches is that they do not need to previously see the exact code they need to detect. However, the machine learning aspect increases the false positive rate.

In the context of CPSs, there are two possible ways to collect data for analysis.

- **Host Based IDS:** Host based IDSs analyze logs recorded on a single node. The advantage of host based approaches is their independence of other nodes and the corresponding ease of detecting host-level misbehavior (Mitchell & Chen, 2014).
- **Network Based IDS:** These approaches analyze network activity to find compromised nodes (Kasinathan, Pastrone, Spirito, & Vinkovits, 2013). The advantage of this approach is that other, dedicated, and non-compromised nodes are used to identify misbehaving nodes in a network. Dedicated nodes could be equipped with external power sources and more computational power (Wallgren, Raza, & Voigt, 2013).

However, in the context of CPSs, also other indicators such as the physical process itself could be used for intrusion detection. Cardenas et al. (2009) state that traditional IDSs only analyze device or network logs while control systems could be used to monitor the physical process. Anomalies in the physical process could be an indicator for an ongoing attack that might not be detected by traditional IDSs.

Hardware

Secure hardware components need to provide a number of security properties in order to increase the overall security of a system (Vasudevan, Owusu, Zhou, Newsome, & McCune, 2012). The properties considered most important are the following three:

- **Isolated Execution:** A fundamental concept in hardware security is the so-called security by isolation concept (Vasudevan, Owusu, Zhou, Newsome, & McCune, 2012). In this concept, an execution environment is split into two worlds, the normal world and the secure world. The normal world is then used as general-purpose execution environments (GPEE) while the secure world servers as a secure execution environment (SEE). The security by isolation principle can be realized using different hardware elements (Anderson, Bond, Clulow, & Skorobogatov, 2006), on a single CPU (ARM TrustZone (Winter, 2008), Intel Trusted Execution Technology (TXT), AMD SVM), or in software (Madnick & Donovan, 1973). Isolated execution allows software developers to run certain parts of their software in complete isolation from other code that is executed at the same device. Current operating systems (OS) provide isolation at a process level. Security by isolation helps to mitigate the impact of viruses and worms as well as exploits. Also, passive physical and side-channel attacks can be mitigated and thus, confidentiality, and integrity of CPSs is increased. The drawback with this approach is that, if the OS itself is compromised, also the isolation mechanisms are circumvented. Also, Bond and Anderson (2001) highlight that secured execution environments can be targeted by so-called API attacks. The simplest form of such an attack is to issue valid API commands in an unexpected sequence. To account for this type of attack, measures such as security analysis (for example Common Criteria Certification (Mellado, Fernandez-Medina, & Piattini, 2007)) needs to be conducted.
- **Secured Storage:** The need to store confidential data such as key material on a CPS highlights the importance of secured storage. A secured storage therefore should be capable of guaranteeing data integrity and secrecy for any kind of data. Storage secured by software measures is considered to be insecure, as any physical attack can be applied to storage media that is extracted from its coating (Vasudevan, Owusu, Zhou, Newsome, & McCune, 2012). A (now already outdated but simple) possible approach to mitigate physical attacks is to seal the storage by embedding it inside a protective coating that makes the hardware resistant against invasive attacks (Tuyls, et al., 2006). Such protective coatings enable read-proof hardware by being sprayed on traditional hardware. The coating is doped with several random dielectric particles that help to (i) absorb light and UV-light, (ii) make the coating very hard, (iii) provide a certain capacitance of the coating that can be measured by sensors inside of it. These properties not only mitigate physical attacks but also help to identify an ongoing attack by sensing the coating's capacity.
- **Trusted Path:** To provide confidentiality, authenticity and availability for a connection between software and a peripheral such as a sensor, a trusted path needs to be used (Zhou, Gligor, Newsome, & McCune, 2012). Trusted Path are essential to mitigate the problem of malicious applications that try to manipulate data such that a CPS or the associated physical process could be damaged.

Besides these three mentioned properties, Vasudevan et al. (2012) list two additional important properties. Remote Attestation is used to verify the origin of messages from software modules, for example, a remote server could verify the correctness of a client's OS kernel and application. Remote attestation therefore provides data integrity. Secure Provisioning allows data to be sent to a specific software part running on a specific hardware module. For example, data could only be sent to services that were previously verified using remote attestation. Secure provisioning therefore also provides data integrity. Stankovic (2014) notes that in order to meet the security requirements defined for CPSs, hardware support is needed in addition to software mechanisms. He further states that so-called tamper resistant hardware modules will be essential in providing encryption, authentication, attestation, and secured storage.

- **Security Co-Processors:** These are one example of such tamper resistant hardware components (Smith & Weingart, 1999). The security principle used by security co-processors to increase security is isolated execution. Security co-processors are used as trusted devices that execute critical software parts in a tamper resistant environment. The software components that are most frequently executed on security co-processors are cryptographic algorithms such as encryption, decryption, signing and verification (Mclvor, McLoone, & McCanny, 2003). The execution of cryptographic algorithms is especially vulnerable to physical attacks as so-called side-channel attacks can be used to reveal key material or other confidential data (Standaert, Malkin, & Yung, 2009), (Mangard, Oswald, & Popp, 2008). Because side-channel weaknesses might make other security measures such as secured storage useless, the focus in cryptographic co-processor design is often in eliminating all side-channels (Tiri, et al., 2005). In addition to cryptographic operations, there are also other use-case scenarios for security co-processors such as intrusion detection. Zhang et al. (2002) propose to run IDS software on a tamper resistant co-processor instead of a host processor for increased security. This approach has four advantages according to the authors: (i) the intrusion detection is independent from other software components, (ii) the interface between the security- and host processor is very simple, so it is hard to exploit, (iii) the security co-processor can boot the device into a well-known state, (iv) statements made by the software running on a security co-processor can be fully trusted. Security co-processors will be especially useful in the context of cyber-physical systems (Feller, 2014) where, for instance, controller software could be executed in a secure manner. If CPSs are used in industrial processes, many new scenarios such as smart maintenance (Lesjak, et al., 2015) need to be considered for which security co-processors provide confidentiality, integrity, and availability.
- **Trusted Platform Modules (TPM):** TPM are standardized hardware components often associated with personal computers because of their size and power requirements. TPMs typically comprise of several components such as a cryptographic co-processor and secured storage. The CIA attributes provided by a TPM are therefore a combination of the attributes provided by these components. TPMs are capable of providing confidentiality, integrity, and availability for CPSs. Because of the size requirements, CPSs often emulate a TPM's functionality in software (Aaraj, Raghunathan, & Jha, 2008), (Strasser & Stamer, 2008), which poses security risks as well as problems regarding the power consumption of CPU-intensive cryptographic operations. TPMs are decreasing in size, so they nowadays are also included into CPSs (Kinney, 2006) and even smartcards (Akram, Markantonakis, & Mayes, 2014).
- TPM can be used to increase security in CPSs in various other ways too. Hutter and Toegl (2010) present a TPM that is extended by NFC functionality to provide a trusted channel between two devices. The TPM chip is further used for remote attestation that provides trust that the device is

not modified in a malicious way. Kothmayr et al. introduce a two-way authentication (Kothmayr, Schmitt, Hu, Brünig, & Carle, 2013) and end-to-end encryption (Kothmayr, Schmitt, Hu, Brünig, & Carle, 2012) that relies on TPMs in both devices to generate and store RSA keys, and to perform cryptographic operations. Because many IoT devices are resource constraint, the authors argue that TPMs not only need to be included for tamper resistance but also to handle the overhead imposed by using cryptographic security measures. According to Hu et al. (Hu, Tan, Corke, Shih, & Jha, 2010), including TPM into CPSs increases the system's overall price by an average of only 5 percent.

Another possible security feature when using TPM is the so-called authenticated boot as specified by the Trusted Computing Group (TCG). Authenticated boot is a passive method that stores integrity measures such as hashes of software components on the TPM. When booting a device, the integrity measure is applied again and compared against the stored value before loading and executing the software. This security mechanism, however, can only be used to protect a software's integrity at boot time; malicious code that is loaded at run time cannot be detected by such a TPM assisted system. A simple solution to that problem would be to reboot a potentially compromised system to restore a secured system state (Hendricks & Van Doorn, 2004). Raciti and Nadjm-Tehrani (2012) address the problem of many CPSs such as smart meters: the unsecured connection between sensor and controller. They argue that although TPM are included in many solutions nowadays, vulnerabilities persist that still allow CPSs to be attacked. To mitigate some problems, Raciti and Nadjm-Tehrani suggest to include an anomaly detection system in addition to a TPM chip in order to detect potential attacks targeting the communication between sensor and controller.

Tamper resistant hardware is shown to increase security by mitigating various types of physical and also logical attacks. However, as prices for such hardware devices are decreasing, also low cost attacks targeting tamper resistant hardware are possible (Anderson & Kuhn, 1997), (Bao et al., 1997). Anderson and Kuhn (Anderson & Kuhn, Tamper Resistance - a Cautionary Note, 1996) state that trusting a system because of its tamper resistant components is problematic as such systems are broken frequently.

CONCLUSION

The number of IoT devices is rapidly rising and forecasted to reach 50 billion devices by 2020 (Evans, 2011). Initiatives such as Industry 4.0 and Smart Manufacturing will further boost this trend, as they envision connecting production machinery to the Internet. These so-called cyber-physical production systems are attractive targets for adversaries for a number of reasons.

- The number of connected devices is still rapidly increasing while most devices are online 24/7.
- A large number of currently connected devices has no proper security mechanisms implemented or is using default credentials.
- Most of the CPSs are resource constraint which does not allow to implemented traditional security measures.
- Many CPSs process confidential data. Attacks can therefore be used for industrial espionage.
- Attacks might aim at damaging the physical process which could threaten human lives.

Trends in emerging CPS threats (Marinos, Belmonte, & Rekleitis, 2015) show that the number of all top 10 attacks such as DoS attacks, cyber espionage and physically damaging attacks are increasing compared to last year's report. This further highlights the importance of CPS security.

Due to these reasons, an overview of CPS security is given in this work. To be able to categorize attacks as well as the applied countermeasures, the authors have given attack taxonomies for logical as well as for physical attacks. The authors also have shown recent major attacks that increased the public attention regarding IoT security. After that, security measures are discussed for two major aspects of CPSs: on a network level and on the device level. On the network level, all TCP/IP layers and their protocols have been evaluated regarding potential security measures. On device level, software measures and potential security increasing hardware components have been presented. Simply combining some of the presented security measures however might harm a system more than it improves its security (Krawczyk, 2001). Also, a tradeoff between security and other parameters such as overhead needs to be made. Therefore, this publication tends to present an overview of current security related topics rather than suggest to apply certain solutions.

REFERENCES

Aaraj, N., Raghunathan, A., & Jha, N. K. (2008). Analysis and Design of a Hardware/Software Trusted Platform Module for Embedded Systems. *ACM Transactions on Embedded Computing Systems*, 8.

Adomavicius, G., & Tuzhilin, A. (2005). Personalization Technologies: A Process-Oriented Perspective. *Communications of the ACM*, *48*(10), 83–90. doi:10.1145/1089107.1089109

Akram, R. N., Markantonakis, K., & Mayes, K. (2014). Trusted Platform Module for Smart Cards. In *Proceedings of the 2014 6th International Conference on New Technologies, Mobility and Security (NTMS)* (pp. 1-5).

Anderson, R., Bond, M., Clulow, J., & Skorobogatov, S. (2006). Cryptographic Processors - A Survey. *Proceedings of the IEEE*, *94*(2), 357–369. doi:10.1109/JPROC.2005.862423

Anderson, R., & Kuhn, M. (1996). Tamper Resistance - a Cautionary Note. In *Proceedings of the second Usenix workshop on electronic commerce*, 2, pp. 1-11.

Anderson, R., & Kuhn, M. (1997). Low Cost Attacks on Tamper Resistant Devices. In *Proceedings of the International Workshop on Security Protocols* (pp. 125-136).

Aucsmith, D. (1996). Tamper Resistant Software: An Implementation. In *Proceedings of the International Workshop on Information Hiding*, (pp. 317-333).

Bao, F., Deng, R. H., Han, Y., Jeng, A., Narasimhalu, A. D., & Ngair, T. (1997). Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults. In *Proceedings of the International Workshop on Security Protocols* (pp. 115-124).

Baronti, P., Pillai, P., Chook, V. W., Chessa, S., Gotta, A., & Hu, Y. F. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards. *Computer Communications*, *30*(7), 1655–1695. doi:10.1016/j.comcom.2006.12.020

Beggs, C. (2006). Proposed Risk Minimization Measures for Cyber-Terrorism and SCADA Networks in Australia. In *Proceedings of the 5th European conference on information warfare and security (ECIW 2006, Helsinki). Academic Publishing, Reading, UK*, (pp. 9-18).

Bond, M., & Anderson, R. (2001). API-Level Attacks on Embedded Systems. *Computer*, *34*(10), 67–75. doi:10.1109/2.955101

Brachmann, M., Keoh, S. L., Morchon, O. G., & Kumar, S. S. (2012, July). End-to-End Transport Security in the IP-Based Internet of Things. In *Proceedings of the 2012 21st International Conference on Computer Communications and Networks (ICCCN)* (pp. 1-5). IEEE.

Capossele, A., Cervo, V., De Cicco, G., & Petrioli, C. (2015). Security as a CoAP resource: an optimized DTLS implementation for the IoT. In *Proceedings of the 2015 IEEE International Conference on Communications (ICC)* (pp. 549-554). doi:10.1109/ICC.2015.7248379

Cardenas, A. A., Amin, S., & Sastry, S. (2008). Secure Control: Towards Survivable Cyber-Physical Systems. In *Proceedings of the 28th International Conference on*, *Distributed Computing Systems Workshops ICDCS'08* (pp. 495-500).

Cardenas, A. A., Amin, S., Sinopoli, B., Giani, A., Perrig, A., & Sastry, S. (2009). Challenges for Securing Cyber Physical Systems. In *Proceedings of the Workshop on future directions in cyber-physical systems security*, (p. 5).

Cavalieri, S., Cutuli, G., & Monteleone, S. (2010, May). Evaluating Impact of Security on OPC UA Performance. In *Proceedings of the 3rd International Conference on Human System Interaction*, (pp. 687-694). doi:10.1109/HSI.2010.5514495

Celesti, A., Fazio, M., & Villari, M. (2013). SE CLEVER: A secure message oriented Middleware for Cloud federation. In *Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC)*, (pp. 35-40). doi:10.1109/ISCC.2013.6754919

Chaojun, G., Jirutitijaroen, P., & Motani, M. (2015). Detecting False Data Injection Attacks in AC State Estimation. *IEEE Transactions on Smart Grid*, *6*(5), 2476–2483. doi:10.1109/TSG.2015.2388545

Cheng, P., Zhang, H., & Chen, J. (2016). *Cyber Security for Industrial Control Systems: From the Viewpoint of Close-Loop*. CRC Press. doi:10.1201/b19629

Cui, A., & Stolfo, S. J. (2010). A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan. In *Proceedings of the 26th Annual Computer Security Applications Conference* (pp. 97-106). doi:10.1145/1920261.1920276

Dadras, S., Gerdes, R. M., & Sharma, R. (2015). Vehicular Platooning in an Adversarial Environment. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security* (pp. 167-178).

Dagon, D., Gu, G., Lee, C. P., & Lee, W. (2007). A Taxonomy of Botnet Structures. In *Proceedings of the Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual* (pp. 325-339).

Daniela, T. (2011). Communication Security in SCADA Pipeline Monitoring Systems. In *Proceedings of the 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research* (pp. 1-5).

Daws, R. (2016, 11). Akamai: IoT botnet set a record in a year when DDoS attacks increased 71 percent. IoT Tech News.

Denning, D. E. (2000). Cyberterrorism: Testimony Before the Special Oversight Panel on Terrorism Committee on Armed Services US House of Representatives. *Focus on Terrorism, 9*.

Denning, D. E. (2000). Cyberterrorism: The Logic Bomb versus the Truck Bomb. *Global Dialogue*, 2.

Dierks, T. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2*. IETF. doi:10.17487/rfc5246

Evans, D. (2011). The Internet of Things. *How the Next Evolution of the Internet is Changing Everything* (Whitepaper). *Cisco Internet Business Solutions Group*, *1*, 1–12.

Feller, T. (2014). Towards Trustworthy Cyber-Physical Systems. In *Trustworthy Reconfigurable Systems* (pp. 85–136). Springer.

Gallagher, S. (2016, 10). Double-dip Internet-of-Things botnet attack felt across the Internet. *Ars Technica*.

Garcia-Morchon, O., Kumar, S., Struik, R., Keoh, S., & Hummen, R. (2013). *Security Considerations in the IP-based Internet of Things*. IETF.

Gibson, I., Rosen, D. W., & Stucker, B. et al. (2010). *Additive Manufacturing Technologies* (Vol. 238). Springer. doi:10.1007/978-1-4419-1120-9

Hansman, S., & Hunt, R. (2005). A taxonomy of network and computer attacks. *Computers & Security, 24*, 31-43.

Haselsteiner, E., & Breitfuß, K. (2006). Security in Near Field Communication (NFC). In *Proceedings of the Workshop on RFID security.*

Hendricks, J., & Van Doorn, L. (2004). Secure Bootstrap Is Not Enough: Shoring up the Trusted Computing Base. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. doi:10.1145/1133572.1133600

Hentunen, D., & Tikkanen, A. (2014). *Havex Hunts For ICS/SCADA Systems*. F-Secure.

Höller, A., Druml, N., Kreiner, C., Steger, C., & Felicijan, T. (2014). Hardware/Software Co-Design of Elliptic-Curve Cryptography for Resource-Constraint Applications. In *Proceedings of the 51st Annual Design Automation Conference*. ACM. doi:10.1145/2593069.2593148

Horne, B., Matheson, L., Sheehan, C., & Tarjan, R. E. (2001). Dynamic Self-Checking Techniques for Improved Tamper Resistance. In *Proceedings of the ACM Workshop on Digital Rights Management*, (pp. 141-159).

Hu, W., Tan, H., Corke, P., Shih, W. C., & Jha, S. (2010). Toward Trusted Wireless Sensor Networks. *ACM Transactions on Sensor Networks*, 7(1).

Huang, Y., Li, H., Campbell, K. A., & Han, Z. (2011). Defending False Data Injection Attack on Smart Grid Network Using Adaptive CUSUM Test. In *Proceedings of the 2011 45th Annual Conference on Information Sciences and Systems (CISS)* (pp. 1-6).

Hui, J., & Thubert, P. (2011). *Compression format for IPv6 datagrams over IEEE 802.15. 4-based networks*. IETF.

Hurley, C. (2003). *The worldwide wardrive: The myths, the misconceptions, the truth, the future*. Defcon.

Hutter, M., & Toegl, R. (2010). A Trusted Platform Module for Near Field Communication. In *Proceedings of the 2010 Fifth International Conference on Systems and Networks Communications* (pp. 136-141). doi:10.1109/ICSNC.2010.27

ICS-CERT. (2016). *Cyber-Attack Against Ukrainian Critical Infrastructure*.

Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: Perspectives and challenges. *Wireless Networks*, *20*(8), 2481–2501. doi:10.1007/s11276-014-0761-7

Karlof, C., Sastry, N., & Wagner, D. (2004). TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (pp. 162-175). doi:10.1145/1031495.1031515

Karlof, C., & Wagner, D. (2003). Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks*, *1*(2-3), 293–315. doi:10.1016/S1570-8705(03)00008-8

Kasinathan, P., Pastrone, C., Spirito, M. A., & Vinkovits, M. (2013). *Denial-of-Service detection in 6LoWPAN based Internet of Things. In WiMob* (pp. 600–607). doi:10.1109/WiMOB.2013.6673419

Kim, T. T., & Poor, H. V. (2011). Strategic Protection Against Data Injection Attacks on Power Grids. *IEEE Transactions on Smart Grid*, *2*(2), 326–333. doi:10.1109/TSG.2011.2119336

Kinney, S. L. (2006). *Trusted Platform Module Basics: Using TPM in Embedded Systems*. Newnes.

Kocher, P., Lee, R., McGraw, G., Raghunathan, A., & Moderator-Ravi, S. (2004). Security as a New Dimension in Embedded System Design. In *Proceedings of the 41st annual Design Automation Conference*,(pp. 753-760).

Kothmayr, T., Schmitt, C., Hu, W., Brünig, M., & Carle, G. (2012). A DTLS Based End-To-End Security Architecture for the Internet of Things with Two-Way Authentication. In *Proceedings of the 2012 IEEE 37th Conference on*, *Local Computer Networks Workshops (LCN Workshops)* (pp. 956-963).

Kothmayr, T., Schmitt, C., Hu, W., Brünig, M., & Carle, G. (2013). DTLS based security and two-way authentication for the Internet of Things. *Ad Hoc Networks*, *11*(8), 2710–2723. doi:10.1016/j.adhoc.2013.05.003

Krawczyk, H. (2001). The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In *Proceedings of the Annual International Cryptology Conference* (pp. 310-331). doi:10.1007/3-540-44647-8_19

Kwon, C., Liu, W., & Hwang, I. (2014). Analysis and Design of Stealthy Cyber Attacks on Unmanned Aerial Systems. *Journal of Aerospace Information Systems*, *11*(8), 525–539. doi:10.2514/1.I010201

Langner, R. (2011). Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security \& Privacy, 9*, 49-51.

Le, T.-H., Canovas, C., & Clédiere, J. (2008). An Overview of Side Channel Analysis Attacks. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security* (pp. 33-43). doi:10.1145/1368310.1368319

Le Malécot, E., & Inoue, D. (2014). The Carna Botnet Through the Lens of a Network Telescope. In *Foundations and Practice of Security* (pp. 426–441). Springer. doi:10.1007/978-3-319-05302-8_26

Lee, D. (2012, May). Flame: Massive cyber-attack discovered, researchers say. *BBC News*.

Lee, E. A. (2008). Cyber Physical Systems: Design Challenges. In *Proceedings of the 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)* (pp. 363-369).

Lee, J.-S., Jeong, H., Park, J.-H., Kim, M., & Noh, B.-N. (2008). The Activity Analysis of Malicious HTTP-Based Botnets Using Degree of Periodic Repeatability. In *Proceedings of the International Conference on Security Technology SECTECH'08* (pp. 83-86).

Lennvall, T., Svensson, S., & Hekland, F. (2008). A Comparison of WirelessHART and ZigBee for Industrial Applications. In *Proceedings of the IEEE International Workshop on Factory Communication Systems* (pp. 85-88). doi:10.1109/WFCS.2008.4638746

Lesjak, C., Hein, D., Hofmann, M., Maritsch, M., Aldrian, A., Priller, P., . . . Pregartner, G. (2015). Securing Smart Maintenance Services: Hardware-Security and TLS for MQTT. In *Proceedings of the 2015 IEEE 13th International Conference on Industrial Informatics (INDIN)* (pp. 1243-1250).

Lie, D., Thekkath, C., Mitchell, M., Lincoln, P., Boneh, D., Mitchell, J., & Horowitz, M. (2000). Architectural Support for Copy and Tamper Resistant Software. *ACM SIGPLAN Notices*, *35*(11), 168–177. doi:10.1145/356989.357005

Lo, C.-H., & Ansari, N. (2013). CONSUMER: A Novel Hybrid Intrusion Detection System for Distribution Networks in Smart Grid. *IEEE Transactions on Emerging Topics in Computing*, *1*(1), 33–44. doi:10.1109/TETC.2013.2274043

Lun, Y., D'Innocenzo, A., Malavolta, I., & Di Benedetto, M. (2016). Cyber-Physical Systems Security: a Systematic Mapping Study.

Madnick, S. E., & Donovan, J. J. (1973). Application and Analysis of the Virtual Machine Approach to Information System Security and Isolation. In *Proceedings of the workshop on virtual computer systems* (pp. 210-224). doi:10.1145/800122.803961

Mahnke, W., Leitner, S.-H., & Damm, M. (2009). *OPC Unified Architecture*. Springer Science & Business Media. doi:10.1007/978-3-540-68899-0

Mangard, S., Oswald, E., & Popp, T. (2008). *Power Analysis Attacks: Revealing the Secrets of Smart Cards* (Vol. 31). Springer Science & Business Media.

Marinos, L., Belmonte, A., & Rekleitis, E. (2015). *Enisa Threat Landscape (Technical report)*. ENISA.

Mattern, F., & Floerkemeier, C. (2010). From the Internet of Computers to the Internet of Things. In *From active data management to event-based systems and more* (pp. 242–259). Springer. doi:10.1007/978-3-642-17226-7_15

Mclvor, C., McLoone, M., & McCanny, J. V. (2003). Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures. In *Conference Record of the Thirty-Seventh Asilomar Conference on* Signals, Systems and Computers (Vol. 1, pp. 379-384).

Mellado, D., Fernandez-Medina, E., & Piattini, M. (2007). A common criteria based security requirements engineering process. *Computer Standards & Interfaces*, *29*(2), 244–253. doi:10.1016/j.csi.2006.04.002

Miller, B., & Rowe, D. (2012). A Survey of SCADA and Critical Infrastructure Incidents. In *Proceedings of the 1st Annual conference on Research in information technology* (pp. 51-56). doi:10.1145/2380790.2380805

Mishra, S., Shoukry, Y., Karamchandani, N., Diggavi, S., & Tabuada, P. (2015). Secure State Estimation: Optimal Guarantees Against Sensor Attacks in the Presence of Noise. In *Proceedings of the 2015 IEEE International Symposium on Information Theory (ISIT)* (pp. 2929-2933).

Mitchell, R., & Chen, I.-R. (2014). A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. [CSUR]. *ACM Computing Surveys*, *46*(4), 55. doi:10.1145/2542049

Mo, Y., Weerakkody, S., & Sinopoli, B. (2015). Physical Authentication of Control Systems: Designing Watermarked Control Inputs to Detect Counterfeit Sensor Outputs. *IEEE Control Systems*, *35*(1), 93–109. doi:10.1109/MCS.2014.2364724

Modadugu, N., & Rescorla, E. (2004). *The Design and Implementation of Datagram TLS*. NDSS.

Nicholson, A., Webber, S., Dyer, S., Patel, T., & Janicke, H. (2012). SCADA security in the light of Cyber-Warfare. *Computers & Security, 31*, 418-436.

Niruntasukrat, A., Issariyapat, C., Pongpaibool, P., Meesublak, K., Aiumsupucgul, P., & Panya, A. (2016). Authorization Mechanism for MQTT-based Internet of Things. In *Proceedings of the 2016 IEEE International Conference on Communications Workshops (ICC)* (pp. 290-295).

Oorschot, V., Somayaji, A., & Wurster, G. (2005). Hardware-Assisted Circumvention of Self-Hashing Software Tamper Resistance. *IEEE Transactions on Dependable and Secure Computing*, *2*(2), 82–92. doi:10.1109/TDSC.2005.24

Oppliger, R. (1998). Security at the Internet Layer. *Computer*, *31*(9), 43–47. doi:10.1109/2.708449

Ouyang, Y., Le, Z., Liu, D., Ford, J., & Makedon, F. (2008). Source Location Privacy against Laptop-Class Attacks in Sensor Networks. In *Proceedings of the 4th international conference on Security and privacy in communication networks* (p. 5). doi:10.1145/1460877.1460884

Pa, Y. M., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., & Rossow, C. (2015). IoTPOT: Analysing the Rise of IoT Compromises. In *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT 15)*.

Pardo-Castellote, G. (2003). OMG Data-Distribution Service: Architectural Overview. *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops* (pp. 200-206).

Pasqualetti, F., Dörfler, F., & Bullo, F. (2013). Attack Detection and Identification in Cyber-Physical Systems. *IEEE Transactions on Automatic Control*, *58*(11), 2715–2729. doi:10.1109/TAC.2013.2266831

Plósz, S., Farshad, A., Tauber, M., Lesjak, C., Ruprechter, T., & Pereira, N. (2014). Security Vulnerabilities and Risks in Industrial Usage of Wireless Communication. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, (pp. 1-8). doi:10.1109/ETFA.2014.7005129

Raciti, M., & Nadjm-Tehrani, S. (2012). Embedded Cyber-Physical Anomaly Detection in Smart Meters. In *Proceedings of the International Workshop on Critical Information Infrastructures Security*, (pp. 34-45).

Ravi, S., Raghunathan, A., & Chakradhar, S. (2004). Tamper Resistance Mechanisms for Secure Embedded Systems. In *Proceedings of the 17th International Conference on VLSI Design* (pp. 605-611).

Ravi, S., Raghunathan, A., Kocher, P., & Hattangady, S. (2004). Security in Embedded Systems: Design Challenges. *ACM Transactions on Embedded Computing Systems*, *3*(3), 461–491. doi:10.1145/1015047.1015049

Raza, S., Shafagh, H., Hewage, K., Hummen, R., & Voigt, T. (2013). Lithe: Lightweight Secure CoAP for the Internet of Things. *IEEE Sensors Journal*, *13*(10), 3711–3720. doi:10.1109/JSEN.2013.2277656

Referat, B. f. (2013). *Zukunftsbild Industrie 4.0*. Bundesministerium fuer Bildung und Forschung Referat.

Roman, R., Najera, P., & Lopez, J. (2011). Securing the Internet of Things. *Computer*, *44*(9), 51–58. doi:10.1109/MC.2011.291

Saint-Andre, P. (2011). *Extensible Messaging and Presence Protocol (XMPP): Core*. IETF. doi:10.17487/rfc6122

Saint-Andre, P., Smith, K., Tronçon, R., & Troncon, R. (2009). *XMPP: The Definitive Guide*. O'Reilly Media, Inc.

Saltzer, J. H., & Schroeder, M. D. (1975). The Protection of Information in Computer Systems. *Proceedings of the IEEE*, *63*(9), 1278–1308. doi:10.1109/PROC.1975.9939

Sarkar, P. G., & Fitzgerald, S. (2013). Attacks on SSL: A Comprehensive Study of Beast, Crime, Time, Breach, Lucky 13 & RC4 Biases.

Shelby, Z., Hartke, K., & Bormann, C. (2014). *The Constrained Application Protocol (CoAP). Tech. rep*. IETF.

Singh, M., Rajan, M. A., Shivraj, V. L., & Balamuralidhar, P. (2015). Secure MQTT for Internet of Things (IoT). In *Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 746-751).

Smith, S. W., & Weingart, S. (1999). Building a high-performance, programmable secure coprocessor. *Computer Networks*, *31*(8), 831–860. doi:10.1016/S1389-1286(98)00019-X

Somani, G., Gaur, M. S., & Sanghi, D. (2015). DDoS/EDoS attack in Cloud: Affecting everyone out there! In *Proceedings of the 8th International Conference on Security of Information and Networks* (pp. 169-176). doi:10.1145/2799979.2800005

Standaert, F.-X., Malkin, T. G., & Yung, M. (2009). A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 443-461). doi:10.1007/978-3-642-01001-9_26

Standard, O. A. (2014). *MQTT Version 3.1*. OASIS.

Stankovic, J. A. (2014). Research Directions for the Internet of Things. *IEEE Internet of Things Journal*, *1*(1), 3–9. doi:10.1109/JIOT.2014.2312291

Strasser, M., & Stamer, H. (2008). A Software-Based Trusted Platform Module Emulator. In *Proceedings of the International Conference on Trusted Computing* (pp. 33-47).

Tiri, K., Hwang, D., Hodjat, A., Lai, B., Yang, S., Schaumont, P., & Verbauwhede, I. (2005). A Side-Channel Leakage Free Coprocessor IC in 0.18 µm CMOS for Embedded AES-based Cryptographic and Biometric Processing. In *Proceedings of the 42nd Design Automation Conference* (pp. 222-227).

Tuyls, P., Schrijen, G.-J., Škorić, B., Van Geloven, J., Verhaegh, N., & Wolters, R. (2006). Read-Proof Hardware from Protective Coatings. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 369-383).

Ukil, A., Bandyopadhyay, S., Bhattacharyya, A., Pal, A., & Bose, T. (2014). Lightweight security scheme for IoT applications using CoAP. *International Journal of Pervasive Computing and Communications*, *10*(4), 372–392. doi:10.1108/IJPCC-01-2014-0002

Urbina, D. I., Giraldo, J. A., Cardenas, A. A., Tippenhauer, N. O., Valente, J., Faisal, M., & Sandberg, H. et al. (2016). Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1092-1105). doi:10.1145/2976749.2978388

Vasudevan, A., Owusu, E., Zhou, Z., Newsome, J., & McCune, J. M. (2012). Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give Me? In *Proceedings of the International Conference on Trust and Trustworthy Computing* (pp. 159-178). Springer Berlin Heidelberg. doi:10.1007/978-3-642-30921-2_10

Vinoski, S. (2006). Advanced Message Queuing Protocol. *IEEE Internet Computing*, 10(6).

Vuković, O., & Dán, G. (2014). Security of Fully Distributed Power System State Estimation: Detection and Mitigation of Data Integrity Attacks. *IEEE Journal on Selected Areas in Communications*, *32*(7), 1500–1508. doi:10.1109/JSAC.2014.2332106

Wallgren, L., Raza, S., & Voigt, T. (2013). Routing Attacks and Countermeasures in the RPL-based Internet of Things. *International Journal of Distributed Sensor Networks*, *9*(8), 794326. doi:10.1155/2013/794326

Weingart, S. H. (2000). Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses. Advanced Message Queuing Protocol*International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 302-317). doi:10.1007/3-540-44499-8_24

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, *265*(3), 94–104. doi:10.1038/scientificamerican0991-94

Whitman, M. E., & Mattord, H. J. (2011). *Principles of Information Security*. Cengage Learning.

Winter, J. (2008). Trusted Computing Building Blocks for Embedded Linux-based ARM TrustZone Platforms. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing* (pp. 21-30). doi:10.1145/1456455.1456460

Wurster, G., van Oorschot and Paul, C., & Somayaji, A. (2005). A generic attack on checksumming-based software tamper resistance. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S\&P'05)*, (pp. 127-138). doi:10.1109/SP.2005.2

Xue, M., Wang, W., & Roy, S. (2014). Security Concepts for the Dynamics of Autonomous Vehicle Networks. *Automatica*, *50*(3), 852–857. doi:10.1016/j.automatica.2013.12.001

Zhang, X., van Doorn, L., Jaeger, T., Perez, R., & Sailer, R. (2002). Secure Coprocessor-based Intrusion Detection. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop* (pp. 239-242). doi:10.1145/1133373.1133423

Zhou, Z., Gligor, V. D., Newsome, J., & McCune, J. M. (2012). Building Verifiable Trusted Path on Commodity x86 Computers. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (pp. 616-630). doi:10.1109/SP.2012.42

Zhu, Q., & Basar, T. (2015). Game-Theoretic Methods for Robustness, Security, and Resilience of Cyberphysical Control Systems: Games-in-Games Principle for Optimal Cross-Layer Resilient Control Systems. *IEEE Control Systems*, *35*(1), 46–65. doi:10.1109/MCS.2014.2364710

Zorzi, M., Gluhak, A., Lange, S., & Bassi, A. (2010). From Today's Internet of Things to a Future Internet of Things: A Wireless-and Mobility-Related View. *IEEE Wireless Communications*, *17*(6), 44–51. doi:10.1109/MWC.2010.5675777

Zuehlke, D. (2010). SmartFactory - Towards a factory-of-things. *Annual Reviews in Control*, *34*(1), 129–138. doi:10.1016/j.arcontrol.2010.02.008

KEY TERMS AND DEFINITIONS

CIA: Security attributes confidentiality, integrity, and availability that need to be protected. Many attacks target one or multiple of these attributes.

Industrial IoT: Inspired by initiatives such as Industry 4.0 or Smart Manufacturing, IoT concepts are applied to industrial machinery. These machines are connected to the Internet, thus making them accessible from everywhere.

Intrusion Detection System: Used to monitor networks or devices to detect ongoing attacks. In combination with other measures, also successfully defeated attacks should be detected by IDSs.

Logical Attack: Logical attacks can be conducted using existing interfaces to a device, such as network interfaces or a debug interface. This type of attack can be done remotely without physical access to the device.

Network Security: Due to IoT devices and even CPS being connected with other devices and even the Internet, security at the network layer needs to be provided. Security measures can be applied at different network layers such as the transport layer.

Physical Attack: Physical attacks require physical access to the device under attack. This type of attack can be invasive, semi-invasive or non-invasive, which denotes the severity of modifications an attacker performs with the attacked device.

Tamper Resistance: Devices that should not reveal any confidential information need to be tamper resistant. Tamper resistance is achieved mostly through hardware measures but can also be realized in software only.

2018 IEEE International Conference on RFID (RFID)

QSNFC: Quick and Secured Near Field Communication for the Internet of Things

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Sarah Haas, Rainer Matischek
Design Center Graz
Infineon Technologies Austria AG
Graz, Austria
{sarah.haas, rainer.matischek}@infineon.com

***Abstract*—Near Field Communication (NFC) is used for a wide range of security-critical applications such as payment or access control. Although such applications require secured data transfer, the NFC protocol does not include transport layer security. Other protocols that are built on top of NFC, such as the NFC data exchange format (NDEF), only provide insufficient security measures. Therefore, implemented security solutions are often application specific and do not follow well-established standards. To facilitate NFC usage in the Internet of Things (IoT) where millions of devices need to be secured, an efficient and sufficiently secured NFC-based protocol needs to be developed. In this paper, we present the Quick and Secured NFC (QSNFC) protocol. Our protocol is capable of performing more efficient key agreements for recurring connections, and thus, can be used as an efficient alternative to the Transport Layer Security (TLS) protocol.**

***Index Terms*—Near Field Communication; Internet of Things; Secure Communication; Transport Layer Security.**

I. Introduction

The Internet of Things (IoT) is rapidly growing due to devices being used in a wide range of domains such as smart homes, transportation, healthcare, or industrial scenarios. To assist the rapid growth in the number of application domains as well as in the number of IoT devices, several enabling technologies are required. Al-Fuqaha et al. [1] identify the latest developments in Radio Frequency Identification (RFID), smart sensor technology, and communication technologies and protocols as such enabling technologies. Together with RFID, the authors also mention Near Field Communication (NFC) as a very promising technology for the IoT since many smartphones nowadays are equipped with NFC-enabling technology. In the context of IoT related communication protocols, security is an often neglected aspect as highlighted by the increase in IoT related security breaches [2]. Such security breaches can be fatal if IoT devices are used in domains where malicious functionality could harm human lives such as industrial settings [3], or in healthcare applications [4]. To provide secured communication for IoT devices, protocols that are well known from the traditional Internet cannot be used due to their performance requirements. Especially, if considering NFC, protocols based on the Transmission Control Protocol (TCP) entail a large communication overhead and thus, are infeasible for most devices and scenarios.

978-1-5386-1456-3/18/$31.00 © 2018 IEEE

Although it is often believed that the limited communication range of NFC obviates the need for dedicated security measures [5], Haselsteiner and Breitfuß demonstrate that eavesdropping data is possible up to 10 m [6]. If transferred data is protected by weak security measures or even transferred unprotected, attacks are threatening the confidentiality of critical information [7], [8]. Plósz et al. [9] compare the provided security of various wireless communication technologies with NFC. The authors state that although there are several security related mechanisms defined in the NFC standard, many attacks are possible despite these mechanisms. Chen et al. [10] and Chatta et al. [11] list a large number of attacks that are feasible for attackers to perform due to weak or insufficient security in the NFC protocol. To mitigate such problems, many approaches for secured NFC communication have been proposed (see Section III Background and Related Work). However, the drawback with these approaches is that no standardized protocols are used. On the one hand, this fact complicates the use of NFC in IoT applications since the security of each application specific protocol needs to be proven separately. On the other hand, applying protocols with proven security features that where designed for the Internet to NFC communication entails a large overhead.

Contributions. We make the following contributions in this paper. We demonstrate Quick and Secured NFC (QSNFC), a protocol that is suited for NFC-equipped IoT devices. The provided security features are similar to TLS, while the protocol will require fewer messages to be exchanged during key agreement. Thus, the presented protocol will be suitable for any NFC-based IoT scenario, while allowing easy security proofs. To the best knowledge of the authors, no comparable protocol for NFC has been presented yet.

Outline. The remainder of this paper is structured as follows. We define our system model and list corresponding assumptions in Section II. In Section III, background information on involved technologies as well as related work for secured communication are given. Our QSNFC approach is presented in Section IV and evaluated regarding its performance and security in Section V. Section VI discusses example use-cases. This paper is then concluded with Section VII where also future work will be discussed.

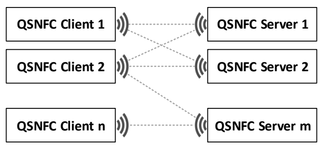


Fig. 1: System model in our proposed QSNFC approach: Over time, clients are capable to connect to an arbitrary number of servers. Servers can manage connections from an arbitrary number of clients (not simultaneous).

II. System Model and Assumptions

The system model we are considering when designing the QSNFC protocol is shown in Fig. 1. As shown in that model, the protocol is supposed to support an arbitrary number of QSNFC *clients* as well as an arbitrary number of QSNFC *servers*. Although the notion of server-client is not common in NFC solutions, we use these terms here to be compatible to other network related protocols such as TLS. Thus, we identify the following four entities in our system model:

QSNFC client: The QSNFC client is the entity that tries to establish a secured communication channel with the QSNFC server. Since this entity is initiating the NFC communication, it can be seen as the *active component* in NFC terms.

QSNFC Server: The QSNFC server is contacted by the QSNFC client in order to establish a secured communication channel. In NFC terms, the QSNFC server would be denoted as the *passive component*.

Communication channel: The communication channel that we assume in our system model is an NFC channel with the potential presence of an undetected adversary.

Adversary: The adversary present in our system model is assumed to be capable of eavesdropping and modifying ongoing NFC data. There is no assumption regarding the range in which these malicious activities are feasible.

III. Background and Related Work

A. Key Agreement and Transport Layer Security (TLS)

If two or more parties need to agree on a shared secret in the potential presence of an adversary, key agreement protocols are used. Usually, key agreement is performed between two parties over an unsecured channel such as the Internet. The final shared secret that is used as a key is composed of influences from all involved parties without revealing the key to any potential adversary that is capable of eavesdropping the key agreement process. One of the most widespread key agreement protocols is the Diffie-Hellman (DH) protocol [12] that is used for key agreement in the TLS protocol [13]. TLS uses TCP as transport protocol and is used to secure connection oriented applications such as web browsing, emails, or instant messaging. Due to its connection oriented nature, it introduces a lot of overhead which might not be suitable for resource constraint devices. Therefore, the Datagram Transport Layer Security (DTLS) protocol [14] was introduced, which uses the User Datagram Protocol (UDP) as its transport protocol. Both protocols have in common that at least one communication round trip time (RTT) is required for the key agreement process when establishing a secured channel.

B. Authenticated Encryption (AE)

AE is capable of providing data confidentiality, integrity, and authenticity by combining symmetric cryptography with Message Authentication Codes (MAC) in a secured way [15]. The widespread Advanced Encryption Standard (AES) provides modes of operation (e.g. AES-CCM, which is used in TLS) that are capable of providing AE [16].

C. Zero Round Trip Time (0-RTT)

The Internet Engineering Task Force (IETF) is currently working on the new TLS 1.3 standard that also includes a 0-RTT requirement [17]. The requirement specifies that the key agreement for recurring connections should not require a traditional handshake and thus, no round trip communication. Recurring connections are specified by the IETF as connections where the two communication partners previously already have established a secured channel, including the 1-RTT handshake required by the key agreement. Protocols that meet the 0-RTT requirement are, for example, OPTLS [18] and Google's Quick UDP Internet Connections (QUIC) protocol [19] that is designed for UDP connections.

D. Quick UDP Internet Connections (QUIC)

QUIC is a protocol presented and developed by Google to enable the transfer of websites via Hypertext Transfer Protocol (HTTP) over UDP instead of TCP [20]. The main goal of QUIC is to improve the perceived performance of web applications, compared to HTTP over TCP. This improvements are achieved by relying on multiplexed UDP connections and by using 0-RTT secured connections that are capable of providing the same level of security as TLS [21]. To make the protocol robust while using unreliable UDP packet transfer, QUIC also includes mechanisms to deal with packet loss, congestions, and error corrections. QUIC is integrated in current versions of Chrome and Chromium and according to Google deployed on thousands of their servers [22]. Also, an IETF working group for QUIC was founded in 2016.

E. Secured Near Field Communication (NFC)

NFC is a contactless communication technology that is based on several Radio-Frequency Identification (RFID) standards. Similar to High Frequency (HF) RFID, NFC operates at a frequency of $13.56\,\mathrm{MHz}$ and has a relatively short communication range of typically up to $10\,\mathrm{cm}$ with a bit rates of up to $848\,\mathrm{kbps}$. NFC is used in a very diverse range of fields, the most well known and widespread of them being payment applications. The IoT is believed to be a new major field for NFC applications [23] that will post new challenges for NFC technology, such as standardized secured protocols.

Although security is not a major topic in RFID related research, some promising approaches have been presented.

TABLE I: Comparison with related work. We compare the security attributes confidentiality, integrity, and authenticity. In addition, we state if a standardized protocol is used and if the used protocol is efficient in terms of communication overhead.

	Confidentiality	Integrity	Authenticity	Standardized	Efficient
Secret sharing [24]	✗/✓	✗	✗/✓	✗	✗
Secure UHF-RFID Tag [25]	✓	✗	✓	✗	✗
Mobile payment [26]–[28]	✓	✓	✓	✗	✗
Healthcare [29], [30]	✓	✓	✓	✗	✗
Car immobilizer [31]	✓	✓	✓	✗	✗
TLS over NFC [32]	✓	✓	✓	✓	✗
QSNFC [this work]	✓	✓	✓	✓	✓

For instance, Toyoda and Sasase [24] present a secret sharing mechanism that aims at confidentialiy distributing keys. Li et al. [25] present authentication and authorization mechanisms between tag and reader such that a trust relationship between these two devices can be established. However, due to the more powerful communication capabilities of NFC, more complex algorithms can be realized compared to RFID. Especially in the payment sector where security is of utmost importance, many application specific security solutions are presented [26]–[28]. Also in healthcare, where security weaknesses could directly impact the health of users or even threaten their lives, concepts for secured NFC communication are presented [29], [30]. Of course, there are also many other useful application scenarios for secured NFC communication, such as, for example, an NFC-based car immobilizer [31]. All of these approaches have in common that they implement application specific security mechanisms which is critical regarding the use of NFC in IoT devices since no general security assessment for the technology can be made. Urien and Piramuthu [32] try to mitigate this problem by proposing to use TLS over NFC. However, the TLS protocol was designed for TCP-based connections, and thus, entails a large protocol overhead. A comparison is given in Table I.

IV. Quick and Secured NFC

A. Classification in Layer Model

Before specifying our proposed QSNFC protocol in detail, we are going to classify it according to the TCP/IP protocol architecture layer as shown in Fig. 2. In that figure, the similarities to TLS and DTLS are highlighted. Similar to TLS and DTLS our QSNFC protocol resides directly underneath the actual application and provides capabilities for secured data transfer to the upper layer. As a transport protocol, the NFC Data Exchange Format (NDEF) that is based on application protocol data unit (APDU) packets is used. NDEF itself provides limited security measures, such as signature records. However, these security measures are shown to be vulnerable to certain attacks [33]. Therefore, we use NDEF as a transport protocol for QSNFC only, without relying on any of the available security features of NDEF.

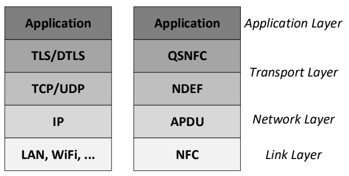


Fig. 2: Protocol stacks for TLS/DTLS and QSNFC respectively, layered according to the TCP/IP model. Both TLS/DTLS and QSNFC reside underneath the application layer and provide their functionality to higher layers, while relying on lower-layer protocols to perform data transfer.

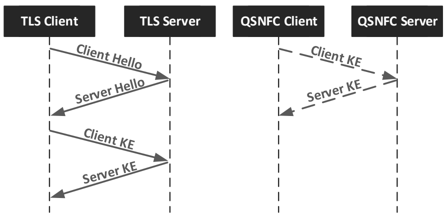


Fig. 3: Round trips required for TLS and QNFC. The TLS connection setup requires two round trips, while the QSNFC connection setup requires one round trip the first time two devices establish a connection and zero rount trips after that.

B. Connection Establishment

Since data transfers using QSNFC rely on secured data channels, key agreement needs to be performed. To meet the 0-RTT requirement for recurring connections, the client needs to cache information about the server if a successful *initial handshake* is performed. The performance of *subsequent handshakes* can then be improved by using this cached information. Fig. 3 demonstrate the handshake process in comparison to TLS. To identify cached information, QUIC uses a set of *URI*, *hostname*, and *port number*. Since this information is not available in NFC, unique identifiers will be used to identify entities. However, the handshake process itself that comprises of initial and subsequent handshake is *not* modified.

Initial handshake. Since on the first connection attempt the client has no cached information about the respective server, an initial handshake needs to be performed. To initiate this handshake, the client sends a so-called *inchoate client hello (CH)* message to the server, which recognizes the inchoate information and replies with a *reject (RJ)* message. This RJ message contains the following information:

(i) The server's long-term DH public value. This public key is used for the generation of subsequent keys and thus, needs to be cached by the client.

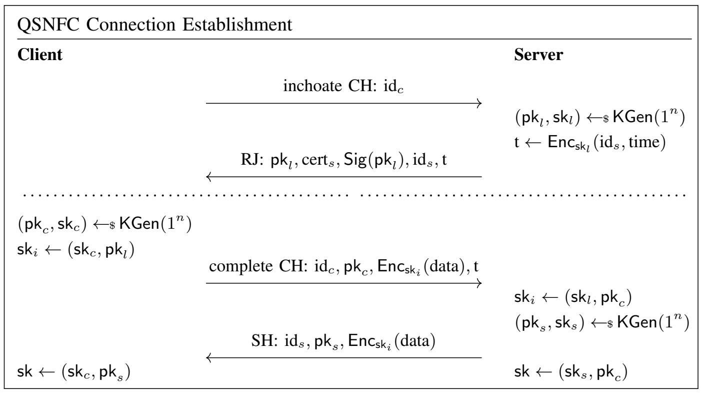

QSNFC Connection Establishment
Client
Server
inchoate CH: id_c
$(\mathsf{pk}_l, \mathsf{sk}_l) \leftarrow_\$ \mathsf{KGen}(1^n)$
$\mathsf{t} \leftarrow \mathsf{Enc}_{\mathsf{sk}_l}(\mathrm{id}_s, \mathrm{time})$
RJ: $\mathsf{pk}_l, \mathrm{cert}_s, \mathsf{Sig}(\mathsf{pk}_l), \mathrm{id}_s, \mathsf{t}$
$(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow_\$ \mathsf{KGen}(1^n)$
$\mathsf{sk}_i \leftarrow (\mathsf{sk}_c, \mathsf{pk}_l)$
complete CH: $\mathrm{id}_c, \mathsf{pk}_c, \mathsf{Enc}_{\mathsf{sk}_i}(\mathrm{data}), \mathsf{t}$
$\mathsf{sk}_i \leftarrow (\mathsf{sk}_l, \mathsf{pk}_c)$
$(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow_\$ \mathsf{KGen}(1^n)$
SH: $\mathrm{id}_s, \mathsf{pk}_s, \mathsf{Enc}_{\mathsf{sk}_i}(\mathrm{data})$
$\mathsf{sk} \leftarrow (\mathsf{sk}_c, \mathsf{pk}_s)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_s, \mathsf{pk}_c)$

Fig. 4: Connection establishment in QSNFC. All message types are inherited from QUIC [22]; only the content of these messages is adapted to be better suited for NFC. The messages above the dotted line represent the *initial handshake*. The messages underneath the dotted line represent the *subsequent handshake* only. The parameters are: the client's and server's id ($\mathrm{id}_{c|s}$), the long-term DH public and secret value (pk_l, sk_l), the server's and client's ephemeral DH public and secret value ($\mathrm{pk}_{s|c}$, $\mathrm{sk}_{s|c}$), the server's certificate (cert_s), the initial key (sk_i) and the final shared key (sk).

(ii) A certificate chain that authenticates the server and that needs to be verified during the initial handshake.
(iii) A signature of the long-term DH public value that is signed using the private key from the provided certificate chain's leaf certificate.
(iv) A source address token that contains the server's unique ID and a nonce from the server. This information is protected using AE. The client needs to send this token back to the server in subsequent handshakes to demonstrate ownership of the server's identity.

After the client has received this information, it can authenticate the server's long-term DH public value using the provided certificate chain and signature. In addition, the certificate chain is validated using a higher-ranking certificate. After that, the client sends a *complete CH* that contains the client's ephemeral DH public value as well as an optional payload that can already be encrypted using a key generated from the server's long-term DH public value and the client's ephemeral DH public value.

Subsequent handshake. Since the client already is in possession of the server's long-term DH public value, it can calculate a shared key using its own ephemeral DH public value. The client can then send a *complete CH*, without first sending a *inchoate CH* message as is done in the initial handshake. Thus, the first RTT from the initial handshake is not required and encrypted data can be sent to a known server instantly.

If the handshake is successful, a *server hello (SH)* message is sent by the server as response to the complete CH. The SH is encrypted using a key generated from the server's long-term public DH public value and the client's ephemeral DH public

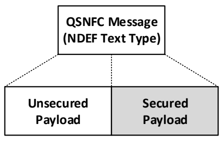


Fig. 5: Basic structure of QSNFC messages. A text record type NDEF message comprises unsecured and secured payload.

value. The SH message also contains the server's ephemeral DH public value. The complete connection establishment is shown in Fig. 4. After both involved entities are in possession of each others' ephemeral DH public value, a *forward-secure key* can be calculated for the connection. Thus, after the SH message is sent and received, both communicating entities switch to encrypting packets with the forward-secure keys.

C. Connection Tear Down

Although there is no *connection* concept in NFC, we previously discussed connection establishment. During this connection establishment, keys between server and client are exchanged and stored at the communication partners. In addition, also information regarding the other communication partner, such as a source address token need to be cached. However, since there are no concepts such as out-of-order packet reception or multiple streams that are known from TCP-based connection, also no connection tear down is needed in our QSNFC protocol. Methods regarding the replacement of cached keys and information will be discussed in Section IV-E.

D. Packet Structure

The basic packet structure of each QSNFC message is shown in Fig. 5. As can be seen there, any QSNFC message comprises unsecured as well as secured payload inside a *text record type* NDEF message. Similar to TLS and DTLS that use TCP and UDP as their transport protocols, we build QSNFC on top of NDEF messages for the following two reasons: (i) NDEF is a standardized data exchange format for NFC. Similar to TLS over TCP and DTLS over UDP we can utilize it as transport protocol without any modification to the underlying protocol. (ii) The security aspect of data transfer is cleanly separated from the data transfer aspect. That is, limitations such as maximum APDU size do not need to be considered in our proposed QSNFC protocol.

The handshake protocol shown in Fig. 4, comprises four different message types: CH messages, RJ messages, SH messages, and standard data messages (SD) that are shown in Fig. 6. The specified field lengths are calculated for $128\,\mathrm{bit}$ keys. All of these four message types contain three unsecured message attributes that are common to all of them.

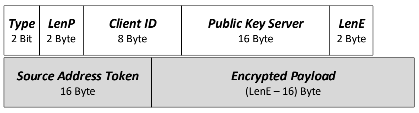
- *Type:* The message's type with allowed options `0b00` CH, `0b01` RJ, `0b10` SH, and `0b11` SD.
- *LenP:* Specifies the length of unsecured payload contained in this message. The length of this field is $2\,\mathrm{Byte}$.
- *LenE:* Specifies the length of secured payload contained in this message. The length of this field is $2\,\mathrm{Byte}$.

CH messages can be either an inchoate CH message or a complete CH message. The packet structure for both of these two types is shown in Fig. 6a. If a client initiates the initial handshake by sending an inchoate CH message, the *Client ID* is set accordingly with each other attribute being empty.

- *Client ID:* A unique ID identifying the client. The Client ID is contained in inchoate CH as well as in complete CH messages. The size of this attribute is $8\,\mathrm{Byte}$.
- *Public Key Client:* The client's ephemeral DH public value according to the handshake protocol shown in Fig. 4. This attribute is only set for complete CH messages and is $16\,\mathrm{Byte}$ in size.
- *Source Address Token:* Only sent by the client for complete CH messages. The value is obtained in the server's RJ message during initial handshake and stored at the client, for instance, in an SE. This attribute's size $16\,\mathrm{Byte}$.
- *Encrypted Payload:* In case of complete CH messages, also an encrypted payload is contained (see Fig. 4). The size of this field is determined by the *LenE* attribute.

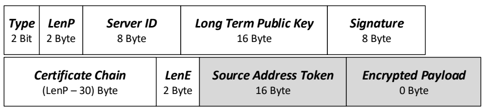
RJ messages are sent as a response to inchoate CH messages. The message contains information from the server that is required to perform connection establishment and key agreement. In contrast to other message types, no additional arbitrary payload can be included in RJ messages. The structure of RJ messages is shown in Fig. 6b.

- *Server ID:* A unique ID identifying the server. The Server ID is used by the client to match cached information to the correct server. This attribute has a size of $8\,\mathrm{Byte}$.



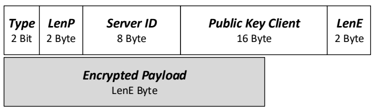
Type 2 Bit
LenP 2 Byte
Client ID 8 Byte
Public Key Server 16 Byte
LenE 2 Byte
Source Address Token 16 Byte
Encrypted Payload (LenE – 16) Byte

(a) CH message structure.

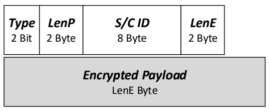


Type 2 Bit
LenP 2 Byte
Server ID 8 Byte
Long Term Public Key 16 Byte
Signature 8 Byte
Certificate Chain (LenP – 30) Byte
LenE 2 Byte
Source Address Token 16 Byte
Encrypted Payload 0 Byte

(b) RJ message structure.



(c) SH message structure.



(d) SD message structure.

Fig. 6: Message structures for CH, RJ, SH, and SD messages. The specified field lengths are are calculated for $128\,\mathrm{bit}$ keys.

- *Long Term Public Key:* The long-term DH public value that is used for key agreement and to calculate initial keys. Using this key, the client is capable of sending encrypted payload inside complete CH messages. The Long Term Public Key has a length of $16\,\mathrm{Byte}$.
- *Signature:* A signature of the Long Term Public Key that is used by the client to validate the integrity of that key. The signature has a size of $8\,\mathrm{Byte}$.
- *Certificate Chain:* The certificate chain that is used by the client to authenticate the server. Also, the private key of the certificate chain's leaf certificate is used to create the previously mentioned Signature. The Certificate Chain has a variable length, depending on the issued certificates.
- *Source Address Token:* This token is used by the client in subsequent handshakes. It includes the server's identifier and a nonce, and is protected by AE using the server's private key. The attribute has a size of $16\,\mathrm{Byte}$.
- *Encrypted Payload:* In RJ messages, only the Source Address Token is contained in the encrypted payload.

SH messages are sent by the server in response to a successful connection establishment. The structure of this message type is shown in Fig. 6c.

- *Server ID:* A unique ID identifying the server. The Server ID is used by the client to match cached information to the correct server. This attribute has a size of $8\,\mathrm{Byte}$.
- *Public Key Server:* The servers's ephemeral DH pub-

lic value according to the handshake protocol shown in Fig. 4. After reception of this attribute, server and client can calculate a forward-secure key to protect data transferred with subsequent SD messages. The size of this attribute is 16 Byte.

- *Encrypted Payload:* SH messages can contain arbitrary payload that needs to be transferred from the server to the client. This information is protected by AE using the previously established initial keys.

SD messages are exchanged between server and client after successful handshakes. Thus, after a forward-secure key was established between these two entities, SD messages with minimal protocol overhead can be used to transfer arbitrary payload in an efficient but secured way.

- *S/C ID:* The respective ID of either server or client is included to identify the sender of an SD message. The length of this attribute is 8 Byte.
- *Encrypted Payload:* Payload of arbitrary length that is protected by AE using the previously established ephemeral forward-secure keys.

E. Cached Data Replacement

The 0-RTT capability of QSNFC entails that information such as public keys and source address tokens need to be cached at server and client side. Since many NFC-enabled IoT devices are resource constraint in terms of memory capacity, a mechanism for cached data replacement needs to be included in our proposed QSNFC protocol. Depending on the use-case scenario in which the protocol is used, different replacement strategies might be better suited than others. Therefore, we briefly discuss three cached data replacement methods that are suitable for IoT devices due to their minimal overhead in terms of complexity and resource requirements [34].

Least Frequently Used (LFU): In the LFU algorithm, an access counter for each cached dataset is kept that counts the number of usages of that respective cached dataset. After a fixed number of connection establishments, all counters are reset. If a dataset needs to be evicted from memory, the dataset with the smallest access count is selected for replacement.

+ Only counters are needed which is a minimal overhead.
– The required regular reset of counters might lead to the eviction of often used datasets.

Least Recently Used (LRU): The LRU algorithm is a special variante of the LFU algorithm. Instead of counters, timestamps are kept for each cached dataset. Whenever a dataset is accessed, the timestamp is updated. If a datasets needs to be evicted from memory, the dataset that was accessed the farthest back in history is selected for replacement.

+ Frequently accessed datasets not falsely evicted.
– Resource constraint devices such as smart cards do not provide the required timestamps.

First In First Out (FIFO): Cached datasets are kept in a queue. Each new dataset is added at the front of the queue. If a dataset needs to be evicted from memory, the last element in the queue is selected for replacement.

+ Smallest overhead of all three methods.
– Dataset that gets cached first gets evicted first although that element might be the most used one.

V. Evaluation

A. Security Analysis

As shown in the packet structures (see Fig. 5), each packet that is transmitted using our proposed QSNFC protocol contains a section dedicated to secured payload. To protect the confidentiality, integrity, and authenticity of this secured payload, AE with either initial keys or ephemeral forward-secure keys is used. Depending on which type of key is used, two levels of secrecy can be provided: (i) Initial data that is protected using initial keys is protected at a level similar to TLS session resumption with session tickets. (ii) If the forward-secure keys are used, even greater secrecy can be provided since these keys are ephemeral. However, depending on the application and use-case, probably only one message round trip is needed. In this case, the initial keys only will be used to protect the data, without ever using the forward-secure keys. Any information that is transmitted unsecured in QSNFC (e.g. server and client identifiers, or public keysis) is considered non-critical. That means, an adversary would gain no advantage by learning this information. To highlight the provided security, we analyse the countermeasures provided by QSNFC for each of the threats to NFC that were identified by Haselsteiner and Breitfuß [6].

Eavesdropping: Since confidential information is transmitted protected by AE at any step of QSNFC (during the handshake and SD messages), an eavesdropper would only be able to learn information that is considered public, such as server and client identifiers, or public keys.

Data Corruption, Data Modification, Data Insertion: An adversary would not be able to corrupt, modify, or insert data in the secured payload section of QSNFC without such failures being detected by the protocol since the secured payload is protected by AE. However, *denial-of-service (DoS)* attacks cannot be mitigated by QSNFC since an adversary can corrupt transferred information at any time, and thus, cause data to be invalid. Also, if an adversary is able to modify information such as server or client identifiers, successful DoS is possible.

Denial-of-Service (DoS) attacks: DoS attacks cannot be mitigated by QSNFC (and any other wireless or contactless communication protocol since data corruption can only be detected but not prevented.

Man-in-the-Middle (MITM) attacks: By relying on certificates for authentication, and on a DH based key agreement, MITM attacks mitigated by QSNFC.

Physical attacks: Cryptographic operations that are required for our proposed key agreement process can either be performed in software, or in a dedicated hardware secure element, such as SIM cards or security controllers. To provide a higher level of security, tamper-resistant security controllers need to be used, such that potential adversaries are not able to extract confidential information using physical attacks [35].

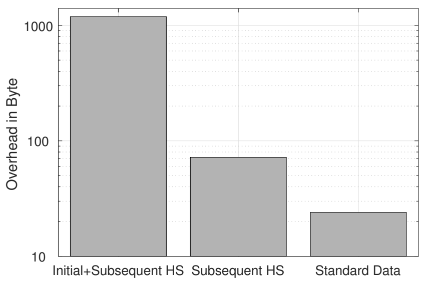


Fig. 7: Comparing three scenarios: initial and subsequent handshake (HS), subsequent HS, and standard data (SD) messages.

B. Protocol Overhead

To analyze the protocol overhead entailed by additional security mechanisms, we analyze three different communication scenarios: (i) Server and client have never complete a handshake, so an initial handshake followed by a subsequent handshake is required. (ii) Server and client only need to perform the subsequent handshake. (iii) Standard data transfer using SD messages that can be exchanged after a completed handshake. An overview of the resulting overhead for all three scenarios is shown in Fig. 7. The resulting overheads that can be seen there are 1182 Byte for initial+subsequent handshake, 72 Byte for subsequent handshake only, and 24 Byte for an exchange of two SD messages. The largest part of the overhead required by the initial handshake results from the included certificate chain. For this evaluation, we used a self-signed certificate that was generated with the following two commands: `openssl ecparam -name secp521r1 -genkey -out key.pem` and `openssl req -new -x509 -key key.pem -out cert.pem -days 365`. The generated certificate already has a size of roughly 1000 Byte. Using a chain of certificates would result in an even larger overhead. By not requiring initial handshakes for recurring connections, roughly 90 percent of overhead can be avoided, compared to traditional protocols such as TLS that involve key agreements. Thus, it can easily be seen why introducing a secured protocol for NFC that is capable of fulfilling the 0-RTT requirement is crucial.

C. Necessary Trade-off

Since most IoT devices are resource constraint, the adoption of a secured communication protocol also entails drawbacks for these devices in terms of complexity, energy consumption, cost, and memory requirements. However, since we see a secured communication channel as a given requirement, we only compare our presented approach to other protocols that involve key agreement, specifically TLS. While QSNFC significantly reduces the protocol overhead for recurring connections, additional, non-volatile memory is required to store cached information. While the additional memory required by QSNFC might be unproblematic for most IoT devices, very constraint devices might require additional memory to be added. This additional requirement will likely cause a slight increase in complexity, energy consumption, and subsequently device costs compared to using TLS. This means, a trade-off between communication efficiency and device complexity, energy consumption and costs needs to be considered.

VI. Example Use-Cases

Depending on the use-case scenario, the roles of server and client might be assigned differently, since a client must be able to validate the certificate chain provided by the server (see Section IV-B). For validation, the client either needs to be in possession of a higher-ranking certificate or have an active Internet connection. Therefore, we list three use-cases that we see as the most common scenarios for our QSNFC protocol and briefly discuss the role assignment.

Card and Reader: The reader in this scenario acts as active NFC device and provides the required energy to power the smartcard through its NFC field. Therefore, in this scenario the reader should be assigned the client role and initiate the connection establishment. Also, a reader will have the capability to validate the server's (smartcard) certificate chain over the Internet in most cases.

Smartphone and IoT Device: In this scenario, the smartphone should initiate the QSNFC handshake and thus, act as a client. Since all modern smartphones are equipped with ample storage and Internet connections, the required validation of the server's certificate chain is also feasible in such a setting.

Machine-to-Machine (M2M): In M2M communication settings such as Robot-to-Machine, the assignment of client and server role cannot be determined in general. The roles should be assigned accordingly, such that the validation of the server's certificate chain is feasible for the client.

VII. Conclusion and Future Work

To foster the use of NFC-technology in IoT devices and use-cases, a standardized and secured, yet efficient protocol is required. Currently, either application specific security solutions, or protocols that entail too much overhead such as TLS are used to secure NFC-based data transfers. The protocol presented in this publication, QSNFC, is designed with both standardized security mechanisms and efficiency in mind. The protocol fulfils the 0-RTT requirement to increase the performance of recurring connections between devices. Data confidentiality, integrity, and authenticity for transferred data is provided by relying on AE, while the imposed protocol overhead is kept at a minimum. As a trade-off compared to traditional protocols that involve key agreement such as TLS, our proposed algorithm requires more local memory to store cached information. As future work, we plan to investigate protocol improvements that further reduce the protocols overhead. Thus, making secured NFC data transfer even more efficient and suitable for IoT devices.

Acknowledgment

We thank our shepherd, Shashi Ramamurthy, and all the anonymous reviewers for their constructive comments. This work has been performed within the IoSense (http://iosense.eu) project. This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia. IoSense is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2016 and April 2019. More information: https://iktderzukunft.at/en/.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A Critical Analysis on the Security Concerns of Internet of Things (IoT)," *International Journal of Computer Applications*, vol. 111, no. 7, 2015.

[3] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and Privacy Challenges in Industrial Internet of Things," in *Design Automation Conference (DAC), 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[4] L. M. R. Tarouco, L. M. Bertholdo, L. Z. Granville, L. M. R. Arbiza, F. Carbone, M. Marotta, and J. J. C. de Santanna, "Internet of Things in Healthcare: Interoperatibility and Security Issues," in *Communications (ICC), IEEE International Conference on*. IEEE, 2012, pp. 6121–6125.

[5] L. Finžgar and M. Trebar, "Use of NFC and QR code Identification in an Electronic Ticket System for Public Transport," in *Software, Telecommunications and Computer Networks (SoftCOM), 19th International Conference on*. IEEE, 2011, pp. 1–6.

[6] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," in *Workshop on RFID security*, 2006, pp. 12–14.

[7] M. Emms and A. van Moorsel, "Practical Attack on Contactless Payment Cards," in *HCI2011 Workshop-Heath, Wealth and Identity Theft*, 2011.

[8] P. Fraga-Lamas and T. M. Fernández-Caramés, "Reverse Engineering the Communications Protocol of an RFID Public Transportation Card," in *RFID (RFID), 2017 IEEE International Conference on*. IEEE, 2017, pp. 9–11.

[9] S. Plósz, A. Farshad, M. Tauber, C. Lesjak, T. Ruprechter, and N. Pereira, "Security Vulnerabilities and Risks in Industrial Usage of Wireless Communication," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. IEEE, 2014, pp. 1–8.

[10] C. H. Chen, I. C. Lin, and C. C. Yang, "NFC Attacks Analysis and Survey," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on*. IEEE, 2014, pp. 458–462.

[11] N. A. Chattha, "NFC - Vulnerabilities and Defense," in *Information Assurance and Cyber Security (CIACS), 2014 Conference on*. IEEE, 2014, pp. 35–38.

[12] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[13] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Requests for Comments, RFC 5246, August 2008.

[14] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," Internet Requests for Comments, RFC 6347, January 2012, accessed on 2017-11-20. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6347.txt

[15] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.

[16] D. McGrew and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)," Internet Requests for Comments, RFC 6655, July 2012.

[17] B. Dowling, M. Fischlin, F. Günther, and D. Stebila, "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1197–1210.

[18] H. Krawczyk and H. Wee, "The OPTLS Protocol and TLS 1.3," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 81–96.

[19] M. Fischlin and F. Günther, "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 1193–1204.

[20] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: an Experimental Investigation of QUIC," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 609–614.

[21] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, "How Secure and Quick is QUIC? Provable Security and Performance Analyses," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 214–231.

[22] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 183–196.

[23] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[24] K. Toyoda and I. Sasase, "Secret Sharing Based Unidirectional Key Distribution with Dummy Tags in Gen2v2 RFID-enabled Supply Chains," in *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 2015, pp. 63–69.

[25] Q. Li, Z. Sun, J. Huang, S. Liu, J. Wang, N. Yan, L. Wang, and H. Min, "Secure UHF-RFID Tag for Vehicular Traffic Management System," in *RFID (RFID), 2017 IEEE International Conference on*. IEEE, 2017, pp. 26–29.

[26] K. S. Kadambi, J. Li, and A. H. Karp, "Near-Field Communication-Based Secure Mobile Payment Service," in *Proceedings of the 11th international Conference on Electronic Commerce*. ACM, 2009, pp. 142–151.

[27] M. Pasquet, J. Reynaud, and C. Rosenberger, "Secure Payment with NFC Mobile Phone in the SmartTouch Project," in *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*. IEEE, 2008, pp. 121–126.

[28] U. B. Ceipidor, C. M. Medaglia, A. Marino, S. Sposato, and A. Moroni, "KerNeeS: A protocol for mutual authentication between NFC phones and POS terminals for secure payment transactions," in *Information Security and Cryptology (ISCISC), 2012 9th International ISC Conference on*. IEEE, 2012, pp. 115–120.

[29] D. Sethia, D. Gupta, T. Mittal, U. Arora, and H. Saran, "NFC Based Secure Mobile Healthcare System," in *Communication Systems and Networks (COMSNETS), 6th International Conference on*. IEEE, 2014, pp. 1–6.

[30] A. J. Jara, M. A. Zamora, and A. F. Skarmeta, "Secure use of NFC in medical environments," in *RFID Systems and Technologies (RFID SysTech), 2009 5th European Workshop on*. VDE, 2009, pp. 1–8.

[31] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi, "Smart Keys for Cyber-Cars: Secure Smartphone-based NFC-enabled Car Immobilizer," in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 233–242.

[32] P. Urien and S. Piramuthu, "LLCPS and SISO: A TLS-Based Framework with RFID for NFC P2P Retail Transaction Processing," in *RFID (RFID), 2013 IEEE International Conference on*. IEEE, 2013, pp. 152–159.

[33] M. Roland, J. Langer, and J. Scharinger, "Security Vulnerabilities of the NDEF Signature Record Type," in *Near field communication (NFC), 2011 3rd International Workshop on*. IEEE, 2011, pp. 65–70.

[34] S. Podlipnig and L. Böszörmenyi, "A Survey of Web Cache Replacement Strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

[35] T. Korak, T. Plos, and M. Hutter, "Attacking an AES-Enabled NFC Tag: Implications from Design to a Real-World Scenario," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2012, pp. 17–32.

Automated Authentication Credential Derivation for the Secured Configuration of IoT Devices

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Andrea Höller, Sarah Haas, Rainer Matischek
Design Center Graz
Infineon Technologies Austria AG
Graz, Austria
{andrea.hoeller, sarah.haas, rainer.matischek}@infineon.com

***Abstract*—The number of embedded systems and resource constrained devices is steadily increasing due to trends such as the Internet of Things (IoT) and the Industrial IoT. With that, also the frequency and extent of cyber-attacks that target these systems are rapidly increasing. Not only are most devices not adequately secured against such attacks, but also use default settings and authentication credentials. These problems are especially critical if the devices are deployed in industrial contexts where managing configurations and authentication credentials is a complex and inconvenient process. Therefore, in this paper, we present a device configuration approach that automatically derives authentication credentials from device configurations. We demonstrate that the additional performance required by our approach is acceptable while the provided security is reasonable when compared to traditional authentication approaches such as passwords. The security benefits of our approach are highlighted by an extensive security and threat analysis that demonstrates that 9 out of 10 identified threats are mitigated by our approach.**

***Index Terms*—Embedded Security; Device Configuration; Automated Credential Derivation; Authenticated Key Exchange.**

I. Introduction

Internet of Things (IoT) devices are constantly exposed to potential adversaries and threats due to them being connected to the Internet continuously [1], [2]. Not only are privately used IoT devices targeted by attacks [3], but also IoT devices used in industrial contexts. Providing security for such Industrial IoT (IIoT) devices is especially crucial since security weaknesses might reveal confidential information, harm industrial processes, or in extreme cases, might cause physical damage and threaten human lives [4]. In addition to these safety critical issues, also privacy concerns due to industrial espionage caused by infeasible IoT device security need to be considered [5]. Several studies have shown that among the security weaknesses of IoT devices, using weak or even default authentication credentials is one of the primary reason for successful attacks [6], [7], [8]. Cam-Winget et al. [9] point out that very often remote access channels used for firmware and configuration updates are vulnerable to such weaknesses. In order to mitigate these issues, using sophisticated authentication mechanisms such as two-factor authentication could be one possible solution. However, due to most IoT devices being resource constrained, applying such concepts will not be possible [10]. Thus, most systems still rely on traditional authentication credentials such as username and password combinations. In order to increase the security of IoT devices still using such authentication credentials, passwords must frequently be changed while complying with password composition policies (i.e., requiring symbols and numbers in passwords). However, enforcing such policies often leads to even weaker passwords being chosen by users [11].

Therefore, in this paper, we present an automated credential derivation process used for secured configuration of IoT devices. To alleviate users of the need to choose sophisticated authentication credentials, our proposed approach will derive these credentials from previously applied configuration updates. As an example, let us consider a simple WiFi door sensor that can be used for monitoring purposes. We assume the only configuration parameters of such a sensor are its sampling frequency (`SINT`), a username (`USER`), and the corresponding password (`PASSWORD`). Fig. 1 shows an exemplary initial configuration C_0 in which `SINT` is set to an interval of 10 minutes and no `PASSWORD` for the default `USERNAME` root is configured. In this example, a user might change the sampling interval `SINT` but not the default `PASSWORD`, as highlighted in our example. In contrast to that, if our proposed approach is applied, updating `SINT` from configuration C'_0 to the value in C'_1, will also cause the default `PASSWORD` to change.

To obviate the need for users to remember these automatically generated authentication credentials, we demonstrate two mechanisms that are used to manage the authentication credentials for IoT devices. We also propose a hardware architecture that is capable of providing tamper resistance to protect confidential information. To demonstrate the provided security level of our proposed approach, we will compare the achievable authentication credential strength to a traditional password-based approach.

Contributions. In this paper, we present an automated authentication credential derivation process that improves the security of IoT devices while not complicating their usage. Authentication credentials are automatically derived whenever a configuration update is performed. Thus, insecure default passwords are changed as soon as the device is configured for its first use. Configurations are managed by a central instance, such that users do not need to remember their authentication credentials. To the best knowledge of the authors, no such contribution was previously made.

978-1-5386-4155-2/18/$31.00 ©2018 IEEE

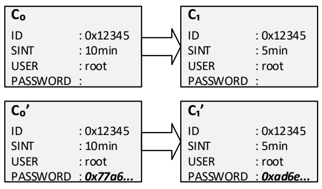


Fig. 1. Example configuration update changing the sampling interval `SINT` only ($C_0 \rightarrow C_1$) compared to using our proposed approach where the same change would trigger the automated credential derivation ($C_0' \rightarrow C_1'$).

TABLE I
COMPARING THE ATTRIBUTES CONFIGURATION MANAGEMENT (MANAGED), SUFFICIENT SECURITY (SECURED), AND AUTOMATED CREDENTIAL DERIVATION (CRED. DERIV.) WITH RELATED WORK.

Related work	Managed	Secured	Cred. deriv.
Perera et al. [22]	✓	✗	✗
Nastic et al. [23]	✓	✗	✗
Perumal et al. [24]	✓	✗	✗
Santoso and Vun [25]	✓	✓	✗
Ulz et al. [26]	✓	✓	✗
This work	✓	✓	✓

Outline. The remainder of this paper is organized as follows. In Section II we give background information on technologies used in our approach and discuss related work. We then define our system model and discuss assumptions made regarding this model in Section III. Our proposed automated credential derivation process is then presented in Section IV and evaluated in Section V. We then conclude this paper in Section VI where also potential future work is discussed.

II. BACKGROUND AND RELATED WORK

In this section, we give background information on technologies involved in our proposed approach, as well as discuss related work for IoT device configuration.

A. Key Agreement Protocols

Key agreement protocols are used to perform key agreement between two or more communication partners over an unsecured channel such as the Internet. Usually, during key agreement, all involved partners can influence the key agreement process. The final key is composed of influences from all involved partners without revealing the key to any adversary that is capable of eavesdropping communication over the unsecured communication channel. One of the most widely used key agreement protocols, Diffie-Hellman (DH) [12], is used in the Transport Layer Security (TLS) protocol.

Encrypted key exchange (EKE) protocols belong to the category of key agreement methods that use passwords to authenticate the partners involved in the key agreement process [13]. The password is used as shared knowledge between involved partners and is incorporated into the key agreement process such that only partners that are in possession of the correct password can mutually agree on a key. The resulting session key is considered to be appropriately secure even if the shared knowledge is drawn from a small set of values. In the Simple Password-Based Encrypted Key Exchange (SPAKE) protocol [14] a modified DH algorithm that uses a shared password for key derivation during key agreement is used.

B. Tamper Resistant Hardware

Tamper resistant hardware provides a secured execution environment (SEE) as well as secured storage. Therefore, these components can be used for the execution of critical code parts such as cryptographic algorithms and for storing confidential information such as key material. A device that is labeled as tamper resistant [15] is capable of mitigating physical attacks such as non-invasive and invasive side-channel attacks [16] by applying appropriate countermeasures. The level of security that is provided by a certain SE can be assessed based on the common criteria (CC) information technology security evaluation [17], such that SEs can be compared based on their provided security level.

C. Authentication for IoT Devices

Jan et al. [18] state that authenticating devices before communicating with these devices is critical, especially in the IoT where a high number of potentially unsecured devices are present. The authors, however, highlight that due to most devices being resource constrained, no complex cryptographic operations can be performed. For example, an approach that performs mutual authenticated Diffie-Hellman key exchange using public keys by Xu et al. [19] might be infeasible due to the need to store many public keys. Roman et al. [20] state that an infrastructure for mutual authentication for IoT devices will be needed to account for such resource constrained devices. The authors also discuss an important principle that is applied in this paper: system security for constrained devices should rely on *what I have* and *what I know*. Liu et al. [21] discuss authentication protocols for the IoT and state that such a protocol has several tasks, one of them being *key switching*.

D. Related Work Secured Device Configuration

Configuring devices in the IoT is an active topic in research due to the various challenges presented by the large number of resource constrained devices. One approach to handle the large number of devices that need to be configured is to use self-configuration mechanisms [27], [28]. If self-configuration is not applicable, manual configuration processes, as well as initial provisioning methods need to be secured and simplified as stated by Truong et al. [29]. To support the configuration process, Nastic et al. [23] suggested using a central configuration management solution. Perumal et al. [24] presented an IoT device management framework that is suitable for smart home scenarios. In this framework, IoT device can be managed by a smartphone. However, configurations are stored and transferred unprotected. Regarding the distribution of configuration updates, using the Internet is the most common approach [23], [22], and thus, security needs to be considered. However, most solutions do not consider complete system

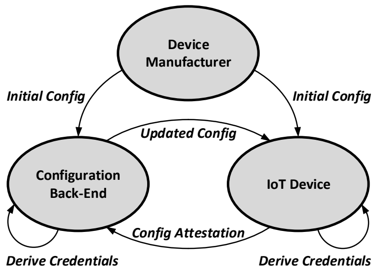


Fig. 2. Proposed principle of using configuration updates for automated credential derivation. Configuration updates are performed by different entities during an IoT device's entire lifecycle.

security (protocol, device, and overall system) but rather cover specific security aspects. Santoso and Vun [25] presented a secured configuration architecture for smart home appliances. The approach relies on mutual authentication based on a pre-shared secret. In their approach, a smartphone is used to manage existing configurations. However, the authors did not elaborate on how this shared secret is initially transferred to the device. They also did not specify, if a shared secret can be changed, for instance, if a device is resold. Ulz et al. [26] demonstrated an approach based on Near Field Communication (NFC) and dedicated hardware security elements. This approach only allows configuration updates via NFC and includes a secured communication protocol. However, the used symmetric cryptography results in a key distribution problem that was not covered by the authors. Still, since this approach is very promising, we extend it by the automated authentication credential derivation that is presented in this paper. A summary of related work is shown in TABLE I.

III. System Model and Assumptions

For our automated authentication credential derivation and IoT device configuration approach, we assume a system model that comprises the three entities shown in Fig. 2.

The **IoT Device** is the device for which configuration updates secured by automatically generated authentication credentials should be performed. On this device, the configuration update process, as well as the configuration data, need to be protected by appropriate security measures. Also, the authentication credential derivation process needs to be protected by appropriate security measures. We assume that potential attackers will be able to gain physical access to this device; therefore, appropriate countermeasures to protect confidential information need to be taken.

The **Device Manufacturer (DM)** produces the IoT device. Since in most cases devices are shipped pre-configured, the DM is responsible for applying initial configurations that lead to automatically generated initial authentication credentials as well. We assume the DM trustworthy.

The **Configuration Back-End (CBE)** is used to manage device configurations for an arbitrary number of IoT devices. Any configuration change except the *initial configuration* is initiated from this entity. The initial configuration is applied by the DM, and thus, this information needs to be imported into the CBE in our approach. We assume the CBE to be adequately secured against any type of attack. That is, the confidential information that is stored there is assumed to be protected against security breaches.

As can be seen in the system model shown in Fig. 2, any update process initiated from the CBE triggers a configuration attestation process from the IoT device. We assume such an attestation process that is capable of attesting the currently applied configuration to a remote instance to be existent in our system model, since the focus of this paper is on the automated credential derivation process. Configuration attestation processes that are suitable for IoT devices have been presented in literature [26], [30], [31].

IV. Automated Credential Derivation

In this section, our proposed automated authentication credential derivation process for the secured configuration of IoT devices is presented. We will discuss the basic process and the session key generation. After that, our approach is compared to traditional password-based authentication. We then list mechanisms that are specific to private or industrial use of IoT devices. Finally, we briefly discuss the hardware architecture we propose for a secured IoT device configuration process.

A. Basic Process

To automate the authentication credential derivation process, applied configurations of IoT devices are used in our approach. Any configuration update will thus trigger the derivation of new authentication credentials. The basic process is shown in Fig. 3. For this example we assume two entities, *Alice* and *Bob* that want to perform a secured configuration update. We assume that both Alice and Bob are in possession of the same shared secret, the k-th iteration of Bob's configuration, C_k. Having this information, Alice and Bob perform a session key generation based on C_k that yields the session key SK_k. Alice then encrypts the configuration update that results in the k+1-th configuration C_{k+1} with this session key and sends it to Bob who is able to decrypt that information using SK_k. After verifying and either applying or rejecting the configuration update, Bob informs Alice about the configuration applied to again establish the same level of shared knowledge. If Bob applies C_{k+1}, he and Alice will be able to generate a session key based on this information. If C_{k+1} is rejected, Alice and Bob still will be able to use C_k as basis for their session key generation process.

Advantages. Using this mechanism entails the following two advantages for users, compared to traditional authentication mechanisms such as passwords:

1) Device security is increased since any configuration change triggers the automated creation of new authen-

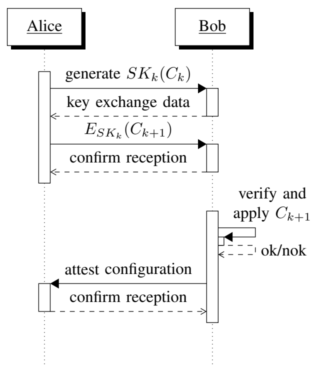


Fig. 3. Sequence diagram demonstrating the configuration update and attestation process where Alice wants to send new configuration data to Bob. Both communicating partners are in possession of knowledge regarding the currently applied configuration and thus, the shared knowledge needed to generate a session key SK.

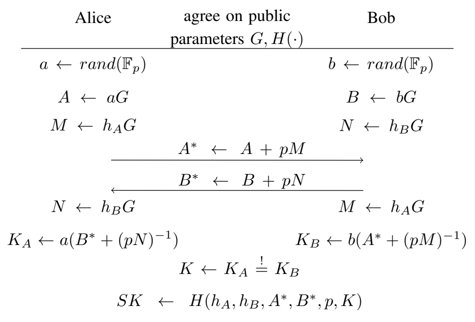

Alice
agree on public parameters $G, H(\cdot)$
Bob
$a \leftarrow rand(\mathbb{F}_p)$
$b \leftarrow rand(\mathbb{F}_p)$
$A \leftarrow aG$
$B \leftarrow bG$
$M \leftarrow h_A G$
$N \leftarrow h_B G$
$A^* \leftarrow A + pM$
$B^* \leftarrow B + pN$
$N \leftarrow h_B G$
$M \leftarrow h_A G$
$K_A \leftarrow a(B^* + (pN)^{-1})$
$K_B \leftarrow b(A^* + (pM)^{-1})$
$K \leftarrow K_A \overset{!}{=} K_B$
$SK \leftarrow H(h_A, h_B, A^*, B^*, p, K)$

Fig. 4. SPAKE2 protocol [14] that is modified in our presented approach. In this protocol, G is the ECC generator point, $H(\cdot)$ is a one-way function, h_A and h_B are Alice's and Bob's hashed identities respectively, and p is a shared secret between Alice and Bob.

tication credentials. It is basically impossible to operate devices using default authentication credentials.

2) Users do not need to remember sophisticated passwords since authentication credentials are automatically derived from device configurations which are managed by CBE. Thus, users basically use the CBE as a password manager.

B. Session Key Generation

The session key generation shown in the sequence diagram in Fig. 3 will be performed by using SPAKE2 [14] which is an EKE protocol. This protocol uses a username and password combination in the key derivation of the session key generation process. The protocol is shown in Fig. 4. Since SPAKE2 relies on Elliptic-Curve Diffie-Hellman (ECDH) [32] the ECC generator point G and a one-way function $H(\cdot)$ are defined as public parameters between *Alice* and *Bob*. After that, the respective user identities u_A and u_B as well as a shared secret p are used in the key derivation process to generate a session key SK. It is important to distinguish two types of secret in this information flow: (i) the shared secret (p) that is used for mutual authentication between involved parties and (ii) the session key (SK) that is generated by the algorithm to encrypt subsequent communication. A new shared secret p' could then be transferred using the encrypted channel.

Modifications: If we now apply the SPAKE2 algorithm to our process previously defined in Fig. 3, only a minor adoption to the algorithm needs to be made. For the session key SK to be dependent on the currently applied configuration C_k, we redefine the shared secret between Alice and Bob as

$$p := H(C_k).$$

By applying this definition, each session key generated will be dependent on C_k. However, defining C_k as shared secret in the EKE process also implies that C_k is used for mutual authentication between Alice and Bob. Therefore, configuration data needs to be treated as confidential information as was assumed in our system model. The advantage of this approach is that any change to a configuration made will automatically trigger an authentication derivation process based on the new configuration and thus, will mitigate the problem of vulnerable IoT devices due to relying on default username and password combinations.

C. Comparison to Password-Based Authentication

Assumptions. Since we are relying on the security provided by the SPAKE2 algorithm, we refer to the corresponding security proof by Abdalla and Pointcheval [14]. Further, we assume that the device is using the hardware architecture proposed in Section IV-E and thus, provides sufficient countermeasures to mitigate physical or side-channel attacks.

To evaluate the achievable level of security provided by our presented approach, we will discuss the strength of our authentication credentials by applying the Password Quality Indicator (PQI) presented by Ma et al. [33]. The authors discuss why entropy alone cannot be used as a quality indicator for passwords and define the $PQI(D, L)$ where D is the Levenshtein distance between two strings and L is the effective password length. According to Ma et al. a credentials are considered *good* if $D \geq 3$ and $L \geq 14$. Since in our approach authentication credentials are automatically derived from configurations, we need to apply these parameters to configuration data.

Ma et al. [33] define D as the Levenshtein distance between passwords and a dictionary of words. To be applicable for our evaluation, we not only need to consider a dictionary but a set of *observable parameters*. Such parameters could include, for example, a WiFi name. Both D and L are then applied to the values of configuration key-value pairs only. Applied to the simple example shown in Fig. 1, we would use

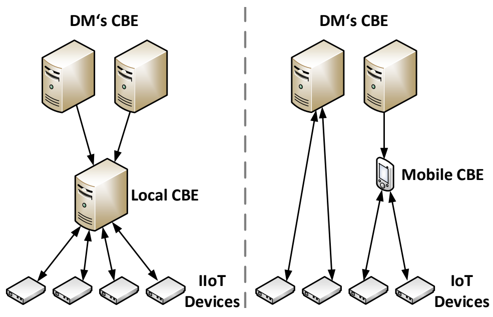


Fig. 5. Context specific mechanisms for configuration management. A local CBE is deployed to manage IIoT device configurations. Consumer IoT devices are either managed by the manufacturer's global CBE or by a mobile CBE.

a set of $\{wifi1, root\}$ for authentication credential derivation. Obviously, this results in $D = 1$ and $L = 9$ which is considered insecure by our measure. In comparison, passwords with a length of 9 characters and a Levenshtein distance of 1 can be considered quite common (e.g., *password1*). This shows that in general, our proposed approach would not be more secure than relying on user-defined passwords. To mitigate this problem, a configuration update could be automatically extended by random data that can be securely generated by a true random number generator (TRNG) [34] provided by most Secure Elements (SEs). In our approach, we suggest to extend configurations that are created at the CBE with a so-called nonce. The nonce is generated by the CBE and added as a configuration parameter to the encrypted configuration. The IoT device then is capable of extracting the transferred nonce from applied configuration updates. By doing so, both D and L can be increased to an arbitrary length. Similar approaches have been shown to enhance the security of password-based authentication methods [35].

D. Context Specific Mechanisms

Depending on the context in which an IoT device is operating, different mechanisms for device configuration management and password reset are required. Therefore, we propose different system architectures that are shown in Fig. 5. In *industrial scenarios* where IIoT device configurations contain confidential information that needs to be kept private, a local CBE can be deployed in an internal network. As can be seen in the left half of Fig. 5, such a local CBE needs to import initial configurations from the respective DMs. After that, the previously discussed device configuration and update process is performed between managed IIoT devices and the local CBE only. Since the local CBE can be viewed as a single point of failure, appropriate measures to properly secure information need to be taken. In *personal settings* it is infeasible to deploy a local CBE. Therefore, we propose two different system architectures that are shown in the right half of Fig. 5. On the one hand, a DM's CBE can be used for device configuration management. In this case, the configuration update and attestation process is performed between these two entities. If similar to the industrial scenario confidential configuration data needs to be kept private, we propose the use of a mobile CBE, such as also present in the system model presented by Ulz et al. [26]. Compared to the other two system architectures, data loss might be more probable when using such a mobile CBE. Therefore, a configuration reset and thus, authentication credential reset mechanism needs to be included in our presented approach as well. A *configuration reset* will be required whenever a CBE's configuration database is inconsistent such that the currently applied configuration on the managed device is not known to the configuration database. Such inconsistencies could be caused by loss of data on a mobile or local CBE. As a potential measure, we propose to include a hard-reset method into IoT devices that reset the currently applied configuration to the initial configuration C_0 which can be easily imported again into any CBE. The download of initial configurations could be achieved, for example, by downloading the information from DM's CBE or by applying QR codes to the respective devices that contain the required information. This approach would be similar to current approaches where default credentials are printed on stickers that are attached to the devices.

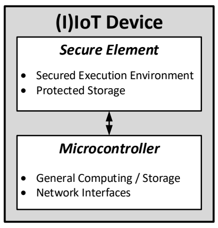


Fig. 6. Proposed hardware architecture for (I)IoT devices.

E. Hardware Architecture

Configurations stored on devices may contain confidential information such as key material or production-relevant information in the case of industrially used devices. To protect this confidential information, appropriate security measures need to be taken in hardware as well. Since attackers might be able to gain physical access, we propose to use the hardware architecture shown in Fig. 6 that suggests including an SE into IoT devices. In our proposed architecture, this SE is responsible for performing security critical operations such as the automated credential derivation presented in this paper. In addition, confidential configuration data is stored in the SE due to its tamper resistant nature. The micro-controller is used for general purpose computing tasks, and thus, a dual-execution principle is applied [36]. In addition, the device's required network interfaces are provided by the micro-controller.

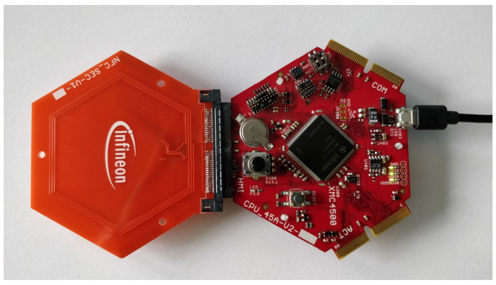


Fig. 7. Research IoT device prototype used for performance evaluation. The upper hexagon shaped board contains the XMC4500 microcontroller, the SLE78 SE is embedded in the lower hexagon shaped board.

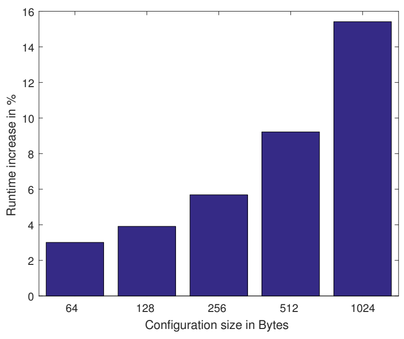


Fig. 8. Relative runtime increase due to applying hash function to configurations of different sizes. As a baseline we use a standard SPAKE2 implementation that requires a 32 Bytes hash operation for the password.

V. Evaluation

The evaluation of our presented credential derivation process is twofold. First, we discuss the overhead compared to traditional authenticated key exchange algorithms in a performance analysis. Second, we also analyse the security properties of our presented approach in a threat analysis.

A. Performance Analysis

To evaluate our proposed approach for automated authentication credential derivation, we use a research prototype according to the hardware architecture shown in Fig. 6. The prototype comprises an *Infineon XMC4500* microcontroller and an *Infineon SLE78* SE. The protoype is shown in Fig. 7. Since in our proposed architecture all security relevant operations are executed on the SE, the complete EKE process is implemented on this controller. To highlight the extent of runtime overhead resulting from the modified SPAKE2 algorithm, we conducted a performance analysis. As shown by Pieber et al. [37], running SPAKE2 on resource constrained hardware such as Infineon's SLE78 results in a larger runtime when compared to traditional ECDH. Depending on the configuration size, our approach requires extended hashing operations compared to the standard SPAKE2 implementation. As a baseline, we consider SPAKE2 with block sizes of 16 Bytes each for username and password, resulting in the hash function being executed on 32 Bytes of data. Fig. 8 shows the resulting overhead when hashing configurations of different sizes instead of a single password. As can be seen there, a configuration of 512 Bytes would result in an increase of runtime of roughly 10% compared to the basic SPAKE2 implementation. However, in their paper, Pieber et al. [37] show that pre-computing the required hash values can reduce the runtime of a SPAKE2 implementation to values similar to traditional ECDH. Of course, pre-computing these values based on applied configurations is also a possibility for our proposed approach. Thus, mitigating the additionally required runtime during session key generation.

B. Threat Analysis

To demonstrate the robustness of our presented approach against various types of attacks, we conduct a threat analysis [38]. In this analysis, we identify the involved *Entities (E)* and *Assets (A)* that need to be protected by our approach. We then list potential *Threats (T)*, and *Countermeasures (C)* that are provided by our approach to mitigate these threats. If a threat is not entirely mitigated by our approach, the residual *Risks (R)* are also listed. In addition, we categorize all threats according to the STRIDE threat model [39]. Threats are categorized by their potential impact, according to the following criteria: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. Although we do not claim that our presented threat analysis is exhaustive, we do think that it adequately highlights the security of our proposed automated authentication credential derivation process.

Entities. The following entities are identified in our automated authentication derivation process. If necessary, we list assumptions regarding the respective entity to narrow the scope of this threat analysis.

(E1) The IoT or IIoT device that is being configured.

(E2) The DM's CBE. Since the security aspects of a CBE are out of scope for this paper, we assume that the DM's CBE is sufficiently secured against attacks such that no loss of confidential data will occur there.

(E3) The device user's local or mobile CBE. We also assume that the users's CBE is sufficiently secured against attacks such that no confidential data will be lost by attacks targeting the user's CBE.

(E4) A potential adversary. We do not make any assumption about the extend of attacks an adversary is able to perform. That is, we assume the adversary is able to perform remote attacks as well as physical attacks.

Assets. We identify the following assets that need to be protected by our proposed approach.

(A1) The IoT device itself must be protected from malicious actions that might be enabled by security weaknesses.

(A2) The configuration data that is transferred must be protected since it might contain confidential information such as keys or production relevant information for IIoT devices.

Threats. After identifying involved entities and assets that need to be protected, we are going to list identified threats, categorize them based on the STRIDE threat model, and list corresponding countermeasures or residual risks.

(T1) An adversary might be able to eavesdrop transferred data, and thus, be able to learn confidential information.
STRIDE: I
(C1) Transferred configuration data is encrypted using session keys. Therefore, data confidentiality is provided.

(T2) An adversary might act as man-in-the-middle (MITM) and impersonate the CBE and the IoT device respectively.
STRIDE: S, T, R, I
(C2) When using the generated session key for authenticated encryption, data confidentiality, integrity, and authenticity can be provided.

(T3) The adversary can act as MITM during key agreement.
STRIDE: S, T, R, I, E
(C3) Since our approach is based on SPAKE2, MITM attacks are mitigated by mutual authentication and the DH principle.

(T4) An adversary easily can learn the initial configuration C_0, record all data transfers and thus, infer any subsequent authentication credential.
STRIDE: S, T, R, I, E
(C4) Since session keys that cannot be learned by the adversary are used to protect transferred data, the adversary cannot learn any subsequent authentication credential.

(T5) An adversary might learn a configuration C_k by observing the IoT device's environment and behaviour. The adversary then is able to infer the current session key.
STRIDE: S, T, R, I, E
(C5) Random information that is added to configuration data and transferred to the IoT device mitigates this threat.

(T6) The IoT device's user might not change the initial configuration C_0, and thus, no new authentication credential is derived automatically.
STRIDE: S, T, E
(C6) A devices that is running on default configurations will not be useful for the user. For instance, the device must at least be connected to a network.

(T7) An adversary might perform attacks such as trying to provoke buffer overflows to compromise the device and reveal confidential information.
STRIDE: T, I
(C7) Since all cryptographic operations are performed at the SE, and confidential information is also stored there, such attacks are mitigated by the SE's security measures.

(T8) An adversary might perform physical attacks targeting the device to reveal confidential information.
STRIDE: T, I
(C8) In our approach, we are using tamper resistant SE. The SE's level of security is verified by the CC certification process.

(T9) Intentional or unintentional backdoors might exist in software or hardware (e.g. for debugging purposes) that can be exploited by an adversary.
STRIDE: S, T, R, I, D, E
(C9) When including a CC certified SE, the trustworthiness of all hardware and software components of the device need to be verified in a certification process.

(T10) Denial-of-Service (DoS) attacks targeting the proposed configuration interface with automated authentication credential derivation.
STRIDE: D
(C10) Since all cryptographic operations are performed at the SE, normal operation of the IoT device is not influenced by DoS that target the configuration interface.
(R10) However, a residual risk remains, since such DoS attacks will of course drain the device's battery by triggering operations at the SE.

VI. Conclusions and Future Work

In this paper, we present a novel automated authentication credential derivation process that is suited for personal as well as industrial used IoT devices. Instead of relying on users to change authentication credentials, configuration updates trigger the automated derivation of new authentication credentials. To increase the usability of our approach, we do not require users to remember these authentication credentials any more. We propose a system architecture that, besides managing configurations, also keeps track of a user's derived authentication credentials. To account for different usage scenario of devices, we present and discuss different configuration update and configuration-reset mechanisms, that we deem suitable for industrial or personal scenarios respectively. Thus, while increasing system security due to automatically triggered authentication credential updates, usability compared to traditional approaches is also improved. The runtime evaluation of our presented approach highlights that the resulting overhead due to using a modified SPAKE2 algorithm is in an acceptable range. The threat analysis then demonstrates, that 9 out of 10 threats can effectively be mitigated by our proposed approach. The only residual risk, DoS attacks that drain the device's battery, cannot be mitigated by any other known approach other than turning the device off.

As future work, we plan to investigate methods for initial key and configuration provisioning at the DM's facility that is capable of protecting this confidential information from the DM that is deploying the data on the device. This would allow customers to pre-configure devices such that the authentication credentials are only known to them to further improve the usefulness of our approach.

Acknowledgment

This work has been performed within the IoSense (http://iosense.eu) project. This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia. IoSense is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2016 and April 2019. More information: https://iktderzukunft.at/en/.

References

[1] R. H. Weber, "Internet of things - new security and privacy challenges," *Computer law & security review*, vol. 26, no. 1, pp. 23–30, 2010.

[2] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, 2014.

[3] H. Ning, H. Liu, and L. T. Yang, "Cyberentity Security in the Internet of Things," *Computer*, vol. 46, no. 4, pp. 46–53, 2013.

[4] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling CyberPhysical Systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[5] L. Da Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[6] M. Patton, E. Gross, R. Chinn, S. Forbis, L. Walker, and H. Chen, "Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT)," in *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*. IEEE, 2014, pp. 232–235.

[7] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security Analysis on Consumer and Industrial IoT Devices," in *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*. IEEE, 2016, pp. 519–524.

[8] E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[9] N. Cam-Winget, A.-R. Sadeghi, and Y. Jin, "Can IoT be Secured: Emerging Challenges in Connecting the Unconnected," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*. IEEE, 2016, pp. 1–6.

[10] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and Privacy Challenges in Industrial Internet of Things," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[11] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of Passwords and People: Measuring the Effect of Password-Composition Policies," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 2595–2604.

[12] W. Diffie and M. Hellman, "New Directions in Cryptography ," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[13] S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks ," in *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*. IEEE, 1992, pp. 72–84.

[14] M. Abdalla and D. Pointcheval, "Simple Password-Based Encrypted Key Exchange Protocols," in *CT-RSA, LNCS*, vol. 3376. Springer, pp. 191–208.

[15] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper Resistance Mechanisms for Secure Embedded Systems," in *VLSI Design, 2004. Proceedings. 17th International Conference on*. IEEE, 2004, pp. 605–611.

[16] A. Zankl, H. Seuschek, G. Irazoqui, and B. Gulmezoglu, "Side-Channel Attacks in the Internet of Things," *Solutions for Cyber-Physical Systems Ubiquity*, pp. 325–357, 2017.

[17] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[18] M. A. Jan, P. Nanda, X. He, Z. Tan, and R. P. Liu, "A Robust Authentication Scheme for Observing Resources in the Internet of Things Environment," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on*. IEEE, 2014, pp. 205–211.

[19] J. Xu, W.-T. Zhu, and D.-G. Feng, "An improved smart card based password authentication scheme with provable security," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 723–728, 2009.

[20] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[21] J. Liu, Y. Xiao, and C. P. Chen, "Authentication and Access Control in the Internet of Things," in *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. IEEE, 2012, pp. 588–592.

[22] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen, "Sensor Discovery and Configuration Framework for The Internet of Things Paradigm," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 94–99.

[23] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning Software-Defined IoT Cloud Systems," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. IEEE, 2014, pp. 288–295.

[24] T. Perumal, S. K. Datta, and C. Bonnet, "IoT Device Management Framework for Smart Home Scenarios," in *Consumer Electronics (GCCE), 2015 IEEE 4th Global Conference on*. IEEE, 2015, pp. 54–55.

[25] F. K. Santoso and N. C. Vun, "Securing IoT for Smart Home System," in *Consumer Electronics (ISCE), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 1–2.

[26] T. Ulz, T. Pieber, C. Steger, S. Haas, R. Matischek, and H. Bock, "Hardware-Secured Configuration and Two-Layer Attestation Architecture for Smart Sensors," in *Digital System Design (DSD), 2017 Euromicro Conference on*. IEEE, 2017, pp. 229–236.

[27] I. Chatzigiannakis, H. Hasemann, M. Karnstedt, O. Kleine, A. Kroller, M. Leggieri, D. Pfisterer, K. Romer, and C. Truong, "True Self-Configuration for the IoT," in *Internet of Things (IOT), 2012 3rd International Conference on the*. IEEE, 2012, pp. 9–15.

[28] S.-M. Kim, H.-S. Choi, and W.-S. Rhee, "IoT Home Gateway for Auto-Configuration and Management of MQTT Devices," in *Wireless Sensors (ICWiSe), 2015 IEEE Conference on*. IEEE, 2015, pp. 12–17.

[29] H.-L. Truong and S. Dustdar, "Principles for Engineering IoT Cloud Systems," *IEEE Cloud Computing*, vol. 2, no. 2, pp. 68–76, 2015.

[30] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "SEDA: Scalable Embedded Device Attestation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 964–975.

[31] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, and S. Zeitouni, "DARPA: Device Attestation Resilient to Physical Attacks," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 171–182.

[32] R. Schroeppel, H. Orman, S. OMalley, and O. Spatscheck, "Fast Key Exchange with Elliptic Curve Systems," *Advances in Cryptology-CRYPT0'95*, pp. 43–56, 1995.

[33] W. Ma, J. Campbell, D. Tran, and D. Kleeman, "Password Entropy and Password Quality," in *Network and System Security (NSS), 2010 4th International Conference on*. IEEE, 2010, pp. 583–587.

[34] B. Sunar, W. J. Martin, and D. R. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," *IEEE Transactions on computers*, vol. 56, no. 1, 2007.

[35] C.-W. Lin, J.-J. Shen, and M.-S. Hwang, "Security Enhancement for Optimal Strong-Password Authentication Protocol," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 2, pp. 7–12, 2003.

[36] M. Sabt, M. Achemlal, and A. Bouabdallah, "The Dual-Execution-Environment Approach: Analysis and Comparative Evaluation," in *IFIP International Information Security Conference*. Springer, 2015, pp. 557–570.

[37] T. Pieber, T. Ulz, C. Steger, and R. Matischek, "Hardware Secured, Password-based Authentication for Smart Sensors for the Industrial Internet of Things," in *International Conference on Network and System Security*. Springer, 2017, pp. 632–642.

[38] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat Modeling as a Basis for Security Requirement," in *Symposium on requirements engineering for information security (SREIS)*, vol. 2005. Citeseer, 2005, pp. 1–8.

[39] M. Howard and D. LeBlanc, *Writing Secure Code*. Microsoft Press, 2003.

Sensing Danger: Exploiting Sensors to Build Covert Channels

Thomas Ulz, Markus Feldbacher, Thomas Pieber and Christian Steger
Institute of Technical Informatics, Graz University of Technology, Graz, Austria
{*thomas.ulz, thomas.pieber, m.feldbacher, steger*}*@tugraz.at*

Keywords: Sensor, Exploit, Security, Side-Channel, Covert Channel.

Abstract: Recent incidents have shown that sensor-equipped devices can be used by adversaries to perform malicious activities, such as spying on end-users or for industrial espionage. In this paper, we present a novel attack scenario that uses unsecured embedded sensors to build covert channels that can be used to bypass security mechanisms and transfer information between isolated processes. We present covert channels that require read- and write-access for sensor registers as well as a covert channel that transfers data by just triggering sensor readings so that malicious behavior cannot be distinguished from normal sensor usage. For each presented covert channel we discuss the trade-off between data rate and the likelihood of being detected as well as potential countermeasures. The fastest covert channel we implemented achieves a data rate of 4844 bit/s while the stealthiest but slower covert channel cannot be distinguished from normal user behavior. To highlight the significance of these security issues, we used popular platforms, such as Linux and Android, to evaluate the presented covert channels. However, we do not make any assumption regarding the device's platform, and thus we believe that the presented exploits pose a significant security risk for any sensor-equipped device.

1 INTRODUCTION

Nowadays, sensors are embedded into nearly every device to improve the device's usefulness. Applications of such sensor-equipped devices are basically unlimited and include, for example, environmental monitoring (Srbinovska et al., 2015), healthcare applications (Nguyen et al., 2016), or industrial applications (Chi et al., 2014). Also, modern smartphones contain multiple embedded sensors that are used to improve user experience (Yu et al., 2015). Regardless of the application domain, embedded sensors are seen as an enabling technology for improved functionality such as context awareness (Perera et al., 2014). However, including embedded sensors into everyday objects also entails several security risks. The most addressed security issue regarding sensors is the privacy aspect of sensor data (Suo et al., 2012). Since sensors observe the environment, they sense private information, such as health care data (Yi et al., 2016) or industrial processes (Sadeghi et al., 2015). A loss of such private sensor data can lead to severe consequences that can even result in severe financial losses for a business if intellectual property or customer data is lost in a security breach. Therefore, the privacy of sensor data usually is considered to be of high importance. The second security issue related to sensors is the trustworthiness of sensor data (Suo et al., 2012). In so-called deception attacks (Kwon et al., 2013), an adversary influences a system's behavior by manipulating sensor data. If the manipulated sensor data is used to control a system or a process, the system could be physically damaged or even threaten human lives due to its malicious behavior (Derler et al., 2012). Therefore, the trustworthiness of sensor data also is considered to be of high importance. Finally, insufficient and too coarse permissions for accessing sensors also present security issues in sensors that need to be addressed (Shrivastava et al., 2017). However, such issues most often are associated with privacy concerns. In this paper, we are going to exploit insufficiently secured sensor interfaces to transfer data between two processes that are otherwise prevented from exchanging data. A so-called covert channel poses an immense security risk for systems since the security implications range from leaking private information to compromising a system so that its intended behavior is either manipulated or disabled. We present three different covert channels that provide a trade-off in covert channel data rate and the likelihood of such a covert channel being detected by a user or some software mechanism such as auditing sensor usage (Mirzamohammadi et al., 2017). The data is transferred by exploiting unprotected sensor registers in all three presented approaches. We do not claim that the list of covert channels presented in this paper is exhaustive. Instead, with this paper we want to bring attention to such security issues and highlight the importance of mitigating them.

Contributions. In brief, we make the following contributions in this paper. To the best of our knowledge, we are the first to present these concepts. We present three sensor-based covert channels that are enabled by unprotected registers in embedded sensors. These covert channels differ in the achievable data rate and the channel's likelihood of being detected. In addition, we present countermeasures to mitigate the presented covert channels. We also demonstrate a sensor-based covert channel that is based on exploiting a security weakness in Android's sensor management system. To facilitate the evaluation of sensors regarding exploitable vulnerabilities, we developed an easy-to-use modular and extendible framework. We provide this framework on GitHub[1].

Outline. The remainder of this paper is structured as follows. In Section 2, we are going to briefly introduce side-channels and covert channels, and categorize these attacks. We list other state-of-the-art covert channels in Section 3 and discuss their performance. After that, we define the underlying system-model we assume and discuss possible resulting threats in Section 4. In Section 5, we demonstrate our register-based covert channels and discuss potential countermeasures. Section 6 discusses covert channels based on exploiting Android's sensor manager. The framework we developed for evaluating sensor-based covert channels is then presented in Section 7. In Section 8, we evaluate the presented covert channels as well as our framework's functionality. This paper is then concluded with Section 9.

2 COVERT CHANNELS

The term covert channel was coined by Lampson (Lampson, 1973) in 1973 when he defined a covert channel as a communication channel that is *not intended* for information transfer at all. Usually, covert channels facilitate information transfer between processes that are otherwise prohibited from communicating with each other by the system. In order to build a covert channel, the data that needs to be transferred is embedded in events that are observable by other processes such as a processes's system load (Lampson, 1973). Such observable events are denoted as so-called *side-channels*.

[1]https://github.com/Grundplatte/SensIO

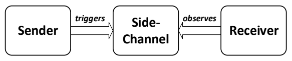


Figure 1: Basic concept of a covert channel.

Side-Channels can be exploited for either *active* or *passive* side-channel attacks. In active side-channel attacks, an attacker actively tampers with the device, thus requiring physical access (Genkin et al., 2016). In passive side-channel attacks, the effects that are caused by one process are observed by another process. This can be used to reveal confidential information such as cryptographic keys by monitoring processes for *unintentionally* leaking side-channel information, such as timing, power consumption, or electromagnetic emanation (Kim and Quisquater, 2007; Longo et al., 2015; Luo et al., 2015). If a process *intentionally* triggers such observable effects, data can be transferred by the process and received by another process, thus establishing a so-called *covert channel*.

Covert Channels in general comprise three entities, a *sender - receiver* pair, and the *side-channel* that is used to build the respective covert channel. Figure 1 illustrates a basic covert channel and the data flow between these entities. (i) The **sender** is in possession of data that it wants to transfer to the receiver. However, the system prevents the sender from using conventional methods, such as shared memory or sockets, to transfer its data. Therefore, the sender utilizes side-channel information that can be manipulated by the sender. The (ii) **side-channel** is used by the sender and receiver as a stealthy transport medium for their data transfer. (iii) The **receiver** needs to be capable of observing the side-channel's state changes. In addition, it must be synchronized with the sender so that the start and end of the transferred data stream can be correctly identified. In an ideal scenario, the receiver is also able to distinguish between state changes of the side-channel that are either caused by the sender or by normal system operation.

Depending on a side-channel's nature, different data rates can be achieved. The achievable speed depends on two factors. The first determining factor is the *component's speed*, i.e. sensor-based covert channels (Carrara and Adams, 2016) will be slower than covert channels based on components that are optimized for performance such as memory. The second determining factor is the *word size* that can be transferred. A covert channel that is capable of transmitting a multi-bit word per time unit will be faster than a covert channel that only can transfer a 1-bit word per time unit. A basic overview that compares the achievable covert channel data rates for the most common side-channels is shown in Figure 2. We also classified

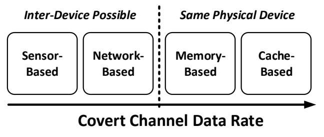


Figure 2: Classification of different covert channel types according to the covert channel data rate.

the covert channels based on their *scope*. The faster covert channels (memory- and cache-based) require the sender and receiver to reside on the same physical device so that both processes share the same memory or cache. In contrast to that, sensor- and network-based covert channels can transfer data between processes that are running either on the same physical device or on different devices as long as they can access the *same* shared medium.

3 RELATED WORK

In this section, we list related work for exploiting different system components.

3.1 Cache-Based Covert Channels

Modern processor- and system-architectures entail leaking side-channel information due to these systems being optimized for performance or energy efficiency (Wang and Lee, 2006). One of these side-channels that leak information is cache memory. Cache-based side-channels do not rely on weaknesses in the operating system (OS) or a virtual machine monitor, and thus, these attacks are considered to be highly practical (Liu et al., 2015). The side-channel that is exploited for cache-based covert channels is the timing difference between a cache *hit* and a cache *miss*. If a process is capable of intentionally causing cache hits or misses, data can be encoded in these events. A cache miss can be provoked by flushing all data from cache regions (Osvik et al., 2006; Yarom and Falkner, 2014; Gruss et al., 2016). A very fast and reliable cache-based covert channel that is capable of bit rates over 45 KByte/s with a bit error rate of 0% was presented in literature (Maurice et al., 2017).

3.2 Memory-Based Covert Channels

Since the memory in modern processors and systems is shared between cores, memory-based side-channels are used to reveal confidential information and to build covert channels (Zhang et al., 2012). One commonly used method for inter-process communication, shared memory, is usually prohibited by process isolation, such as sandboxes or virtual machines. However, side-channel information can be used to bypass these protection mechanisms, for example by exploiting memory deduplication (Xiao et al., 2013). Other side-channels exploit timing differences while locking the memory bus (Wu et al., 2011). A DRAM-based side-channel was presented (Pessl et al., 2016) for which the authors claimed raw bit rates of up to 2 Mbit/s while the bit error probability stayed below 1%. However, the authors did not state a bandwidth for 0% bit errors.

3.3 Network-Based Covert Channels

Exploiting network protocols to build network-based covert channels is one of the earliest known methods for stealthy data transfer. Network-based covert channels are used to bypass network protection mechanisms such as firewalls or virtual local area networks (VLANs) that otherwise are used to monitor or prevent unwanted data transfer (Zander et al., 2007a). To hide transferred data in network packets, various protocols at different network layers are exploited. On the network layer, information can be hidden in protocol headers such as in the 802.11 protocol's sequence control field (Frikha et al., 2008), or in the Received Signal Strength Indicator (RSSI) (Tuptuk and Hailes, 2015). On the Internet and Transport layer, many approaches use Transmission Control Protocol/Internet Protocol (TCP/IP) header fields to hide data in network packets (Ahsan and Kundur, 2002; Giffin et al., 2002; Zander et al., 2007b). On the application layer, various protocol fields can be used to hide information (Ameri and Johnson, 2017). Also, covert channels that work *independently* of any network protocol were presented (Cabuk et al., 2004; Ji et al., 2009).

3.4 Sensor-Based Covert Channels

Malicious use of sensors and their data traditionally involves spying on events or humans to reveal confidential information (Perrig et al., 2004). If an event (e.g. entering a password) triggers physical effects such as vibration that can be measured by a nearby sensor, sensor-based side-channels can be used for malicious activities (Aviv et al., 2012). Also other sensors such as ambient light sensors can be used to steal confidential information on a smartphone (Spreitzer, 2014). In the same manner, sensor-based covert channels can be built by triggering physical effects from one process and by measuring these effects from another process. For instance, covert channels based on the temperature of devices were pre-

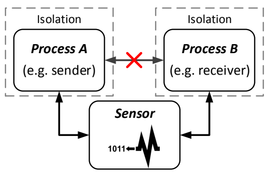


Figure 3: Underlying covert channel system-model.

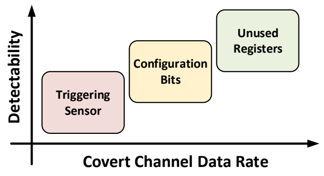


Figure 4: Trade-off between data rate and detectability of our presented covert channels.

sented (Brouchier et al., 2009; Guri et al., 2015). In contrast to approaches that require the modification of physically observable values, covert channels might also tamper with measured sensor values for stealthy data transfer (Tuptuk and Hailes, 2015).

4 SYSTEM- & THREAT-MODEL

In this section, we present the system model that is used to exploit embedded sensors and discuss the threat-model and potential impact of covert channels.

4.1 System-Model

To exploit security weaknesses in sensors for building sensor-based covert channels, we consider the system-model shown in Figure 3. This model is comprised of at least two potentially *isolated processes* and a *shared sensor*. The isolation between processes (e.g. sandboxes) prevents any direct data exchange between these processes. However, in our system-model both processes can access the *same* shared sensor. We do not make any assumption regarding the type of sensor that is present in this system.

4.2 Threat-Model

In our threat-model, we identify two scenarios that are enabled by transferring data over a covert channel. An isolated process might be able to send private data or receive instructions via this covert channel.

1. We assume that an isolated *process A* holds confidential information that an attacker wants to communicate to another *process B*. For instance, *process A* might monitor a video stream to detected movements in a security system. However, *process A* is prohibited from sharing the video stream. If *process A* is capable of transferring information stealthily to *process B* using a covert channel, data privacy is broken.
2. As second scenario, we assume that *process A* is capable of controlling some physical process such as a robot's actuators. To prevent malicious control actions, *process A* is isolated from the network. Instead, control actions are solely triggered by sensor data. However, an attacker might be able to send control commands to *process A* via *process B* and a covert channel.

To successfully establish a sensor-based covert channel between two processes, an attacker needs to be capable of executing modified code in both involved processes. We assume that an attacker is capable of running the required malicious code through any state-of-the-art attack such as code injection (Poeplau et al., 2014).

5 SENSOR REGISTER EXPLOITS

In this section, we present three exploits of embedded sensors that we use to build sensor-based covert channels. All three exploits are based on direct access to the sensor. That is, access to the sensor is not limited by any mechanism such as managed sensor access (Milette and Stroud, 2012). In all three approaches, sensor registers are used to transfer information between processes. The approaches differ in the achievable covert channel data rate and the likelihood of detecting such a covert channel (*detectability*). Also, the required effort for mitigating the different covert channels differs. A comparison of all three approaches regarding these three attributes is shown in Figure 4, where green indicates a covert channel easy to mitigate and red indicates a covert channel that is hard to mitigate.

5.1 Unused Registers

Embedded sensors usually contain unused registers that are either reserved or not required by the sensor's current mode of operation. Similar to network-based covert channels, these registers can be exploited for transferring data.

Reserved registers are registers that are neither used by the sensor to publish information (e.g.

status flags), nor to store data such as configuration parameters for the sensor. Sensors, such as the `HTS221` (STMicroelectronics, 2016) humidity and temperature sensor or the `LSM9DS1` (STMicroelectronics, 2015) magnetometer, accelerometer, and gyroscope, contain many such reserved registers. The reserved registers are listed in the respective data-sheets. Although the data-sheets often state that these registers *must not* be changed, they are often still read- *and* writeable.

Unused registers are registers that might be used in some sensor operation modes, but are unused in other modes. For instance, many sensors such as the `LSM9DS1` sensor allow to set thresholds (e.g. `INT_THS_*` registers) that are used to activate flags that indicate if the threshold is exceeded. However, if the threshold monitoring is disabled (in the `ACT_THS` register), the threshold registers are unused, and thus can be used for data transfer in a covert channel. Since the register that indicates which thresholds are monitored is readable, a malicious process easily can determine which threshold registers are unused.

5.1.1 Covert channel design

Unused registers facilitate a very simple covert channel design. Information can be written into a register by one process, while the other process reads the register and subsequently confirms reception of data by modifying the same register. In our design, we use the register's MSB as a flag to signal successful reception by the receiver. Therefore, not all bits of a register can be used for data transfer. Such a covert channel that is hiding information in registers is comparable to network-based covert channels that use reserved protocol fields or bits (Rezaei et al., 2013) for hiding transferred data. Depending on the sensor's register size, the amount of data that can be transferred in one transmission varies. Both `HTS221` and `LSM9DS1` contain 8 bit registers which allows 7 bits of data to be sent in each transmission.

5.1.2 Detectability

Reserved registers can be read and written to; however, writing values to these registers might have an impact on a sensor's correct functionality. Depending on the resulting impact of writing arbitrary values to such a register, covert channels that are based on writing and reading these registers could easily be detected. In comparison to that, a covert channel that is based on not required registers is harder to detect. As long as the register's value has no impact on the sensor's functionality, the covert channel does not influence the sensor's behavior.

5.1.3 Countermeasures

To mitigate a covert channel based on unused registers, various countermeasures can be used. (i) If write access to reserved registers is disabled, these registers cannot be exploited to build covert channels. (ii) Write access to unused registers must be disabled whenever a register is not required in the sensor's current mode of operation. Whenever the register content is required, write access for the respective register can be granted again by the sensor. (iii) Write-only configuration registers could mitigate such covert channels since the receiver would not be capable of reading transmitted data anymore. However, write-only registers also complicate updating the register's value, if the current value would be required first. For example, updating only a certain part of the register such as a threshold's exponent without modifying the remaining bits, requires bit-wise operations, such as `AND`, `OR`, and `XOR`, for registers.

5.2 Configuration Bits

Embedded sensors are configured using so-called configuration registers. In these registers, various different settings are often combined for efficiency reasons. Similarly to exploiting whole registers that are unused, certain bits of these configuration registers can often also be exploited.

Reserved bits of configuration registers can be used to transfer data in a covert channel. For example, bits [7:6] in the `HTS221` sensor's `AV_CONF` register are reserved bits that do not influence any configuration state. However, similar to unused registers, the data-sheet states that these bits *must not* be changed to not cause unwanted sensor behavior.

LSBs of configured values such as thresholds can be used to hide transferred data in a covered channel. This approach is similar to hiding data in the LSBs of header fields in network-based covert-channels. If chosen correctly, manipulating the LSBs of, e.g., a threshold value only has a negligible impact on the sensor's expected functionality.

Unused configuration bits can be present in configuration registers if the number of available options is smaller than the maximum number that can be represented by the respective part of the configuration register. For instance, in the `OPT3001` (Instruments, 2014) ambient light sensor's configuration register, 2 bits are reserved for a configuration parameter that has three available options. As shown in Table 1, if the MSB (Mode[1]) is set to '1', the LSB (Mode[0]) can be used to transfer data in a covert channel.

Table 1: `OPT3001` ambient light sensor modes of operation.

Mode	Mode[1]	Mode[0]
`shutdown`	0	0
`single-shot`	0	1
`continuous`	1	0
`continuous`	1	1

5.2.1 Covert channel design

A covert channel that is based on exploiting configuration bits can be based on the same principles as a covert channel that exploits unused registers. In a first step, the targeted configuration bits need to be determined. After that, these bits are used to transfer data in a covert channel by encoding data in these available bits. The recipient of data reads the respective bits and confirms if data is successfully read. Similarly to exploiting unused registers, one bit is required that is used as status flag for confirming that the receiver successfully read the transferred data. Therefore, such a covert channel requires at least *two* available bits. If only one bit is available, both sender and receiver must be synchronized by other measures such as a clock which slows down the covert channel. Compared to a covert channel that is able to exploit a whole unused register, a covert channel that is only able to utilize some bits of a register will provide lower covert channel data rates.

5.2.2 Detectability

Depending on the configuration bits that are exploited to build the covert channel, the detectability also varies. If reserved bits are used, correct sensor functionality might be influenced, and this may lead to easy detection of the covert channel. If bits that only have a minimal or no impact on the sensor's functionality are manipulated, the covert channel is harder to detect. However, toggling configuration parameters might cause sensors to restart their current measurement. Therefore, data transfer must be timed in order to minimize such detectable effects.

5.2.3 Countermeasures

To mitigate covert channels that are based on exploiting configuration bits, the following countermeasures can be implemented on a sensor. (i) Disabling write access to reserved bits mitigates misuse of these bits by covert channels. (ii) Write-only configuration registers mitigate covert channels that are based on exploiting configuration bits since the recipient of data is unable to read the register. However, similar to countermeasures for unused registers (Subsection 5.1.3), bit-wise operations, such as `AND`, `OR`, and `XOR`, will be required for registers.

5.3 Triggering Sensors

Both register exploit methods (unused registers and configuration bits) require read- and write-access to the same register. However, as briefly discussed in the respective countermeasure subsections, countermeasures to mitigate these exploits can easily be implemented in software or hardware. Nevertheless, exploiting embedded sensors via registers is still possible even if these countermeasures are implemented on a sensor. If there are *read-only* registers at a sensor that can be updated by certain *events*, and these events can be triggered by one process, a covert channel according to the definition shown in Figure 1 can still be built. For example, on most sensors status flags in registers are used to indicate a finished sensing process. If the sensor is not operated in a continuous sensing mode but in a single-shot mode, one process is capable of updating these status flags by triggering sensor readings. The status flags can then be used to encode information in various ways. For example, information can be encoded in timing differences, or, if multiple status bits exist, directly in these status bits.

Timing differences between sensor readings can be used to encode information. For example, a binary '1' could be transferred by requesting sensor readings with a time interval between readings of 100 ms (10 Hz). A binary '0' would then be transferred by using a different timing interval, for example, 50 ms (20 Hz). The receiver then needs to observe the status bit to get timing intervals and to infer the corresponding data. However, the drawback of such an approach is that sender and receiver need to be synchronized to guarantee precise timings. In addition, the receiver would need to poll status bits with a high frequency.

Directly encoding information in status bits is possible if there is more than one status bit that can be triggered by the sender. For example, sensors that are capable of sensing more than one physical property also contain multiple status flags to indicate a finished sensing process for each property. For example, the `HTS221` (STMicroelectronics, 2016) temperature and humidity sensor includes status registers for both physical properties. Using both registers, a 2-bit word can be encoded by triggering either no, one of both, or both sensors simultaneously. However, similarly to measuring timing differences, this approach would require precise synchronization between sender and receiver to distinguish a transferred '00' from the status flags default value that often is also set to '00'. To discard this requirement, data can be encoded by

Table 2: Status flags for two sensors and the available states for a 2-bit word and a 1-bit word respectively.

S1	S2	2-bit word	1-bit word
0	0	’00’ *and* no data	no data
0	1	’01’	’0’
1	0	’10’	’1’
1	1	’11’ *and* sensor ready	sensor ready

triggering one sensor to transmit a binary ’1’, and by triggering the other sensor to transmit a binary ’0’. Both mentioned approaches are compared in Table 2.

5.3.1 Covert channel design

Since covert channels based on encoding information in status bits do not require synchronization between processes, we consider this type of covert channel more practical. Therefore, we are going to discuss a covert channel based on transmitting a 1-bit word using the status flags of two distinct sensors. In its default setting, the sensor’s status flag is set to ’0’ and indicates that no sensor reading is ready at the moment. If a sensor reading is available, the respective status flag is set to ’1’. If the sensor’s measured value is read, the status flag is reset to ’0’ again. In our covert channel, the sender triggers both sensors. After the sensing process is completed, the sender reads one of the two sensor measurements to reset the respective status flag. Information is encoded as a 1-bit word according to Table 2. The receiver can observe the same status flags, and thus receive the transmitted information. The reception of data is confirmed by the receiver by resetting both status flags. Using the encoding shown in Table 2 in an example, the binary sequence ’11010’ would be encoded by resetting the status flags from the sensors `S1`, `S1`, `S2`, `S1`, `S2` respectively. Since the sensing process requires a certain amount of time, this covert channel’s data rate is lower compared to covert channels that directly write information into a sensor’s registers.

5.3.2 Detectability

Compared to directly writing into a sensor’s registers, a covert channel that is based on triggering sensors is harder to detect. Since only the sender triggers sensor readings, while the receiver is only observing status flags, no malicious activity might be noticed when monitoring sensor activities (Mirzamohammadi et al., 2017). The behavior that can be observed in such a case are two processes where one process is using sensors frequently, while the other process is checking the availability of these sensors. If the roles of sender and receiver are switched (Section 7.2), the covert channel’s behavior is comparable to two processes that alternately access the same sensors.

5.3.3 Countermeasures

To mitigate covert channels that are based on triggering sensors, more complex countermeasures are required in comparison to covert channels that exploit read- and write-able registers. In principle, any link between the event that can be triggered by a process and observable information needs to be removed. To mitigate all three discussed covert channels, a sensor management instance that encapsulates sensor access is required. As an example, the Android Sensor Manager (Milette and Stroud, 2012) only allows processes to register for sensor data they are interested in. The manager then determines the superset of all requested sensor configurations. Whenever a new sensor reading is available, all registered processes are notified via an interrupt. Therefore, such a managed approach would remove any status flag that indicates available sensor readings or exceeded thresholds, and thus mitigates covert channels based on such information.

6 MANAGED SENSOR EXPLOITS

Contrary to the previous sensor-based covert channel designs, Android uses a managed sensor approach (Milette and Stroud, 2012) where processes need to subscribe to a sensor manager to get sensor readings based on events. Thus, access to sensor registers as well as manually triggering sensor readings is not possible in Android. However, in this section we demonstrate two different approaches how we exploit the Android sensor manager to build sensor-based covert channels based on triggering sensors.

When registering a listener for any sensor that is supported by the respective hardware platform using Android’s built in sensor manager, the method `SensorManager.registerListener()` is used that takes the type of sensor as well as a *sampling period* as parameters. As stated in Android’s API documentation, this sampling period is only a *suggested delay* that might be altered by *other applications*. Typically, the sampling period can be set based on values predefined in Android (normal, UI, game, fastest) or specified arbitrarily. Thus, if one process registers a listener with a given sampling period which is then influenced by another process, data can be transferred between these processes using this covert channel.

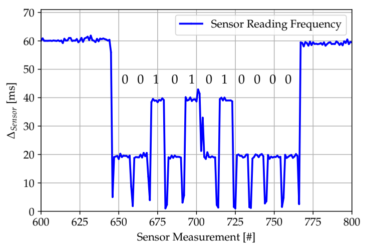


Figure 5: Different sensing intervals in Android.

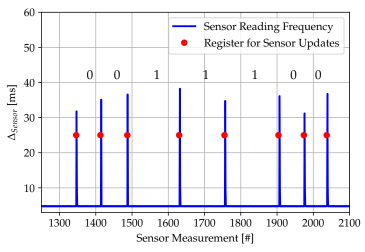


Figure 6: Registering sensor listeners in Android.

6.1 Covert Channel Design

Based on the observation that a process may influence the sampling period of other processes, we present two methods for covert channels in Android.

Frequency encoded. Android's sensor manager provides sensor measurements to all registered processes with the lowest sampling period specified by all registered processes. If the receiver registers a listener with a sampling period that is even lower than the current lowest sampling frequency, it signals to the sender that it now is ready to receive data. That is, by specifying the sensors's lowest sampling period, all processes now receive sensor measurements with the sampling period specified by the receiver. The sender now can encode information by registering with either the same sampling period as the receiver or with an ever lower sampling period, as shown in Figure 5. Thus, information is encoded by different sampling periods or sensing frequencies.

Outlier intervals. Whenever a new listener is registered using Android's sensor manager, one sensor measurement is provided with a time interval to the previous measurement that can clearly be detected as outlier. Depending on the hardware, we observed either outliers of too low (Figure 5) or too high (Figure 6) sampling periods. That is, even if a process already is registered for a sensor using the lowest possible sampling period supported by the hardware, information can still be transferred by registering new listeners to provoke such outliers. As shown in Figure 6, the interval between outliers can then be used to encode information that is transferred over the respective covert channel.

6.2 Detectability

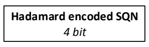
Covert channels based on registering sensor listeners cannot easily be detected since the switching of sampling periods due to other processes registering and un-registering listeners is expected behavior. A process that is minimized is expected to un-register its listener, while re-registering them if the process is activated again. However, if sensor access is audited (Han et al., 2017), malicious access patterns could be detected if the auditing tool is trained accordingly. In contrast, popular code analysis tools such as FlowDroid (Arzt et al., 2014) are currently not capable of detecting our presented covert channels. However, also these tools can be appropriately trained such that the presented covert channels can be found.

6.3 Countermeasures

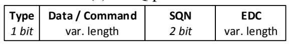
To mitigate the presented covert channels, changes to Android's sensor manager need to be implemented. If arbitrary sampling periods are banned and only predefined sampling periods are used, the sampling periods need to be defined such that they are multiples of each other. For example, if the predefined sampling period *fastest* is defined as 10 ms, *game* could be defined as 20 ms, *UI* as 80 ms, and *normal* as 160 ms. By doing so, the sensor internally can provide sensor measurements with the system's lowest specified sampling period, while each process receives sensor measurements with its specified sampling period only.

7 TEST FRAMEWORK

To facilitate easier vulnerability testing of embedded sensors, we present a modular covert channel framework that is structured into the following four abstraction layers. Hardware specific aspects are implemented in a *hardware abstraction* layer that is comprised of the following three sub-layers: (i) The lowest layer (*access abstraction*) implements access to embedded sensors through various technologies, such as I2C or SPI. (ii) The *sensor abstraction* layer implements sensor specific aspects such as register



(a) REQ packet.



(b) RES packet.

Figure 7: Packet structures of REQ and RES packets.

mappings. (iii) All sensor-based exploits that are presented in Section 5 are implemented in the *exploit abstraction* layer. (iv) Protocol specific functionality (Subsections 7.1 – 7.3) is implemented in the *covert channel abstraction* layer.

7.1 Error Detection and Correction

As other processes might also access sensors, a covert channel needs to be considered a *noisy channel*. However, an error-free data transmission through a noisy channel can be achieved if the data is sufficiently encoded by appropriate coding schemes (Shannon, 1948). In our covert channel framework, we use Error Correcting Codes (ECCs) as well as Error Detecting Codes (EDCs) to reliably transfer our messages, as will be discussed in Section 7.2.

7.2 Packet Structure and Flow

The packet structures of request (REQ) and response (RES) packets in our approach are shown in Figure 7.

REQ packet. The only information contained in a REQ packet is the Sequence Number (SQN) encoded by a Hadamard ECC. In general, a Hadamard ECC encodes a k bit message in a 2^k bit codeword. Due to the exponential relationship between payload size and codeword size, we use a 2 bit SQN, resulting in a total size of 4 bits. The Hadamard ECC is proven optimal for $k \leq 7$ (Bouyukliev and Jaffe, 2001). In case of transmission errors, the ECC ensures that information can be recovered and errors are detected as long as *less than half* of the bits are flipped.

RES packet. All RES packets start with a type field that specifies whether the message contains *data* or a *command*. Commands that are supported by our covert channel framework are shown in Table 3. Identical to REQ packets, RES packets also contain a 2 bit SQN. Due to their length, we do not use ECCs for RES packets. Instead, a Berger EDC (Berger, 1961) is used to detect transmission errors that need to be handled by the communication principle implemented in our framework. A k bit Berger code is capable of checking a maximum of $n = 2^k - 1$ bits information. Thus, the resulting data/command and EDC lengths can be derived from the packet's total size (Table 4).

Table 3: Supported commands.

Code	Description
0	Increment packet data size (Section 7.3)
1	Decrement packet data size (Section 7.3)
2	Stop data transmission
3	Reverse data direction

Table 4: Valid size options.

Data [bits]	EDC [bits]	Packet [bits]
5	3	11
13	4	20
29	5	37
61	6	70

To manage communication flow in our presented covert channels, we employ a request/response mechanism. A successful request/response cycle is comprised of the reception of a REQ packet by the sender and the reception of the RES packet by the receiver. Similar to HTTP, the actual data that is transferred is contained in the RES packet. Both, REQ and RES packets contain a SQN that is used to manage communication flow. Matching REQ and RES packets can be identified by their matching SQN. The receiver increases the SQN after successfully receiving the corresponding RES packet, as shown in Figure 8. By repeatedly sending the same SQN, receiver and sender can indicate that the expected packet was not successfully received. Retransmissions are caused in three scenarios:

1. When establishing the covert channel, the sender might not be ready to send the requested data and no matching RES packet is sent to the receiver's initial REQ packet. The receiver continuously transmits this initial packet until a covert channel is established, thereby synchronizing the states of receiver and sender.
2. A REQ packet can be lost due to a noisy data channel. The receiver repeatedly sends its REQ packet until a matching RES packet is received.
3. A RES packet can also be lost due to a noisy data channel. The sender repeatedly sends its RES packet until a REQ packet with an incremented SQN is received.

The commands supported by our covert channel framework (Table 3) also include two commands related to data flow. (i) To indicate the end of an ongoing data transfer, the sender sends a stop command to the receiver. (ii) The roles of sender and receiver can be reversed by sending the respective command. Thus, our covert channel also supports *bidirectional* communication.

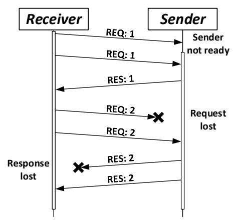


Figure 8: Data flow and handling of lost packets.

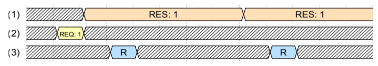
7.3 Adaptive Packet Length

Information is transferred by interacting with a sensor in all presented sensor-based covert channels. Thus, other processes that also interact with the *same* sensor might introduce bit errors into our covert channel. These bit errors can be detected by our approach due to RES packets containing an EDC. However, as shown in Figure 9(a), this leads to frequent retransmissions of large chunks of information.

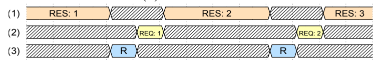
As already briefly discussed in Section 7.2, our covert channel supports dynamic packet sizes that can be used to minimize negative effects caused by bit errors. In addition, our framework also supports finding packet size templates for finding optimal sizes.

Finding Size Template. Although decreasing the size of RES packets may lead to less retransmissions, the overhead increases due to additional REQ packets (Figure 9(c)). Therefore, we propose analysing the potential covert channel before starting any data transmission. That is, the sender only observes the channel for sensing activity and tries to calculate an optimal packet size if other processes are accessing the sensor frequently (Figure 9(b)).

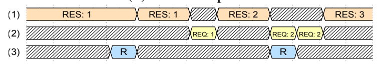
Dynamic Packet Size. If an optimal packet size cannot be determined, e.g., if another process is accessing a sensor infrequently, a covert channel might be unable to transfer any data. Therefore, we introduce a dynamic packet size approach. Whenever bit errors are detected using the EDC, it is assumed that another process is accessing the same sensor as the current covert channel. As a consequence, the packet size is reduced, which is indicated by sending the respective command (Table 3). Reducing the packet size results in a lower amount of retransmitted data as shown in Figure 9(c). Packet sizes are increased again if a certain amount of successfully transferred RES packets is exceeded.



(a) Static size.



(b) Size Template.



(c) Dynamic Size.

Figure 9: Different transfer modes. (1) sender, (2) receiver, and (3) another process reading the sensor (R).

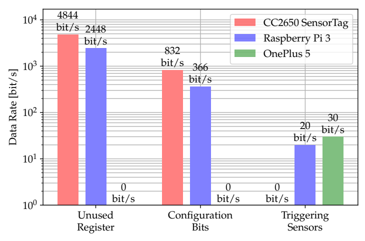


Figure 10: Evaluation of covert channel data rates.

8 EVALUATION

To evaluate the presented covert channels and the functionality of our covert channel framework, we use the following three hardware platforms and OSs:

1. *CC2650 SensorTag*; *TI-RTOS* 2.20
2. *Raspberry Pi 3, Sense HAT*; *Raspbian Strech*
3. *OnePlus 5* & *Android Emulator*; Android 8.0

8.1 Covert Channel Data Rates

To evaluate data rates and to validate the classification shown in Figure 4, we measured data rates on different platforms. The data rates we achieved in our evaluation are shown in Figure 10. As can be seen there, only the Raspberry Pi 3 with attached Sense HAT allows all three covert channels to be implemented. On the CC2650 SensorTag, only unused registers and configuration bits can be exploited. Contrary to that, the only side-channel that can be exploited in Android is triggering sensors. Both covert channels implemented on the CC2650 SensorTag running TI-RTOS achieve higher data rates compared to our implementations on the Raspberry Pi 3 due to the deterministic scheduling of TI-RTOS. Processes in TI-RTOS are scheduled alternately, compared to the indeterministic

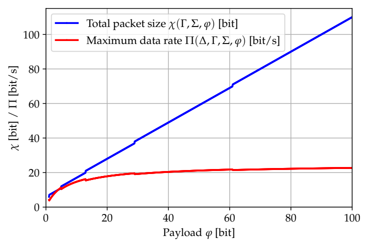
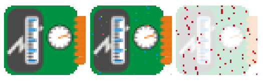


Figure 11: Covert channel data rate according to (1).



(a) Source. (b) Received. (c) Errors.

Figure 12: EDC and retransmissions disabled.

scheduling of Raspbian. Compared to that, the covert channel based on triggering sensors does not offer a very high data rate. However, the maximal achievable covert channel data rate Π depends on the time Δ required for a sensor reading. Π is then defined in dependence of the payload size φ, and the total round trip size $\chi(\varphi,\Gamma)$ that is comprised of all protocol fields (Γ), EDC ($\lceil \log_2(\Gamma+\varphi)\rceil$), and ECC ($\Sigma$).

$$\Pi(\Delta,\Gamma,\Sigma,\varphi) = \varphi/(\chi(\Gamma,\Sigma,\varphi)\Delta) \quad (1)$$
$$\chi(\Gamma,\Sigma,\varphi) = \Gamma+\Sigma+\varphi+\lceil \log_2(\Gamma+\varphi)\rceil \quad (2)$$

These two functions are evaluated and plotted for a range of different payload sizes in Figure 11. The covert channel data rate converges to a value of 24 bit/s on a Raspberry Pi 3 platform. Since we do not consider any delay caused by the bus or program execution in (1), we consider our achieved covert channel data rate of 20 bit/s on that platform, close to the theoretical maximum.

8.2 EDC Retransmission Functionality

To evaluate the implemented EDC and retransmission functionality (Section 7.1), as well as the overhead resulting from such retransmissions we simulated a noisy covert channel. As evaluation setting, a covert channel based on triggering sensors on a Raspberry Pi Sense HAT was used. Noise on these registers was introduced by running a process that accesses the same sensor as our covert channel every 10 s. We then transferred an image (4 kB) over this noisy channel. Figure 12 shows the original image as well as the received image that contains roughly 100 pixel errors if our EDC based retransmission of packets is disabled. Enabling these features results in an error-free image being transferred over our covert channel.

Overhead. Enabling EDC and retransmissions in a setting *without* noise introduces an transmission time overhead of roughly 6%. While the image with disabled EDC and retransmissions is transferred in 2998 s, transferring the image with EDC and retransmissions requires 3169 s. If noise is introduced by a process that is accessing the sensor every 10 s, transfer time increases to 3989 s in this setting as the noise requires 123 of 1062 packets to be retransmitted.

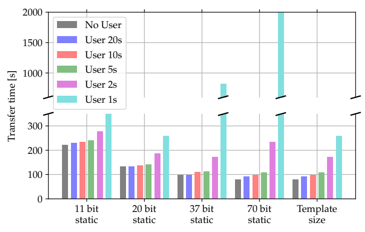


Figure 13: Evaluation of different packet sizes.

8.3 Static / Dynamic Packet Size

To evaluate the performance of different packet sizes that are supported by our framework (Table 4), we test these packet sizes with noise generated by different *noise profiles*. We test profiles with *no* interfering user actions as well as well as profiles where users access the same sensor that our covert channel is using. The evaluation results in Figure 13 highlight that there is no packet size that is capable of providing the fastest transfer time for each noise profile. However, if sensor access by the user is cyclic, our template method is able to provide best results for each noise profile.

If noise is introduced by a user that is accessing the sensor infrequently, or if no user was present during our template phase, the dynamic switching of packet sizes implemented in our covert channel framework is capable of ensuring a reliable and fast data transfer. Figure 14 evaluates such a scenario where a user starts to access the same sensor that our covert channel is using after the data transfer already started. In our evaluation, we compare two static packet sizes of 11 bits and 37 bits as well as our dynamic packet size mechanism that is configured to switch between packet sizes of 11 bits, 20 bits and 37 bits. We can identify the following three phases shown in Figure 14, where each data point represents the successful submission of one data packet or an

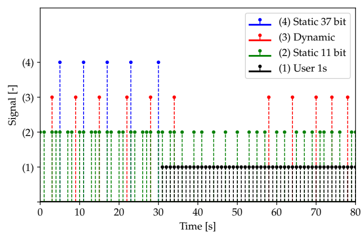


Figure 14: Evaluation of dynamic packet size.

interfering sensor access, respectively. (i) In the interval from 0 s to 30 s no interfering sensor access occurs. The dynamic packet size is set to 37 bits since this ensures the fastest data transfer. Both static variants transfer packets without errors. (ii) After 30 s, the user process starts to interfere, and thus packets with a size of 37 bits cannot be transferred any more due to the generated noise, as shown in Figure 9(a). Smaller packets are successfully transferred with a lower probability in this phase. Our dynamic packet size approach is now sending command messages to decrease the packet size. (iii) Starting at 58 s, the dynamic packet size approach can successfully transfer packets again due to decreasing the packet size.

8.4 Comparison to State-of-the-Art

The currently fastest data rates were reported for cache- and DRAM-based covert channels. The Flush+Flush (Gruss et al., 2016) cache-based covert channel is capable of data rates of up to 3.8 Mbit/s while the fastest DRAM-based covert channel achieves data rates of up to 2 Mbit/s. However, when comparing these covert channel data rates, it has to be considered that modern CPU caches and DRAMs are capable of achieving bandwidths in the range of 20 Gbit/s to 100 Gbit/s (Molka et al., 2015). That is, the presented covert channels use roughly 1% of the technology's possible bandwidth. Other memory based covert channels (Luo et al., 2015) provide a data rate of 747 bit/s and are slower than our fastest covert channel. The fastest reported network-based covert channel supports data rates of up to some kbit/s (Zander et al., 2007b); however, it also has to be considered for these covert channels that the network technology would provide a bandwidth of at least 100 Mbit/s. Other sensor-based covert channels reported data rates of 345 bit/s, which matches our second fastest covert channel. If considering the relatively low bandwidth provided by the I2C bus that is used in our evaluation, we claim that our presented covert channel implementations provide highly competitive data rates and bus utilization compared to the state-of-the-art.

9 CONCLUSION

In this paper, we presented novel exploits that target unsecured sensor interfaces. We use these exploits to demonstrate three different sensor-based covert channels that provide a trade-off between the achievable covert channel data rate and the likeliness of detecting the malicious behavior. Our fastest covert channel provides data rates of up to 4844 bit/s, while the slowest covert channel only provides a data rate of 20 bit/s but will not be distinguishable from normal user behavior. Our presented Android covert channels are not detected by state-of-the-art code analysis tools. We do not claim that the presented list of exploits is exhaustive, but rather believe that other issues can and will be found in current embedded sensors. To facilitate testing other platforms for security weaknesses, we provide our covert channel framework on GitHub. All countermeasures suggested in this paper can easily be implemented on embedded sensors. Therefore, this paper highlights the importance of implementing such countermeasures to mitigate sensor-based covert channels and to prevent future sensor-related security issues.

ACKNOWLEDGEMENTS

This work has been performed within the IoSense (http://iosense.eu) project. This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia. IoSense is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2016 and April 2019. More information: https://iktderzukunft.at/en/.

REFERENCES

Ahsan, K. and Kundur, D. (2002). Practical Data Hiding in TCP/IP. In *Proceedings of the Workshop on Multimedia Security at ACM Multimedia*, pages 1–8. ACM.

Ameri, A. and Johnson, D. (2017). Covert Channel over Network Time Protocol. In *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*, pages 62–65. ACM.

Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., and McDaniel, P. (2014). FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. *ACM SIGPLAN Notices - PLDI '14*, 49(6):259–269.

Aviv, A. J., Sapp, B., Blaze, M., and Smith, J. M. (2012). Practicality of Accelerometer Side Channels on Smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 41–50. ACM.

Berger, J. M. (1961). A Note on Error Detection Codes for Asymmetric Channels. *Information and Control*, 4(1):68–73.

Bouyukliev, I. and Jaffe, D. B. (2001). Optimal binary linear codes of dimension at most seven. *Discrete Mathematics*, 226(1-3):51–70.

Brouchier, J., Kean, T., Marsh, C., and Naccache, D. (2009). Temperature Attacks. *IEEE Security & Privacy*, 7(2):79–82.

Cabuk, S., Brodley, C. E., and Shields, C. (2004). IP Covert Timing Channels: Design and Detection. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 178–187. ACM.

Carrara, B. and Adams, C. (2016). Out-of-Band Covert ChannelsA Survey. *ACM Computing Surveys (CSUR)*, 49(2):23.

Chi, Q., Yan, H., Zhang, C., Pang, Z., and Da Xu, L. (2014). A Reconfigurable Smart Sensor Interface for Industrial WSN in IoT Environment. *IEEE transactions on industrial informatics*, 10(2):1417–1425.

Derler, P., Lee, E. A., and Vincentelli, A. S. (2012). Modeling CyberPhysical Systems. *Proceedings of the IEEE*, 100(1):13–28.

Frikha, L., Trabelsi, Z., and El-Hajj, W. (2008). Implementation of a Covert Channel in the 802.11 Header. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*, pages 594–599. IEEE.

Genkin, D., Pachmanov, L., Pipman, I., Tromer, E., and Yarom, Y. (2016). ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1626–1638. ACM.

Giffin, J., Greenstadt, R., Litwack, P., and Tibbetts, R. (2002). Covert Messaging through TCP Timestamps. In *International Workshop on Privacy Enhancing Technologies*, pages 194–208. Springer.

Gruss, D., Maurice, C., Wagner, K., and Mangard, S. (2016). Flush+Flush: A Fast and Stealthy Cache Attack. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 279–299. Springer.

Guri, M., Monitz, M., Mirski, Y., and Elovici, Y. (2015). BitWhisper: Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 276–289. IEEE.

Han, W., Cao, C., Chen, H., Li, D., Fang, Z., Xu, W., and Wang, X. S. (2017). senDroid: Auditing Sensor Access in Android System-wide. *IEEE Transactions on Dependable and Secure Computing*, (1):1–1.

Instruments, T. (2014). OPT3001 Ambient Light Sensor (ALS). [Online; accessed 10-March-2018].

Ji, L., Jiang, W., Dai, B., and Niu, X. (2009). A Novel Covert Channel Based on Length of Messages. In *Information Engineering and Electronic Commerce, 2009. IEEC'09. International Symposium on*, pages 551–554. IEEE.

Kim, C. H. and Quisquater, J.-J. (2007). How can we overcome both side channel analysis and fault attacks on RSA-CRT? In *Fault Diagnosis and Tolerance in Cryptography, 2007. FDTC 2007. Workshop on*, pages 21–29. IEEE.

Kwon, C., Liu, W., and Hwang, I. (2013). Security Analysis for Cyber-Physical Systems against Stealthy Deception Attacks. In *American Control Conference (ACC), 2013*, pages 3344–3349. IEEE.

Lampson, B. W. (1973). A Note on the Confinement Problem. *Communications of the ACM*, 16(10):613–615.

Liu, F., Yarom, Y., Ge, Q., Heiser, G., and Lee, R. B. (2015). Last-Level Cache Side-Channel Attacks are Practical. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 605–622. IEEE.

Longo, J., De Mulder, E., Page, D., and Tunstall, M. (2015). SoC It to EM: ElectroMagnetic Side-Channel Attacks on a Complex System-on-Chip. In *Intl. W. on CHES*, pages 620–640. Springer.

Luo, C., Fei, Y., Luo, P., Mukherjee, S., and Kaeli, D. (2015). Side-Channel Power Analysis of a GPU AES Implementation. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 281–288. IEEE.

Maurice, C., Weber, M., Schwarz, M., Giner, L., Gruss, D., Alberto Boano, C., Mangard, S., and Römer, K. (2017). Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium*, NDSS. The Internet Society.

Milette, G. and Stroud, A. (2012). *Professional Android Sensor Programming*. John Wiley & Sons.

Mirzamohammadi, S., Chen, J. A., Sani, A. A., Mehrotra, S., and Tsudik, G. (2017). Ditio: Trustworthy Auditing of Sensor Activities in Mobile & IoT Devices. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, page 14. ACM.

Molka, D., Hackenberg, D., Schöne, R., and Nagel, W. E. (2015). Cache Coherence Protocol and Memory Performance of the Intel Haswell-EP Architecture. In *Parallel Processing (ICPP), 2015 44th International Conference on*, pages 739–748. IEEE.

Nguyen, A., Alqurashi, R., Raghebi, Z., Banaei-Kashani, F., Halbower, A. C., and Vu, T. (2016). A Lightweight and Inexpensive In-ear Sensing System For Automatic Whole-night Sleep Stage Monitoring. In *Proceedings* *of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 230–244. ACM.

Osvik, D. A., Shamir, A., and Tromer, E. (2006). Cache Attacks and Countermeasures: The Case of AES. In *Cryptographers' Track at the RSA Conference*, pages 1–20. Springer.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454.

Perrig, A., Stankovic, J., and Wagner, D. (2004). Security in Wireless Sensor Networks. *Communications of the ACM*, 47(6):53–57.

Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. (2016). DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks. In *USENIX Security Symposium*, pages 565–581.

Poeplau, S., Fratantonio, Y., Bianchi, A., Kruegel, C., and Vigna, G. (2014). Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, NDSS, pages 23–26. The Internet Society.

Rezaei, F., Hempel, M., Peng, D., Qian, Y., and Sharif, H. (2013). Analysis and Evaluation of Covert Channels over LTE Advanced. In *WCNC Intl. Conf., 2013 IEEE*, pages 1903–1908. IEEE.

Sadeghi, A.-R., Wachsmann, C., and Waidner, M. (2015). Security and Privacy Challenges in Industrial Internet of Things. In *Proceedings of the 52Nd Annual Design Automation Conference*, pages 54:1–54:6. ACM.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423.

Shrivastava, A., Jain, P., Demetriou, S., Cox, P., and Kim, K.-H. (2017). CamForensics: Understanding Visual Privacy Leaks in the Wild. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, page 13. ACM.

Spreitzer, R. (2014). PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 51–62. ACM.

Srbinovska, M., Gavrovski, C., Dimcev, V., Krkoleva, A., and Borozan, V. (2015). Environmental parameters monitoring in precision agriculture using wireless sensor networks. *Journal of cleaner production*, 88.

STMicroelectronics (2015). LSM9DS1 iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer. [Online; accessed 10-March-2018].

STMicroelectronics (2016). HTS221: Capacitive digital sensor for relative humidity and temperature. [Online; accessed 10-March-2018].

Suo, H., Wan, J., Zou, C., and Liu, J. (2012). Security in the Internet of Things: A Review. In *Computer Science and Electronics Engineering (ICCSEE), 2012 Intl. Conf. on*, volume 3, pages 648–651. IEEE.

Tuptuk, N. and Hailes, S. (2015). Covert Channel Attacks in Pervasive Computing. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 236–242. IEEE.

Wang, Z. and Lee, R. B. (2006). Covert and Side Channels Due to Processor Architecture. In *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*, pages 473–482. IEEE.

Wu, J., Ding, L., Wang, Y., and Han, W. (2011). Identification and Evaluation of Sharing Memory Covert Timing Channel in Xen Virtual Machines. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 283–291. IEEE.

Xiao, J., Xu, Z., Huang, H., and Wang, H. (2013). Security Implications of Memory Deduplication in a Virtualized Environment. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–12. IEEE.

Yarom, Y. and Falkner, K. (2014). Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *USENIX Sec. Symp.*, pages 719–732.

Yi, X., Bouguettaya, A., Georgakopoulos, D., Song, A., and Willemson, J. (2016). Privacy Protection for Wireless Medical Sensor Data. *IEEE Transactions on Dependable and Secure Computing*, 13(3):369–380.

Yu, J., Zhao, J., Chen, Y., and Yang, J. (2015). Sensing Ambient Light for User Experience-Oriented Color Scheme Adaptation on Smartphone Displays. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 309–321. ACM.

Zander, S., Armitage, G., and Branch, P. (2007a). A Survey of Covert Channels and Countermeasures in Computer Network Protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57.

Zander, S., Armitage, G., and Branch, P. (2007b). An Empirical Evaluation of IP Time To Live Covert Channels. In *Networks, 2007. ICON 2007. 15th IEEE International Conference on*, pages 42–47. IEEE.

Zhang, Y., Juels, A., Reiter, M. K., and Ristenpart, T. (2012). Cross-VM Side Channels and Their Use to Extract Private Keys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 305–316. ACM.

Bibliography

[1] T. Suzuki, H. Tanaka, S. Minami, H. Yamada, and T. Miyata, "Wearable wireless vital monitoring technology for smart health care," in *2013 7th International Symposium on Medical Information and Communication Technology (ISMICT)*, pp. 1–4, IEEE, 2013.

[2] V. Ricquebourg, D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge, "The Smart Home Concept : our immediate future," in *2006 1st IEEE international conference on e-learning in industrial electronics*, pp. 23–28, IEEE, 2006.

[3] D. Lucke, C. Constantinescu, and E. Westkämper, "Smart Factory - A Step towards the Next Generation of Manufacturing," in *Manufacturing systems and technologies for the new frontier*, pp. 115–118, Springer, 2008.

[4] J. M. Hernández-Muñoz, J. B. Vercher, L. Muñoz, J. A. Galache, M. Presser, L. A. H. Gómez, and J. Pettersson, "Smart Cities at the Forefront of the Future Internet," in *The future internet assembly*, pp. 447–462, Springer, 2011.

[5] J. H. Huijsing, F. R. Riedijk, and G. van der Horn, "Developments in integrated smart sensors," *Sensors and Actuators A: Physical*, vol. 43, no. 1-3, pp. 276–288, 1994.

[6] G. Meijer, *Smart Sensor Systems*. John Wiley & Sons, 2008.

[7] N. V. Kirianaki, S. Y. Yurish, N. O. Shpak, and V. P. Deynega, *Data Acquisition and Signal Processing for Smart Sensors*. Wiley New York, 2002.

[8] S. Y. Yurish, "Sensors: Smart vs. Intelligent," *Sensors & Transducers*, vol. 114, no. 3, pp. 1–6, 2010.

[9] A. S. Morris and R. Langari, *Measurement and Instrumentation - Theory and Application*. Academic Press, 2012.

[10] J. Greenough, "The Internet of Things: Market Size, Share & Growth Forecasts." `https://www.businessinsider.de/how-the-internet-of-things-market-will-grow-2014-10?r=US&IR=T`, 2016. (Accessed on 04/02/2019).

[11] Statista, "Iot: number of connected devices worldwide 2012-2025." `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`, 2019. (Accessed on 04/02/2019).

[12] N. Tyler, "Demand for automotive sensors is booming." `http://www.newelectronics.co.uk/electronics-technology/automotive-sensors-market-is-booming/149323/`, 2016. (Accessed on 04/02/2019).

[13] V. C. Gungor and G. P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *IEEE Transactions on industrial electronics*, vol. 56, no. 10, pp. 4258–4265, 2009.

[14] A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[15] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, “DDoS in the IoT: Mirai and Other Botnets,” *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[16] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, K. Deepak, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, “Understanding the Mirai Botnet,” in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1093–1110, 2017.

[17] A. Cui and S. J. Stolfo, “A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan,” in *Proceedings of the 26th Annual Computer Security Applications Conference*, pp. 97–106, ACM, 2010.

[18] M. Patton, E. Gross, R. Chinn, S. Forbis, L. Walker, and H. Chen, “Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT),” in *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, pp. 232–235, IEEE, 2014.

[19] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “IoTPOT: Analysing the Rise of IoT Compromises,” in *Proceedings of the 9th {USENIX} Workshop on Offensive Technologies (WOOT)*, 2015.

[20] P. Derler, E. A. Lee, and A. S. Vincentelli, “Modeling Cyber–Physical Systems,” *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[21] P. Tague, M. Li, and R. Poovendran, “Mitigation of Control Channel Jamming under Node Capture Attacks,” *IEEE Transactions on Mobile Computing*, vol. 8, no. 9, pp. 1221–1234, 2009.

[22] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and Privacy Challenges in Industrial Internet of Things,” in *Design Automation Conference (DAC), 52nd ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2015.

[23] R. Spreitzer, “PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices,” in *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pp. 51–62, ACM, 2014.

[24] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici, “BitWhisper: Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations,” in *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pp. 276–289, IEEE, 2015.

[25] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, “Attack Models and Scenarios for Networked Control Systems,” in *Proceedings of the 1st international conference on High Confidence Networked Systems*, pp. 55–64, ACM, 2012.

[26] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, “Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors,” in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 881–896, 2015.

[27] A. Becher, Z. Benenson, and M. Dornseif, “Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks,” in *International Conference on Security in Pervasive Computing*, pp. 104–118, Springer, 2006.

[28] A. D. Wood and J. A. Stankovic, “Denial of Service in Sensor Networks,” *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[29] T. Zanni, G. Bell, D. Le, H. Shek, A. Tuteja, and D. Yong, “Risk or reward: What lurks within your iot?.” `https://assets.kpmg/content/dam/kpmg/xx/pdf/`

`2017/04/risk-or-reward-what-lurks-within-your-IoT.pdf`, 2017. (Accessed on 04/02/2019).

[30] T. Cox, S. Dombres, P. Prakash, and A. Camarillo, “Almost half of companies still can’t detect iot device breaches, reveals gemalto study.” `https://www.gemalto.com/press/pages/almost-half-of-companies-still-can-t-detect-iot-device-breaches-reveals-` `aspx`, 2019. (Accessed on 04/02/2019).

[31] M. Asplund and S. Nadjm-Tehrani, “Attitudes and Perceptions of IoT Security in Critical Societal Services,” *IEEE Access*, vol. 4, pp. 2130–2138, 2016.

[32] A. Höller, N. Druml, C. Kreiner, C. Steger, and T. Felicijan, “Hardware/Software Co-Design of Elliptic-Curve Cryptography for Resource-Constrained Applications,” in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2014.

[33] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad, “Proposed Embedded Security Framework for Internet of Things (IoT),” in *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, pp. 1–5, IEEE, 2011.

[34] R. Kainda, I. Flechais, and A. Roscoe, “Security and Usability: Analysis and Evaluation,” in *2010 International Conference on Availability, Reliability and Security*, pp. 275–282, IEEE, 2010.

[35] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, “State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures,” in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 76–87, IEEE, 2010.

[36] G. Elahi and E. Yu, “A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs,” in *International Conference on Conceptual Modeling*, pp. 375–390, Springer, 2007.

[37] M. G. Solomon and M. Chapple, *Information Security Illuminated.* Jones & Bartlett Publishers, 2009.

[38] B. W. Lampson, “A Note on the Confinement Problem,” *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[39] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, “ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1626–1638, ACM, 2016.

[40] C. H. Kim and J.-J. Quisquater, “How can we overcome both side channel analysis and fault attacks on RSA-CRT?,” in *Fault Diagnosis and Tolerance in Cryptography, 2007. FDTC 2007. Workshop on*, pp. 21–29, IEEE, 2007.

[41] C. Luo, Y. Fei, P. Luo, S. Mukherjee, and D. Kaeli, “Side-Channel Power Analysis of a GPU AES Implementation,” in *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pp. 281–288, IEEE, 2015.

[42] J. Longo, E. De Mulder, D. Page, and M. Tunstall, “SoC It to EM: ElectroMagnetic Side-Channel Attacks on a Complex System-on-Chip,” in *Intl. W. on CHES*, pp. 620–640,

Springer, 2015.

[43] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications," in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, vol. 2, pp. 583–587, IEEE, 2004.

[44] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully Attacking Masked AES Hardware Implementations," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 157–171, Springer, 2005.

[45] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper Resistance Mechanisms for Secure Embedded Systems," in *VLSI Design, 2004. Proceedings. 17th International Conference on*, pp. 605–611, IEEE, 2004.

[46] M. Sabt, M. Achemlal, and A. Bouabdallah, "The Dual-Execution-Environment Approach: Analysis and Comparative Evaluation," in *IFIP International Information Security Conference*, pp. 557–570, Springer, 2015.

[47] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using ARM TrustZone to Build a Trusted Language Runtime for Mobile Applications," in *ACM SIGARCH Computer Architecture News*, vol. 42, pp. 67–80, ACM, 2014.

[48] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[49] B. A. Forouzan, *Cryptography & Network Security*. McGraw-Hill, Inc., 2007.

[50] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Science & Business Media, 2013.

[51] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "An Efficient Hardware-Based Fault Diagnosis Scheme for AES: Performances and Cost," *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1165–1178, 2012.

[52] S. Mangard and K. Schramm, "Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 76–90, Springer, 2006.

[53] M. Bellare, R. Canetti, and H. Krawczyk, "Message Authentication using Hash Functions - The HMAC Construction," *RSA Laboratories' CryptoBytes*, vol. 2, no. 1, pp. 12–15, 1996.

[54] J. Jonsson, "On the Security of CTR + CBC-MAC," in *International Workshop on Selected Areas in Cryptography*, pp. 76–93, Springer, 2002.

[55] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 531–545, Springer, 2000.

[56] P. Rogaway, "Authenticated-Encryption with Associated-Data," in *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 98–107, ACM, 2002.

[57] R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," in *International Conference on the Theory and Applications*

of Cryptographic Techniques, pp. 453–474, Springer, 2001.

[58] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[59] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, "On the Security of TLS-DHE in the Standard Model," in *Advances in Cryptology–CRYPTO 2012*, pp. 273–293, Springer, 2012.

[60] S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks ," in *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, pp. 72–84, IEEE, 1992.

[61] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[62] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[63] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer Science & Business Media, 2009.

[64] S. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000.

[65] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla, "SWATT: SoftWare-based ATTestation for Embedded Devices," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pp. 272–282, IEEE, 2004.

[66] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, 2011.

[67] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed Software-based Attestation for Node Compromise Detection in Sensor Networks," in *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, pp. 219–230, IEEE, 2007.

[68] International Organization for Standardization/International Electrotechnical Commission and others, "ISO/IEC 18092 Information technology—Telecommunications and information exchange between systems—Near Field Communication—Interface and Protocol (NFCIP-1)," *ISO/IEC*, vol. 18092, 2004.

[69] International Organization for Standardization/International Electrotechnical Commission and others, "ISO/IEC 21481 Information technology—Telecommunications and information exchange between systems—Near Field Communication—Interface and Protocol (NFCIP-2)," *ISO/IEC*, vol. 21481, 2005.

[70] V. Coskun, B. Ozdenizci, and K. Ok, "A Survey on Near Field Communication (NFC) Technology," *Wireless personal communications*, vol. 71, no. 3, pp. 2259–2294, 2013.

[71] J. Ondrus and Y. Pigneur, "An Assessment of NFC for Future Mobile Payment Systems," in *International Conference on the Management of Mobile Business (ICMB 2007)*, pp. 43–43, IEEE, 2007.

[72] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann, "SmartTokens: Delegable Access Control with NFC-Enabled Smartphones," in *International Conference on Trust and*

Trustworthy Computing, pp. 219–238, Springer, 2012.

[73] R. Widmann, S. Grunberger, B. Stadlmann, and J. Langer, "System Integration of NFC Ticketing into an Existing Public Transport Infrastructure," in *2012 4th International Workshop on Near Field Communication*, pp. 13–18, IEEE, 2012.

[74] D. López-de Ipiña, J. I. Vazquez, and I. Jamardo, "Touch Computing: Simplifying Human to Environment Interaction through NFC Technology," *1as Jornadas Científicas sobre RFID*, vol. 21, 2007.

[75] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[76] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," in *Workshop on RFID security*, pp. 12–14, 2006.

[77] S. Plósz, A. Farshad, M. Tauber, C. Lesjak, T. Ruprechter, and N. Pereira, "Security Vulnerabilities and Risks in Industrial Usage of Wireless Communication," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pp. 1–8, IEEE, 2014.

[78] G. Van Damme and K. Wouters, "Practical Experiences with NFC Security on mobile Phones," *Proceedings of the RFIDSec*, vol. 9, p. 27, 2009.

[79] E. Byres and J. Lowe, "The Myths and Facts behind Cyber Security Risks for Industrial Control Systems," in *Proceedings of the VDE Kongress*, vol. 116, pp. 213–218, 2004.

[80] A. A. Cárdenas, S. Amin, and S. Sastry, "Secure Control: Towards Survivable Cyber-Physical Systems," in *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on*, pp. 495–500, IEEE, 2008.

[81] A. A. Cárdenas, S. Amin, and S. Sastry, "Research Challenges for the Security of Control Systems," in *HotSec*, 2008.

[82] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless Sensor Network Security: A Survey," *Security in Distributed, Grid, Mobile, and Pervasive Computing*, vol. 1, p. 367, 2007.

[83] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security Analysis on Consumer and Industrial IoT Devices," in *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*, pp. 519–524, IEEE, 2016.

[84] E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[85] N. Cam-Winget, A.-R. Sadeghi, and Y. Jin, "Can IoT be Secured: Emerging Challenges in Connecting the Unconnected," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2016.

[86] H. Chan and A. Perrig, "Security and Privacy in Sensor Networks," *Computer*, no. 10, pp. 103–105, 2003.

[87] M. Al Ameen, J. Liu, and K. Kwak, "Security and Privacy Issues in Wireless Sensor Networks for Healthcare Applications," *Journal of Medical Systems*, vol. 36, no. 1, pp. 93–101, 2012.

[88] V. Subramanian, S. Uluagac, H. Cam, and R. Beyah, "Examining the Characteristics and Implications of Sensor Side Channels," in *2013 IEEE International Conference on Com-*

munications (ICC), pp. 2205–2210, IEEE, 2013.

[89] N. Tuptuk and S. Hailes, “Covert Channel Attacks in Pervasive Computing,” in *2015 IEEE international conference on pervasive computing and communications (PerCom)*, pp. 236–242, IEEE, 2015.

[90] C. Kwon, W. Liu, and I. Hwang, “Security Analysis for Cyber-Physical Systems against Stealthy Deception Attacks,” in *2013 American Control Conference*, pp. 3344–3349, IEEE, 2013.

[91] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, “Limiting the Impact of Stealthy Attacks on Industrial Control Systems,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1092–1105, ACM, 2016.

[92] R. M. Góes, E. Kang, R. Kwong, and S. Lafortune, “Stealthy Deception Attacks for Cyber-Physical Systems,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 4224–4230, IEEE, 2017.

[93] R. Langner, “Stuxnet: Dissecting a Cyberwarfare Weapon,” *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[94] S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen, “Stealthy Deception Attacks on Water SCADA Systems,” in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 161–170, ACM, 2010.

[95] S. Amin, X. Litrico, S. Sastry, and A. M. Bayen, “Cyber Security of Water SCADA Systems—Part I: Analysis and Experimentation of Stealthy Deception Attacks,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1963–1970, 2013.

[96] A. Teixeira, G. Dán, H. Sandberg, and K. H. Johansson, “A Cyber Security Study of a SCADA Energy Management System: Stealthy Deception Attacks on the State Estimator,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11271–11277, 2011.

[97] R. B. Bobba, K. M. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T. J. Overbye, “Detecting False Data Injection Attacks on DC State Estimation,” in *Preprints of the First Workshop on Secure Control Systems, CPSWEEK*, vol. 2010, 2010.

[98] G. Dan and H. Sandberg, “Stealth Attacks and Protection Schemes for State Estimators in Power Systems,” in *2010 First IEEE International Conference on Smart Grid Communications*, pp. 214–219, IEEE, 2010.

[99] D. Ding, Z. Wang, D. W. Ho, and G. Wei, “Distributed recursive filtering for stochastic systems under uniform quantizations and deception attacks through sensor networks,” *Automatica*, vol. 78, pp. 231–240, 2017.

[100] L. Ma, Z. Wang, Q.-L. Han, and H.-K. Lam, “Variance-Constrained Distributed Filtering for Time-Varying Systems With Multiplicative Noises and Deception Attacks Over Sensor Networks,” *IEEE Sensors Journal*, vol. 17, no. 7, pp. 2279–2288, 2017.

[101] Y. Mo and B. Sinopoli, “Secure Control Against Replay Attacks,” in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 911–918, IEEE, 2009.

[102] R. A. Gupta and M.-Y. Chow, “Performance Assessment and Compensation for Secure Networked Control Systems ,” in *2008 34th Annual Conference of IEEE Industrial Electronics*,

pp. 2929–2934, IEEE, 2008.

[103] Z.-H. Pang and G.-P. Liu, "Design and Implementation of Secure Networked Predictive Control Systems Under Deception Attacks," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1334–1342, 2012.

[104] H. Zhang, Y. Shu, P. Cheng, and J. Chen, "Privacy and Performance Trade-off in Cyber-Physical Systems," *IEEE Network*, vol. 30, no. 2, pp. 62–66, 2016.

[105] M. Chanson, A. Bogner, D. Bilgeri, E. Fleisch, and F. Wortmann, "Privacy-Preserving Data Certification in the Internet of Things: Leveraging Blockchain Technology to Protect Sensor Data," *Journal of the Association for Information Systems*, 2019.

[106] L. Tam, M. Glassman, and M. Vandenwauver, "The psychology of password management: a tradeoff between security and convenience," *Behaviour & Information Technology*, vol. 29, no. 3, pp. 233–244, 2010.

[107] M. L. Das, "Two-Factor User Authentication in Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1086–1090, 2009.

[108] G. Yang, D. S. Wong, H. Wang, and X. Deng, "Two-factor mutual authentication based on smart cards and passwords," *Journal of Computer and System Sciences*, vol. 74, no. 7, pp. 1160–1172, 2008.

[109] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous Two-Factor Authentication in Distributed Systems: Certain Goals Are Beyond Attainment," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 428–442, 2015.

[110] B. Vaidya, D. Makrakis, and H. T. Mouftah, "Improved Two-factor User Authentication in Wireless Sensor Networks ," in *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 600–606, IEEE, 2010.

[111] O. Delgado-Mohatar, A. Fúster-Sabater, and J. M. Sierra, "A light-weight authentication scheme for wireless sensor networks," *Ad Hoc Networks*, vol. 9, no. 5, pp. 727–735, 2011.

[112] A. K. Das, "A secure and effective biometric-based user authentication scheme for wireless sensor networks using smart card and fuzzy extractor," *International Journal of Communication Systems*, vol. 30, no. 1, p. e2933, 2017.

[113] H.-S. Choi, B. Lee, and S. Yoon, "Biometric Authentication Using Noisy Electrocardiograms Acquired by Mobile Sensors," *IEEE Access*, vol. 4, pp. 1266–1273, 2016.

[114] D. Gafurov, K. Helkala, and T. Søndrol, "Biometric Gait Authentication Using Accelerometer Sensor," *JCP*, vol. 1, no. 7, pp. 51–59, 2006.

[115] F. Okumura, A. Kubota, Y. Hatori, K. Matsuo, M. Hashimoto, and A. Koike, "A Study on Biometric Authentication based on Arm Sweep Action with Acceleration Sensor," in *2006 International Symposium on Intelligent Signal Processing and Communications*, pp. 219–222, IEEE, 2006.

[116] Z. Benenson, N. Gedicke, and O. Raivio, "Realizing Robust User Authentication in Sensor Networks," *Real-World Wireless Sensor Networks (REALWSN)*, vol. 14, p. 52, 2005.

[117] M. H. Almeshekah, C. N. Gutierrez, M. J. Atallah, and E. H. Spafford, "ErsatzPasswords: Ending Password Cracking and Detecting Password Leakage," in *Proceedings of the 31st Annual Computer Security Applications Conference*, pp. 311–320, ACM, 2015.

[118] Z. Wang and R. B. Lee, "Covert and Side Channels Due to Processor Architecture," in *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*, pp. 473–482, IEEE, 2006.

[119] Y. Yarom and K. Falkner, "Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-Channel Attack," in *USENIX Security Symposium*, pp. 719–732, 2014.

[120] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+Flush: A Fast and Stealthy Cache Attack," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 279–299, Springer, 2016.

[121] C. Maurice, M. Weber, M. Schwarz, L. Giner, D. Gruss, C. Alberto Boano, S. Mangard, and K. Römer, "Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium*, NDSS, The Internet Society, 2017.

[122] J. Xiao, Z. Xu, H. Huang, and H. Wang, "Security Implications of Memory Deduplication in a Virtualized Environment," in *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pp. 1–12, IEEE, 2013.

[123] J. Wu, L. Ding, Y. Wang, and W. Han, "Identification and Evaluation of Sharing Memory Covert Timing Channel in Xen Virtual Machines," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 283–291, IEEE, 2011.

[124] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks.," in *USENIX Security Symposium*, pp. 565–581, 2016.

[125] S. Zander, G. Armitage, and P. Branch, "A Survey of Covert Channels and Countermeasures in Computer Network Protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 44–57, 2007.

[126] L. Frikha, Z. Trabelsi, and W. El-Hajj, "Implementation of a Covert Channel in the 802.11 Header," in *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*, pp. 594–599, IEEE, 2008.

[127] K. Ahsan and D. Kundur, "Practical Data Hiding in TCP/IP," in *Proceedings of the Workshop on Multimedia Security at ACM Multimedia*, pp. 1–8, ACM, 2002.

[128] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert Messaging through TCP Timestamps," in *International Workshop on Privacy Enhancing Technologies*, pp. 194–208, Springer, 2002.

[129] S. Zander, G. Armitage, and P. Branch, "An Empirical Evaluation of IP Time To Live Covert Channels," in *Networks, 2007. ICON 2007. 15th IEEE International Conference on*, pp. 42–47, IEEE, 2007.

[130] S. Cabuk, C. E. Brodley, and C. Shields, "IP Covert Timing Channels: Design and Detection," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pp. 178–187, ACM, 2004.

[131] L. Ji, W. Jiang, B. Dai, and X. Niu, "A Novel Covert Channel Based on Length of Messages," in *Information Engineering and Electronic Commerce, 2009. IEEC'09. International Symposium on*, pp. 551–554, IEEE, 2009.

[132] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of Accelerometer Side Channels on Smartphones," in *Proceedings of the 28th Annual Computer Security Applications*

Conference, pp. 41–50, ACM, 2012.

[133] J. Brouchier, T. Kean, C. Marsh, and D. Naccache, "Temperature Attacks," *IEEE Security & Privacy*, vol. 7, no. 2, pp. 79–82, 2009.

[134] R. Schlegel, K. Zhang, X.-y. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium*, NDSS, pp. 17–33, The Internet Society, 2011.

[135] L. Deshotels, "Inaudible Sound as a Covert Channel in Mobile Devices," in *Proceedings of the 8th {USENIX} Workshop on Offensive Technologies (WOOT)*, USENIX Association, 2014.

[136] E. Novak, Y. Tang, Z. Hao, Q. Li, and Y. Zhang, "Physical Media Covert Channels on Smart Mobile Devices," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 367–378, ACM, 2015.

[137] A. Al-Haiqi, M. Ismail, and R. Nordin, "A New Sensors-Based Covert Channel on Android," *The Scientific World Journal*, vol. 2014, 2014.

[138] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser, "Covert Channels Using Mobile Device's Magnetic Field Sensors," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 525–532, IEEE, 2016.

[139] K. Kostiainen, J.-E. Ekberg, N. Asokan, and A. Rantala, "On-board Credentials with Open Provisioning," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pp. 104–115, ACM, 2009.

[140] S. Syed and M. Ussenaiah, "The Rise of Bring Your Own Encryption (BYOE) for Secure Data Storage in Cloud Databases ," in *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on*, pp. 1463–1468, IEEE, 2015.

[141] R. Steffen, J. Preissinger, T. Schöllermann, A. Müller, and I. Schnabel, "Near Field Communication (NFC) in an Automotive Environment," in *International Workshop on Near Field Communication*, pp. 15–20, 2010.

[142] J. Suomalainen, "Smartphone Assisted Security Pairings for the Internet of Things," in *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*, pp. 1–5, IEEE, 2014.

[143] M. Miettinen, N. Asokan, T. D. Nguyen, A.-R. Sadeghi, and M. Sobhani, "Context-Based Zero-Interaction Pairing and Key Evolution for Advanced Personal Devices," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 880–891, ACM, 2014.

[144] R. Mayrhofer and H. Gellersen, "Shake Well Before Use: Intuitive and Secure Pairing of Mobile Devices," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 792–806, 2009.

[145] I. Ahmed, Y. Ye, S. Bhattacharya, N. Asokan, G. Jacucci, P. Nurmi, and S. Tarkoma, "Checksum Gestures: Continuous Gestures as an Out-of-Band Channel for Secure Pairing," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 391–401, ACM, 2015.

[146] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity Based IoT Device Authentication,"

in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, 2017.

[147] P. Urien and C. Kiennert, "A New Key Delivering Platform Based on NFC Enabled Android Phone and Dual Interfaces EAP-TLS Contactless Smartcards," in *International Conference on Mobile Computing, Applications, and Services*, pp. 387–394, Springer, 2011.

[148] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, "A comparative study of secure device pairing methods," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 734–749, 2009.

[149] E. Uzun, K. Karvonen, and N. Asokan, "Usability Analysis of Secure Pairing Methods," in *International Conference on Financial Cryptography and Data Security*, pp. 307–324, Springer, 2007.

[150] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pp. 35–49, Springer, 2010.

[151] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies," *IEEE Transactions on Mobile Computing*, vol. 3, pp. 272–285, 2004.

[152] H. He, Z. Zhu, and E. Makinen, "A Neural Network Model to Minimize the Connected Dominating Set for Self-Configuration of Wireless Sensor Networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 6, pp. 973–982, 2009.

[153] A. Fritze, U. Mönks, and V. Lohweg, "A Concept for Self-Configuration of Adaptive Sensor and Information Fusion Systems," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4, IEEE, 2016.

[154] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning Software-Defined IoT Cloud Systems," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pp. 288–295, IEEE, 2014.

[155] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen, "Sensor Discovery and Configuration Framework for The Internet of Things Paradigm," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pp. 94–99, IEEE, 2014.

[156] V. Alimi and M. Pasquet, "Post-distribution provisioning and personalization of a payment application on a UICC-based Secure Element," in *2009 International Conference on Availability, Reliability and Security*, pp. 701–705, IEEE, 2009.

[157] D. Wu, M. J. Hussain, S. Li, and L. Lu, "R2: Over-the-Air Reprogramming on Computational RFIDs," in *2016 IEEE International Conference on RFID (RFID)*, pp. 1–8, IEEE, 2016.

[158] J. Haase, D. Meyer, M. Eckert, and B. Klauer, "Wireless sensor/actuator device configuration by NFC," in *2016 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1336–1340, IEEE, 2016.

[159] M. Roland and J. Langer, "Digital Signature Records for the NFC Data Exchange Format," in *2010 Second International Workshop on Near Field Communication*, pp. 71–76, IEEE, 2010.

[160] M. Roland, J. Langer, and J. Scharinger, "Security Vulnerabilities of the NDEF Signature Record Type," in *2011 Third International Workshop on Near Field Communication*, pp. 65–70, IEEE, 2011.

[161] H. Eun, H. Lee, and H. Oh, “Conditional privacy preserving security protocol for nfc applications,” *IEEE Transactions on Consumer Electronics*, vol. 59, no. 1, pp. 153–160, 2013.

[162] V. Odelu, A. K. Das, and A. Goswami, “SEAP: Secure and Efficient Authentication Protocol for NFC Applications Using Pseudonyms,” *IEEE Transactions on Consumer Electronics*, vol. 62, no. 1, pp. 30–38, 2016.

[163] K. Toyoda and I. Sasase, “Secret Sharing Based Unidirectional Key Distribution with Dummy Tags in Gen2v2 RFID-enabled Supply Chains,” in *RFID (RFID), 2015 IEEE International Conference on*, pp. 63–69, IEEE, 2015.

[164] Q. Li, Z. Sun, J. Huang, S. Liu, J. Wang, N. Yan, L. Wang, and H. Min, “Secure UHF-RFID Tag for Vehicular Traffic Management System,” in *RFID (RFID), 2017 IEEE International Conference on*, pp. 26–29, IEEE, 2017.

[165] P. Urien and S. Piramuthu, “LLCPS and SISO: A TLS-Based Framework with RFID for NFC P2P Retail Transaction Processing,” in *RFID (RFID), 2013 IEEE International Conference on*, pp. 152–159, IEEE, 2013.

[166] K. S. Kadambi, J. Li, and A. H. Karp, “Near-Field Communication-Based Secure Mobile Payment Service,” in *Proceedings of the 11th international Conference on Electronic Commerce*, pp. 142–151, ACM, 2009.

[167] U. B. Ceipidor, C. M. Medaglia, A. Marino, S. Sposato, and A. Moroni, “KerNeeS: A protocol for mutual authentication between NFC phones and POS terminals for secure payment transactions,” in *Information Security and Cryptology (ISCISC), 2012 9th International ISC Conference on*, pp. 115–120, IEEE, 2012.

[168] T. Halevi, D. Ma, N. Saxena, and T. Xiang, “Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data,” in *European Symposium on Research in Computer Security*, pp. 379–396, Springer, 2012.

[169] S. Duünebeil, F. Kobler, P. Koene, J. M. Leimeister, and H. Krcmar, “Encrypted NFC emergency tags based on the German Telematics Infrastructure,” in *2011 Third International Workshop on Near Field Communication*, pp. 50–55, IEEE, 2011.

[170] D. Sethia, D. Gupta, T. Mittal, U. Arora, and H. Saran, “NFC Based Secure Mobile Healthcare System,” in *Communication Systems and Networks (COMSNETS), 6th International Conference on*, pp. 1–6, IEEE, 2014.

[171] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi, “Smart Keys for Cyber-Cars: Secure Smartphone-based NFC-enabled Car Immobilizer,” in *Proceedings of the third ACM Conference on Data and Application Security and Privacy*, pp. 233–242, ACM, 2013.

[172] T. Ulz, T. Pieber, C. Steger, R. Matischek, and H. Bock, “Towards Trustworthy Data in Networked Control Systems: A Hardware-Based Approach,” in *Emerging Technologies and Factory Automation (ETFA), 22nd IEEE International Conference on*, pp. 1–8, IEEE, 2017.

[173] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Arzén, “How Does Control Timing Affect Performance? Analysis and Simulation of Timing using Jitterbug and TrueTime,” *IEEE control systems*, vol. 23, no. 3, pp. 16–30, 2003.

[174] H. Li, Y. Luo, and Y. Chen, “A Fractional Order Proportional and Derivative (FOPD) Mo-

tion Controller: Tuning Rule and Experiments," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 516–520, 2009.

[175] W. Zeng and M.-Y. Chow, "Optimal Tradeoff Between Performance and Security in Networked Control Systems Based on Coevolutionary Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 7, pp. 3016–3025, 2011.

[176] K. Kogiso and T. Fujita, "Cyber-Security Enhancement of Networked Control Systems Using Homomorphic Encryption," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 6836–6843, IEEE, 2015.

[177] X. He, M.-O. Pun, and C.-C. J. Kuo, "Secure and Efficient Cryptosystem for Smart Grid Using Homomorphic Encryption," in *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*, pp. 1–8, IEEE, 2012.

[178] Y. Monnet, M. Renaudin, and R. Leveugle, "Designing Resistant Circuits against Malicious Faults Injection Using Asynchronous Logic," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1104–1115, 2006.

[179] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 19–35, 2016.

[180] D. Gligoroski, S. J. Knapskog, and S. Andova, "Cryptcoding-Encryption and Error-Correction Coding in a Single Step.," in *Security and Management*, pp. 145–151, Citeseer, 2006.

[181] H. Kaneko and E. Fujiwara, "Joint Source-Cryptographic-Channel Coding Based on Linear Block Codes," in *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pp. 158–167, Springer, 2007.

[182] Q. Chai and G. Gong, "Differential Cryptanalysis of Two Joint Encryption and Error Correction Schemes," in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, pp. 1–6, IEEE, 2011.

[183] C. Berrou and A. Glavieux, "Turbo Codes," *Encyclopedia of Telecommunications*, 2003.

[184] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes (1)," in *Communications, 1993. ICC'93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2, pp. 1064–1070, IEEE, 1993.

[185] M. A. Jordan and R. A. Nichols, "The Effects of Channel Characteristics on Turbo Code Performance," in *Military Communications Conference, 1996. MILCOM'96, Conference Proceedings, IEEE*, vol. 1, pp. 17–21, IEEE, 1996.

[186] T. Ulz, M. Feldbacher, T. Pieber, and C. Steger, "Sensing Danger: Exploiting Sensors to Build Covert Channels," in *Information Systems Security and Privacy (ICISSP), Proceedings of the 5th International Conference on*, pp. 100–113, INSTICC, SciTePress, 2019.

[187] G. Milette and A. Stroud, *Professional Android Sensor Programming*. John Wiley & Sons, 2012.

[188] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps," *ACM SIGPLAN Notices - PLDI '14*, vol. 49,

no. 6, pp. 259–269, 2014.

[189] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[190] I. Bouyukliev and D. B. Jaffe, "Optimal binary linear codes of dimension at most seven," *Discrete Mathematics*, vol. 226, no. 1-3, pp. 51–70, 2001.

[191] J. M. Berger, "A Note on Error Detection Codes for Asymmetric Channels," *Information and Control*, vol. 4, no. 1, pp. 68–73, 1961.

[192] M. Zargham and P. G. Gulak, "Fully Integrated On-Chip Coil in 0.13μm CMOS for Wireless Power Transfer Through Biological Media," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 2, pp. 259–271, 2015.

[193] T. Ulz, T. Pieber, A. Höller, S. Haas, and C. Steger, "Secured and Easy-to-Use NFC-Based Device Configuration for the Internet of Things," *IEEE Journal of Radio Frequency Identification (JRFID)*, vol. 1, no. 1, pp. 75–84, 2017.

[194] T. J. Soon, "QR Code," *Synthesis Journal*, vol. 2008, pp. 59–78, 2008.

[195] T. Ulz, T. Pieber, C. Steger, C. Lesjak, H. Bock, and R. Matischek, "SECURECONFIG: NFC and QR-Code based Hybrid Approach for Smart Sensor Configuration," in *Radio Frequency Identification (RFID), 11th IEEE International Conference on*, pp. 41–46, IEEE, 2017.

[196] T. Ulz, T. Pieber, C. Steger, S. Haas, R. Matischek, and H. Bock, "Hardware-Secured Configuration and Two-Layer Attestation Architecture for Smart Sensors," in *Digital System Design (DSD), 2017 Euromicro Conference on*, pp. 229–236, IEEE, 2017.

[197] A.-R. Sadeghi and C. Stüble, "Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms," in *Proceedings of the 2004 workshop on New security paradigms*, pp. 67–77, ACM, 2004.

[198] A. Al Hasib and A. A. M. M. Haque, "A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography," in *2008 Third International Conference on Convergence and Hybrid Information Technology*, vol. 2, pp. 505–510, IEEE, 2008.

[199] M. Abdalla and D. Pointcheval, "Simple Password-Based Encrypted Key Exchange Protocols," in *CT-RSA, LNCS*, vol. 3376, pp. 191–208, Springer, 2005.

[200] T. Ulz, T. Pieber, C. Steger, A. Höller, S. Haas, and R. Matischek, "Automated Authentication Credential Derivation for the Secured Configuration of IoT Devices," in *Industrial Embedded Systems (SIES), 13th IEEE International Symposium on*, pp. 1–8, IEEE, 2018.

[201] T. Ulz, T. Pieber, C. Steger, S. Haas, and R. Matischek, "QSNFC: Quick and Secured Near Field Communication for the Internet of Things," in *Radio Frequency Identification (RFID), 12th IEEE International Conference on*, pp. 1–8, IEEE, 2018.

[202] B. Dowling, M. Fischlin, F. Günther, and D. Stebila, "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1197–1210, ACM, 2015.

[203] M. Fischlin and F. Günther, "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1193–1204, ACM, 2014.

[204] G. Xie and L. Wang, “Stabilization of Networked Control Systems with Time-Varying Network-Induced Delay,” in *2004 43rd IEEE Conference on Decision and Control (CDC)*, vol. 4, pp. 3551–3556, IEEE, 2004.

[205] M. Jungers, E. B. Castelan, V. M. Moraes, and U. F. Moreno, “A dynamic output feedback controller for NCS based on delay estimates,” *Automatica*, vol. 49, no. 3, pp. 788–792, 2013.

[206] S. Myagmar, A. J. Lee, and W. Yurcik, “Threat Modeling as a Basis for Security Requirement,” in *Symposium on requirements engineering for information security (SREIS)*, vol. 2005, pp. 1–8, Citeseer, 2005.

[207] M. Howard and D. LeBlanc, *Writing Secure Code*. Microsoft Press, 2003.

[208] Z. Chen, L. Yin, Y. Pei, and J. Lu, “CodeHop: physical layer error correction and encryption with LDPC-based code hopping,” *Science China Information Sciences*, vol. 59, no. 10, p. 102309, 2016.

[209] T. Ulz, T. Pieber, C. Steger, S. Haas, H. Bock, and R. Matischek, “Bring Your Own Key for the Industrial Internet of Things,” in *Industrial Technology (ICIT), 2017 IEEE International Conference on*, pp. 1430–1435, IEEE, 2017.

[210] T. Ulz, T. Pieber, C. Steger, S. Haas, and R. Matischek, “Sneakernet on Wheels: Trustworthy NFC-based Robot to Machine Communication,” in *Radio Frequency Identification - Technology & Application (RFID-TA), 2017 8th IEEE International Conference on*, pp. 260–265, IEEE, 2017.

[211] T. Ulz, S. Haas, and C. Steger, “Cyber-Physical System and Internet of Things Security: An Overview,” in *Solutions for Cyber-Physical Systems Ubiquity*, pp. 248–277, IGI Global, 2018.