School of Computing and Communications
Lancaster University

Novel Security Mechanisms for Wireless Sensor Networks

Ibrahim Ethem Bagci
B.Sc.
M.Sc.

Submitted for the degree of
Doctor of Philosophy
January 2016

To my family ...

Abstract

Wireless Sensor Networks (WSNs) are used for critical applications such as health care, traffic management or plant automation. Thus, we depend on their availability, and reliable, resilient and accurate operation. It is therefore essential that these systems are protected against attackers who may intend to interfere with operations. Existing security mechanisms cannot always be directly transferred to the application domain of WSNs, and in some cases even novel methods are desirable to give increased protection to these systems.

The aim of the work presented in this thesis is to augment security of WSNs by devising novel mechanisms and protocols. In particular, it contributes to areas which require protection mechanisms but have not yet received much attention from the research community. For example, the work addresses the issue of secure storage of data on sensor nodes using cryptographic methods. Although cryptography is needed for basic protection, it cannot always secure the sensor nodes as the keys might be compromised and key management becomes more challenging as the number of deployed sensor nodes increases. Therefore, the work includes mechanisms for node identification and tamper detection by means other than pure cryptography.

The three core contributions of this thesis are (i) Methods for confidential data storage on WSN nodes. In particular, fast and energy-efficient data storage and retrieval while maintaining the required protection level is addressed. A framework is presented that provides confidential data storage in WSNs with minimal impact on sensor node operation and performance. This framework is further advanced by combining it with secure communication in WSNs. With this framework, data is stored securely on the flash

file system such that it can be directly used for secure transmission, which removes the duplication of security operations on the sensor node. (ii) Methods for node identification based on clock skew. Here, unique clock drift patterns of nodes, which are normally a problem for wireless network operation, are used for non-cryptographic node identification. Clock skew has been previously used for device identification, requiring timestamps to be distributed over the network, but this is impractical in duty-cycled WSNs. To overcome this problem, clock skew is measured locally on the node using precise local clocks. (iii) Methods for tamper detection and node identification based on Channel State Information (CSI). Characteristics of a wireless channel at the receiver are analysed using the CSI of incoming packets to identify the transmitter and to detect tampering on it. If an attacker tampers with the transmitter, it will have an effect on the CSI measured at the receiver. However, tamper-unrelated events, such as walking in the communication environment, also affect CSI values and cause false alarms. This thesis demonstrates that false alarms can be eliminated by analysing the CSI value of a transmitted packet at multiple receivers.

Declaration

This thesis is a presentation of my original research work. No part of this thesis has been submitted elsewhere for any other degree or qualification. All work is my own unless otherwise stated. The work was carried out under the guidance of Dr Utz Roedig, at Lancaster University's School of Computing and Communication.

29th January 2016

Ibrahim Ethem Bagci

Copyright ©2016 by Ibrahim Ethem Bagci.

"The copyright of this thesis rests with the author. No quotations should be published or information and results derived from this thesis without acknowledgement."

Acknowledgements

I would like to express my gratitude to my supervisor, Utz Roedig, for his endless guidance, advice, support and encouragement during my doctoral study. He has helped in many ways. I feel extremely lucky to have him as my supervisor.

I would like to thank my examiners, Stephen Hailes and David Hutchison, for their comments and suggestions. I am also indebted to my previous supervisors, Bulent Tavli and Kemal Bicakci, for their encouragement to pursue a PhD degree.

I have had wonderful colleagues and friends in the department who have helped and supported me. I would like to thank James Brown, Steven Simpson, Paul Alcock, Alex King, Martin Bor, Jose Linares, Noor Shirazi, John Vidler, James Hadley, Arsham Farshad, Mohammad Reza Pourmirza and Arash Ghamari. I am also grateful to our current and former research support staff Carol, Aimee, Richard, Debbie, Liz, Barbara and Charlotte.

I have collaborated with great researchers during my PhD. I would like to thank Thiemo Voigt, Shahid Raza, Antony Chung, Matthias Schulz, Matthias Hollick and Ivan Martinovic.

I have had many good friends in Lancaster. They have made my life more bearable. I would like to thank Hayat, Muzeyyen, Ibrahim, Sertac, Hasan, Mehdi, Mahmoud, Gaurav, Arooj, Yehia, Federica, Musab, Yusuf, Ebru, Bulut and Leticia. I would also like to thank my friends Tugrul, Tuba, Yusuf, Halil and Ulvi in Manchester.

Last but not least, I am grateful to my family for their prayers and supports. I would like to thank my best friend and twin brother Enes, my elder sister Elif, and my parents.

Publications

Some of the materials in this thesis are published in various conferences and workshops. These publications and my contributions on them are listed below.

1. Ibrahim Ethem Bagci, Mohammad Reza Pourmirza, Shahid Raza, Utz Roedig, and Thiemo Voigt. Codo: Confidential Data Storage for Wireless Sensor Networks. In *Proceedings of the 9th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'12)*, vol.Supplement, pp.1-6, 8-11 October 2012

   Contributions: I contributed to the idea of the paper. I implemented Codo framework on Contiki operating system. I did the most of the evaluation of the proposed mechanism. I am the main author of the paper.

2. Ibrahim Ethem Bagci, Shahid Raza, Tony Chung, Utz Roedig, and Thiemo Voigt. Combined Secure Storage and Communication for the Internet of Things. In *Proceedings of the 10th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'13)*, pp.523-531, 24-27 June 2013.

   Contributions: I contributed to the idea of the paper. I implemented the extension for IPsec on Contiki operating system. I did the most of the evaluation of the proposed mechanism. I am the main author of the paper.

3. Ibrahim Ethem Bagci and Utz Roedig. Node Identification Using Clock Skew. In *Proceedings of the 5th Workshop on Real-World Wireless Sensor Networks (RealWSN'13)*, pp. 111-123, 19-20 September 2013.

   Contributions: I contributed to the idea of the paper. I implemented clock

sampling code on Contiki operating system, and linear programming code for clock skew calculation on MATLAB. I did the most of the evaluation of the proposed mechanism. I am the main author of the paper.

4. Ibrahim Ethem Bagci, Utz Roedig, Matthias Schulz, and Matthias Hollick. Short Paper: Gathering Tamper Evidence in Wi-Fi Networks Based on Channel State Information. In *Proceedings of the 7th ACM Conference on Security and Privacy in Wireless & Mobile Networks (WiSec'14)*, pp. 183-188, 23-25 July 2014.

   Contributions: I contributed to the idea of the paper. I implemented tamper detection code on MATLAB. I did the most of the evaluation of the proposed mechanism. I am the main author of the paper.

5. Ibrahim Ethem Bagci, Shahid Raza, Utz Roedig, and Thiemo Voigt. Fusion: coalesced confidential storage and communication framework for the IoT. *Security and Communication Networks*, 2015, DOI: 10.1002/sec.1260.

   Contributions: I contributed to the idea of the paper. I implemented the extension for IPsec on Contiki operating system. I did the most of the evaluation of the proposed mechanism. I am the main author of the paper.

6. Ibrahim Ethem Bagci, Utz Roedig, Ivan Martinovic, Matthias Schulz, and Matthias Hollick. Using Channel State Information for Tamper Detection in the Internet of Things. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC'15)*, 2015.

   Contributions: I contributed to the idea of the paper. I did the experiments. I implemented tamper detection code on MATLAB. I did the most of the evaluation of the proposed mechanism. I am the main author of the paper.

Table of Contents

List of Figures

List of Tables

List of Acronyms

6LoWPAN IPv6 Over Low Power Wireless Personal Area Networks

AES Advanced Encryption Standard

AH Authentication Header

AP Access Point

CBC Cipher-block chaining

CCA Clear Channel Assessment

CFR Channel Frequency Response

CFS Coffee File System

CIR Channel Impulse Response

CPU Central Processing Unit

CSI Channel State Information

CSS Chirp Spread Spectrum

CTR Counter

DCO Digitally Controlled Oscillator

DTLS Datagram Transport Layer Security

DSSS Direct Sequence Spread Spectrum

DSUWB Direct Sequence Ultra Wideband

ECC Elliptic Curve Cryptography

EER Equal Error Rate

ESP Encapsulating Security Payload

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

FTP File Transfer Protocol

FTSP Flooding Time Synchronization Protocol

ICV Integrity Check Value

IKE Internet Key Exchange

IoT Internet of Things

IP Internet Protocol

IPsec Internet Protocol Security

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

IV Initialization Vector

LP Linear Programming

LR Linear Regression

MF Management Frame

MIMO Multiple Input Multiple Output

MISO Multiple Input Single Output

MMSE Minimum Mean Square Estimator

MTU Maximum Transmission Unit

NH Next Header

NIC Network Interface Card

OFDM Orthogonal Frequency Division Multiplexing

PSSS Parallel Sequence Spread Spectrum

RAM Random Access Memory

RF Radio Frequency

ROC Receiver Operating Characteristic

RSSI Received Signal Strength Indication

RSS Received Signal Strength

RTC Real-time Clock

SA Security Associations

SDR Software Defined Radio

SN Sequence Number

SPI Security Parameter Index

SSL Secure Sockets Layer

SVD Singular Value Decomposition

TLS Transport Layer Security

TP True Positive

TPM Trusted Platform Module

TPR True Positive Rate

TOR The Onion Router

UDP User Datagram Protocol

WPAN Wireless Personal Area Network

WSN Wireless Sensor Network

Chapter 1

Introduction

Embedded systems are computer systems designed to handle a specific task, and they are often based on microcontrollers. Microcontrollers are small computers made of processor(s), memory and peripherals on a single circuit board. With increasing developments on microcontroller technology, embedded systems became the most common form of computers. They are being used in everyday objects like smart phones, electric toothbrushes, cars and credit cards. Other reasons for popularity of embedded systems are the improvements to wireless technologies and integration of them with the embedded systems. These *networked* embedded systems are the building blocks of Wireless Sensor Networks (WSNs).

WSNs are made of sensor nodes that sense the environment conditions and report back to a central station, usually called the *sink*. Sensor nodes are low-power embedded systems, and have sensing, processing and communication capabilities. WSNs are used in many application scenarios. Enabling Internet Protocol (IP) on WSNs, as well as on other networked embedded systems, led to realisation of the Internet of Things (IoT). It was estimated by Cisco that there were over 12.5 billion IoT objects used in 2010, and it is predicted to be 50 billion by 2020[1].

WSNs use different wireless protocols based on their application requirements. Some of these protocols include IEEE 802.15.4 and IEEE 802.11 standards, or proprietary pro-

[1] `https://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf`

tocols. New hardware technologies capable of handling different protocols simultaneously on a single device are also being developed. These technologies increase the usage of WSNs even more.

1.1 Problem Statement and Thesis Aims

WSNs are used in many application scenarios, including critical applications such as healthcare, traffic management and plant automation. These applications should work reliably, resiliently and accurately. Therefore, securing of these applications is of paramount importance. There are many security solutions proposed for traditional computer systems, but such solutions are designed to run on computers equipped with vast amounts of processing power and memory, making it impossible to apply them directly on low-powered sensor nodes that have limited processing and memory capabilities. Thus, existing security solutions should be revised, or new solutions should be proposed, to be able to secure WSNs.

The work presented in this thesis aims to enhance security of WSNs by designing novel mechanisms and protocols that will help to optimise the usage of limited node resources. The work contributes to areas that require protection mechanisms but have not yet received much attention from the research community. The issue of secure storage of data on sensor nodes is addressed, and mechanisms for node identification by means other than pure cryptography are presented.

1.1.1 Motivation

Security in WSNs has to consider more than just the communication links. In particular, sensor nodes make use of large local storage space in the form of flash memory to fulfil their tasks. For example, a security camera attached to a sensor node may record images and only provide image data (or the result of an image processing algorithm) on request. This stored data on smart objects must be secured as well in addition to the communication links used to access and transport this data. This is particularly important as sensor nodes may be easy to access and to retrieve by an attacker. Furthermore, sensor nodes

are expected to be cheap and deployed in large quantities. It can be expected that devices are thrown away at the end of their lifetime, and that a proper decommissioning process in which data would be erased is not commercially viable. Therefore, the stored data on sensor nodes should be cryptographically protected. Methods for efficient secure storage using cryptographic methods are presented in this thesis.

Although cryptographic methods are essential for protection, they are expensive for sensor nodes that have limited battery life and scarce processing power. Key management also becomes more challenging with the number of deployed sensor nodes increasing. Necessity for cryptographic methods should be reduced as much as possible. Furthermore, cryptographic methods cannot address all the security problems. For example, a surveillance system might use wireless cameras to monitor critical infrastructures such as an airport or power plant. Classical cryptography can be used to authenticate data transmitted from the camera devices. However, tampering with a device (e.g. movement or change of viewpoint) cannot be detected using cryptographic methods. For these reasons, it is desirable to provide an additional layer of defence. An attacker may also obtain key material and replace a node in the deployment to inject false observation data. For these reasons, it is desirable to provide an additional layer of defence. To prevent these kinds of attacks, various methods have been proposed to bind the identification and authentication to the device hardware. Using special custom chips or hardware characteristics of the device are some of the options. Custom chips are expensive and require changes on the node design. Therefore, mechanisms that use hardware characteristics of the device are more desirable. This thesis presents mechanisms for node identification and tamper detection that do not require cryptographic operations or any additional hardware.

1.1.2 Contributions

The three core contributions of this thesis are:

1. Methods for confidential data storage on WSN nodes.

2. Methods for node identification based on clock skew.

3. Methods for tamper detection and node identification based on Channel State Information (CSI).

1.1.2.1 Confidential Data Storage for Wireless Sensor Networks

Many WSNs are used to collect and process confidential information. Confidentiality must be ensured at all times and, for example, solutions for confidential communication, processing or storage are required. To date, the research community has addressed mainly the issue of confidential communication. Efficient solutions for cryptographically secured communication and associated key exchange in WSNs exist. However, as many WSN applications rely heavily on available on-node storage space, which is easily exposed to an attacker, it is essential to ensure the confidentiality of that stored data. This thesis presents Codo, a confidential data storage solution, which balances platform performance and security requirements. An implementation of Codo for the Contiki WSN operating system is provided, along with a performance evaluation.

Following from this work, the confidential data storage framework is combined with secure communication in WSNs, more specifically in IoT. The future IoT may be based on the existing and established IP. Many IoT application scenarios will handle sensitive data. However, as security requirements for storage and communication are addressed separately, work such as key management or cryptographic processing is duplicated. This part of the thesis presents a framework that allows us to combine secure storage and secure communication in the IP-based IoT. We see how data can be stored securely such that it can be delivered securely upon request without further cryptographic processing.

1.1.2.2 Node Identification Based on Clock Skew

Clocks on wireless sensor nodes experience a natural drift. This clock skew is unique for each node as it depends on the clock's manufacturing characteristics. Clock skew can be used as unique node identifier that is useful for node authentication. We will see how clock skew of a node's clock can be measured directly on the node by utilising the available high-precision radio transceiver clock. A detailed implementation of this

proposed local clock skew tracking method for the Zolertia Z1 platform is presented. We see how the required sampling effort to accurately measure clock skew can be determined. How clock skew measurements can be aligned with existing transceiver operations in order to avoid an increase in energy consumption is also discussed.

1.1.2.3 Tamper Detection and Node Identification Based on Channel State Information

Wireless devices are often used in application scenarios with strict security requirements. Examples are physical intrusion detection systems commonly used to protect factories, airports or government buildings. In such scenarios, additional security features such as tamper detection are highly desirable to complement traditional cryptographic mechanisms. In this work, we see how to use CSI, extracted from off-the-shelf 802.11n Wi-Fi cards, to calculate a tamper-evidence value for transmitters. This value enables detection of tampering due to device movement or replacement. Algorithms for tamper-evidence value computation are described, and the interpretation of this value is discussed and its effectiveness is evaluated. Unfortunately, not only tamper events lead to CSI fluctuations; movement of people in the communication environment has an impact too. Analysis of CSI values of a transmission simultaneously at multiple receivers is proposed to improve distinction of tamper and movement events. A moving person is expected to have an impact on some but not all communication links between transmitter and the receivers. A tamper event impacts on all links between transmitter and the receivers. The necessary algorithms for the proposed multi-receiver tamper detection method are described. In particular, the tamper detection capability in practical deployments with varying intensity of people movement is analysed. In our experiments, the proposed system deployed in a busy office environment is capable of detecting 53% of tamper events while creating zero false alarms.

1.2 Thesis Outline

Chapter 2 begins with a brief background about the topics related to the works in the thesis. It then gives an overview of the related work in security of WSNs, and discusses the related work for the individual chapters of the thesis.

Chapter 3 explains Codo, the confidential data storage solution for WSNs. It gives a design specification, explains the implementation, and evaluates the different aspects of Codo. The chapter then explains the framework for combined secure storage and communication for IoT with a detailed implementation and evaluation.

Chapter 4 describes our clock-skew-based node identification method. It describes the local clock skew determination process and gives an analysis of clock sampling requirements.

Chapter 5 explains tamper detection mechanism based on CSI. It describes a method for tamper evidence computation and gives a detection analysis.

Chapter 6 summarizes the chapters with overall conclusion of the thesis, and discusses the future work.

Chapter 2

Background and Related Work

This chapter first gives a brief background in Section 2.1 about the topics on which the works in the thesis are based. It then discusses the related work for individual chapters in Section 2.2.

2.1 Background

This thesis proposes novel security mechanisms for Wireless Sensor Networks (WSNs). Therefore, this section starts with brief background on Information Security (Section 2.1.1) and WSNs (Section 2.1.2). A combined secure storage and secure communication framework is proposed in Chapter 3. This framework uses IPv6/6LoWPAN and IPsec/ESP protocols, and necessary background is given in Sections 2.1.4 and 2.1.5. Chapter 5 proposes a node identification and tamper detection mechanism using Channel State Information (CSI) of 802.11n networks. An overview of the 802.11n standard is given in Section 2.1.6.

Background information given here is mainly brief explanations of the concepts. More detailed information is given about the proposed solutions in each chapter if necessary.

2.1.1 Information Security

Information Security is a practice of protecting data on a computer system from those with malicious intentions. Its key goals are Confidentiality, Integrity, Availability and

Non-repudiation:

- *Confidentiality* is the prevention of disclosure of the information to unauthorized parties.
- *Integrity* is the prevention of modification of the information by unauthorized parties.
- *Availability* is the ability to provide the information to authorized parties when it is needed.
- *Non-repudiation* is the prevention of a sending/receiving party of a transaction from denying that it sent/received the transmission.

Access Control is another important concept in Information Security. It controls who can access the information on a computer system. It usually consists of three steps:

- *Identification* is claiming what something is or who someone is, for example, providing a user name to a system.
- *Authentication* is verifying the claim, for example asking a password for a given user name.
- *Authorization* is giving someone or something permission to a given system after passing identification and authentication steps.

This thesis proposes a combined *confidential* storage and communication framework, that efficiently provides *integrity*, in Chapter 3. Clock skew values are used to *identify* sensor nodes in Chapter 4. These values can further be used for *authentication*. In Chapter 5, channel characteristics are used to *identify* wireless devices and detect tampering.

2.1.2 Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a wireless network consisting of sensor nodes that are small, low-powered, embedded devices. A sensor node typically has a microcontroller, a

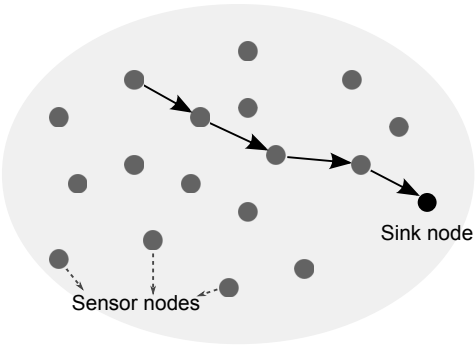


Figure 2.1: A typical multi-hop Wireless Sensor Network.

radio transceiver, a battery and several sensors. Each monitors and senses the environment, and reports back to a central node, usually called the *sink*. Data is generally sent from a node to the sink in a multi-hop fashion, where the intermediate nodes between the node and the sink participate in the sending operation. A typical multi-hop WSN is depicted in Figure 2.1.

WSNs are used in many application scenarios such as environmental, health, home, industrial, military, and so on. Based on the application requirements, they use different wireless protocols. The most popular of these is the IEEE 802.15.4 standard. WSNs also use the IEEE 802.11 standard, Bluetooth and proprietary protocols. An application can use multiple protocols at the same time as well. One example scenario is a home automation system consisting of a wireless camera, a connected light switch, a temperature sensor and a controller that controls these devices. The wireless camera uses 802.11, the light switch uses a proprietary protocol (for example LightwaveRF), and the temperature sensor uses 802.15.4 to connect to the controller. Additionally, new hardware technologies capable of handling different protocols simultaneously on a single device also exist.

2.1.3 IEEE 802.15.4

The IEEE 802.15.4 standard defines the physical layer and media access control for low-data-rate Wireless Personal Area Networks (WPANs), which require low-cost and low-speed communication between devices. 802.15.4 specifies wireless communication techniques, wireless spectrum to be used and media access control algorithms. A simple diagram of the standard is shown in Figure 2.2.

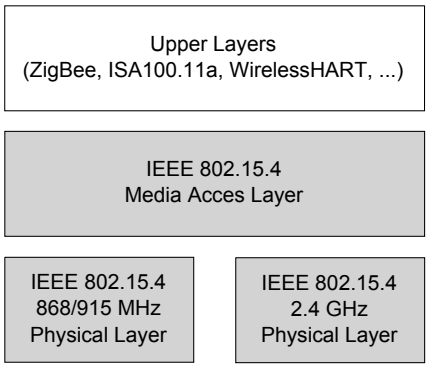


Figure 2.2: A simple diagram of IEEE 802.15.4 protocol stack.

802.15.4 can operate on three different frequency bands: 2.4 GHz (global), 915 MHz (North America), and 868 MHz (Europe). Data rates are 20 to 250 kbps and the transmission range of the devices varies between 10 m and 100 m. 802.15.4 uses Direct Sequence Spread Spectrum (DSSS) (most common), Parallel Sequence Spread Spectrum (PSSS), Chirp Spread Spectrum (CSS) or Direct Sequence Ultra Wideband (DSUWB) technologies for the physical layer modulation technique.

Media access control enables the control of the 802.15.4 packets transmitted over the air. It provides management and data services to the upper layers. Upper layer specifications such as ZigBee, ISA100.11a and WirelessHART extend the standard based on the application requirements.

2.1.4 6LoWPAN

Every device on the Internet has a numerical label, called Internet Protocol (IP) address, that is used for device identification and location addressing. Internet Protocol version 4 (IPv4) is the most used version of the IP protocol. IPv4 has a 32-bit address space, and it can assign 2^{32} addresses. With the increase of the devices available on the Internet, more addresses than IPv4 could handle were needed. Therefore, Internet Protocol version 6 (IPv6) was proposed. IPv6 has a 128-bit address space allowing 2^{128} addresses, more than 7.9×10^{28} times as many as IPv4 can offer.

Although IPv6 allows a huge number of devices to be addressed on the Internet, low powered networked embedded devices were being left out from the Internet because of their

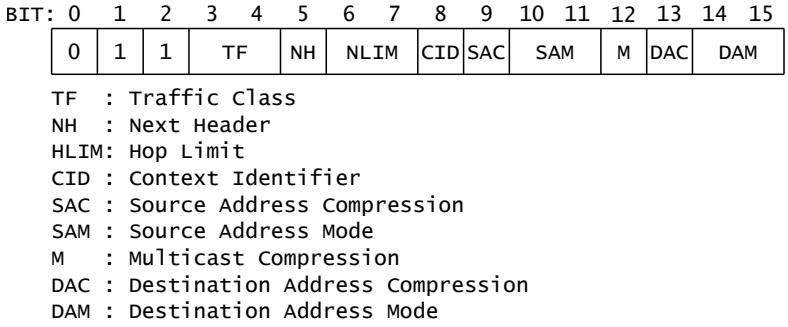


Figure 2.3: LOWPAN_IPHC encoding in 6LoWPAN.

Figure 2.4: General LOWPAN_NHC encoding in 6LoWPAN.

limited processing ability and small packet size. WSNs mostly use the 802.15.4 standard, and packet sizes in 802.15.4 are too small to use IPv6. The Maximum Transmission Unit (MTU) must be at least 1280 bytes in IPv6, while the packet size of 802.15.4 is 127 bytes. IPv6 Over Low Power Wireless Personal Area Networks (6LoWPAN) was introduced to this end, and it allowed embedded devices to use the IP protocol. 6LoWPAN-enabled devices can send and receive IPv6 packets over 802.15.4-based networks, thanks to the encapsulation and header-compression mechanisms of 6LoWPAN. It compresses the IPv6 header, and increases the payload carried in 802.15.4 frames. Enabling IP on networked embedded systems led to the realisation of the Internet of Things (IoT).

Header-compression mechanisms defined by 6LoWPAN use LOWPAN_IPHC for IP header compression and LOWPAN_NHC for the next-header compression. Figure 2.3 shows the LOWPAN_IPHC header. The IP header length is reduced to 2 bytes for single-hop networks and to 7 bytes for multi-hop networks with LOWPAN_IPHC. When the *next header* field is set to 1 in LOWPAN_IPHC, the next header of compressed IPv6 header will be encoded with LOWPAN_NHC. A general LOWPAN_NHC header is shown in Figure 2.4. The length of LOWPAN_NHC header can be 1 or more bytes. The first variable bits are used to identify the next header type, and the remaining bits encode the header information. 6LoWPAN defines LOWPAN_NHC for only IP extension header (LOWPAN_NHC_EH) and the User Datagram Protocol (UDP) header

```
[01]  # ESP SAs using 192 bit long keys (168 + 24 parity)
[02]  add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc
[03]          0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
[04]  add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc
[05]          0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;
```

Figure 2.5: An example of ESP SAs taken from an IPsec configuration file on Ubuntu operating system.

(LOWPAN_NHC_UDP).

The combined secure storage and communication framework proposed in Chapter 3 is based on IPv6/6LoWPAN protocols.

2.1.5 IPsec

IPv6 communications between two end points are secured by Internet Protocol Security (IPsec). IPsec is a protocol suite that includes Authentication Header (AH) and Encapsulating Security Payload (ESP). AH provides integrity and data origin authentication. ESP provides integrity, data origin authentication and also confidentiality. IPsec uses Security Associations (SA) that keep the bundle of algorithms and parameters necessary for AH and ESP operations. There is also a 32-bit indexing parameter, called the Security Parameter Index (SPI), in each SA that is used by a receiver to identify the correct SA.

An example of ESP SAs taken from an IPsec configuration file on the Ubuntu operating system is shown in Figure 2.5. There are two 192-bit keys defined for communications from 192.168.1.100 to 192.168.2.100 and from 192.168.2.100 to 192.168.1.100 on lines 3 and 5, respectively. The numbers after each *esp* keyword represent SPIs, and they must be unique. The 3DES algorithm in CBC mode is used for encryption.

2.1.5.1 IPsec for 6LoWPAN

IPsec is defined for IPv6, and it is not suitable to use directly on 6LoWPAN packets. Therefore, a definition for AH and ESP encodings for 6LoWPAN is provided in [RDH+12], in addition to the IP extension header and the UDP header encodings. They use IP extension header (LOWPAN_NHC_EH) to link AH (LOWPAN_NHC_AH) and ESP

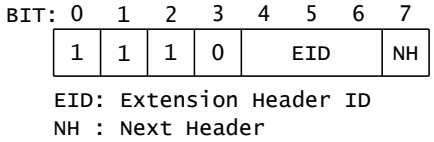


Figure 2.6: LOWPAN_NHC_EH encoding in 6LoWPAN.

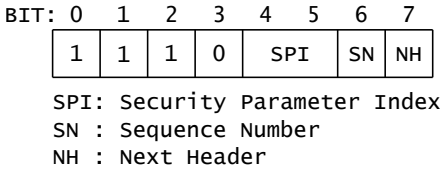


Figure 2.7: LOWPAN_NHC_ESP encoding in 6LoWPAN.

(LOWPAN_NHC_ESP) extension headers.

LOWPAN_NHC_EH header is shown in Figure 2.6. The Extension Header ID field is used to identify whether the next header is an AH or ESP header. The Next Header field is set to 1 to indicate that the next header is LOWPAN_NHC encoded.

Since only IPsec/ESP is used in Chapter 3, an explanation of the LOWPAN_NHC_AH extension header is omitted here. The LOWPAN_NHC_ESP extension header is depicted in Figure 2.7. Its fields are defined as follows:

- The first four bits in the LOWPAN_NHC_ESP represent the NHC ID defined for ESP. These are set to 1110.
- If SPI = 00: the default SPI for the 802.15.4 network is used, and the SPI field is omitted. We set the default SPI value to 1. This does not mean that all nodes use the same SA, but that every node has a single preferred SA, identified by SPI 1.

  If SPI = 01: First 8 bits of the SPI are carried inline; the remaining 24 bits are elided.

  If SPI = 10: First 16 bits of the SPI are carried inline; the remaining 16 bits are elided.

  If SPI = 11: All 32 bits of the SPI are carried inline.
- If SN = 0: The first 16 bits of the sequence number are used. The remaining 16 bits are assumed to be zero.

If SN = 1: All 32 bits of the sequence number are carried inline.

- If NH = 0: The next header field in ESP will be used to specify the next header, and it is carried inline.

  If NH = 1: The next header will be encoded using LOWPAN_NHC. In case of ESP, this would require the end systems to perform 6LoWPAN compression/decompression and encryption/decryption jointly.

Combined secure storage and communication framework proposed in Chapter 3 is based on IPsec/ESP protocols.

2.1.6 IEEE 802.11n

The IEEE 802.11 standard specifies the physical layer and media access control for wireless local-area network communications. 802.11n is an amendment to 802.11, and it was first proposed in 2002 and its final draft was approved in 2009. 802.11n can operate on 2.4 or 5 GHz frequency bands, and it provides data rates from 6 to 600 Mbit/s. Wireless systems based on 802.11 standards are called *Wi-Fi* systems.

The 802.11n standard uses a Multiple Input Multiple Output (MIMO) technique, where multiple antennas are used by both transmitter and receiver. MIMO helps to improve the network reliability and performance with spatial diversity and spatial multiplexing techniques. Network reliability can be improved with spatial diversity by sending the same data from all the antennas, and network performance can be improved with spatial multiplexing by sending independent data streams from different spatial dimensions. Transmitters decide which technique to use according to the channel conditions. Therefore, receivers have to estimate the amplitude changes and phase shifts on the channel, and report back this information to the transmitters. The estimation of the channel is called Channel State Information (CSI). An example of a MIMO system consisting of a transmitter and a receiver with 2 antennas is shown in Figure 2.8.

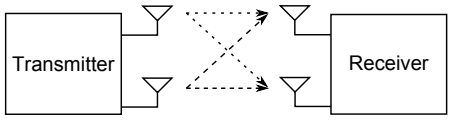


Figure 2.8: A MIMO system consisting of a transmitter and a receiver with 2 antennas.

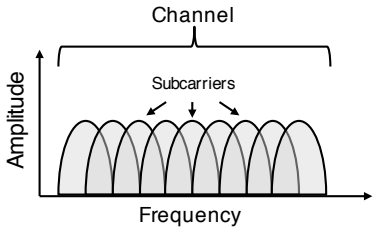


Figure 2.9: An OFDM signal in the frequency domain.

2.1.6.1 Beamforming and Spatial Expansion

802.11n uses Orthogonal Frequency Division Multiplexing (OFDM) as a modulation scheme. In OFDM, information is sent by multiple subcarriers (or subchannels) *sc* within the same single channel. An example of an OFDM signal in the frequency domain is depicted in Figure 2.9.

The wireless channel can be described linearly on each subcarrier as a matrix $\mathbf{H}^{sc}_{R\times T}$ where the signals of each transmission antenna $T^{sc}_1 \dots T^{sc}_T$ are projected onto the reception antennas $R^{sc}_1 \dots R^{sc}_R$. Transmitters can perform beamforming or spatial expansion using filtering matrices $\mathbf{F}^{sc}_{T\times S}$ instead of directly transmitting data streams over each transmission antenna. Spatial expansion improves reliability (but also increases redundancy) by expanding a data stream to several transmission antennas. Beamforming improves performance by allowing the transmission of multiple spatial streams $S^{sc}_1 \dots S^{sc}_S$ in parallel. The transmitter performs Singular Value Decomposition (SVD) on the channel matrix and decides whether spatial expansion or beamforming should be used. If we represent the CSI matrix as $\mathbf{M}^{sc}_{R\times S}$ measured for each subcarrier *sc*, the channel can be modelled as

$$\begin{bmatrix} R_1^{sc} \\ \vdots \\ R_R^{sc} \end{bmatrix} = \underbrace{\begin{bmatrix} H_{1,1}^{sc} & \dots & H_{1,T}^{sc} \\ \vdots & \ddots & \vdots \\ H_{R,1}^{sc} & \dots & H_{R,T}^{sc} \end{bmatrix} \begin{bmatrix} F_{1,1}^{sc} & \dots & F_{1,S}^{sc} \\ \vdots & \ddots & \vdots \\ F_{T,1}^{sc} & \dots & F_{T,S}^{sc} \end{bmatrix}}_{\begin{bmatrix} M_{1,1}^{sc} & \dots & M_{1,S}^{sc} \\ \vdots & \ddots & \vdots \\ M_{R,1}^{sc} & \dots & M_{R,S}^{sc} \end{bmatrix}} \begin{bmatrix} S_1^{sc} \\ \vdots \\ S_S^{sc} \end{bmatrix}$$

which can be written in a shorter form as

$$\mathbf{R}^{sc} = \mathbf{H}_{R\times T}^{sc}\mathbf{F}_{T\times S}^{sc}\mathbf{S}^{sc} = \mathbf{M}_{R\times S}^{sc}\mathbf{S}^{sc}$$

where R, T and S are the numbers of reception antennas, transmission antennas and spatial streams, respectively. Here, $\mathbf{R}^{sc}$, $\mathbf{F}^{sc}$ and $\mathbf{S}^{sc}$ are the corresponding signal vectors per subcarrier sc.

Chapter 5 uses CSI values of 802.11n OFDM networks for tamper detection. CSI values for each subcarrier provide rich information about the wireless channel.

2.2 Related Work

Security research in WSNs has received much attention over the years with the increasing use of WSNs. Unlike the wired or conventional wireless networks, WSNs operate with sensor nodes that have limited resources in terms of processing, memory, power or bandwidth. Therefore, traditional security solutions cannot be applied to WSNs directly, and so security needs special care.

We proceed with an overview of existing work on security in WSNs. The following sections then explain the existing work in detail related to the each chapters of the thesis.

Researchers have presented attacks against the functions of different network protocol

layers of WSNs [CMYP09b]. These attacks can be applied to a single layer or multiple layers of the protocol, and they include jamming, collusion, unfairness, etc. Cryptography is the first solution that comes to mind to circumvent most of these attacks, as is the case in any other systems.

Various cryptographic algorithms are investigated for their suitability to the requirements of sensor nodes with limited resources [CMYP09b]. New cryptographic libraries, methods and platforms are also proposed. Arazi et al. proposed an RSA-based framework against DoS attacks that causes an attacker's resources to become exhausted before the sensor's resources do [AQR07]. secFlek is a Trusted Platform Module (TPM) based on a public-key platform that provides energy efficiency and fast security [SHCO08, HCSO09]. Liu et al. proposed TinyECC, a flexible and configurable Elliptic Curve Cryptography (ECC) library for WSNs [LN08]. Application of pairing-based cryptography to WSNs and its implementation were investigated in [SKSC09]. Kothmayr et al. proposed an end-to-end security solution for IoT-based on Datagram Transport Layer Security (DTLS) [KHS+11].

Cryptographic operations come with the key-management problem. Considering the dynamic structure of WSNs, key management becomes more troublesome [CMYP09b, JBMC10, CY05]. An asynchronous key-distribution scheme for WSNs was proposed in [Han09]. The scheme does not need time synchronization, making it an efficient solution. A new key-management method was proposed by Nilsson at al. [NRLV08]. Their method provides backward and forward secrecy. Hang et al. proposed a new key-agreement protocol [HUW11]. The protocol is based on Diffie-Hellman key-agreement and provides perfect forward secrecy. Gauge et al. presented a solution for secure key assignment for WSNs [GSM09]. The solution is easy and convenient to use and robust against eavesdroppers.

There are security architectures for WSNs that use these cryptographic and key-management solutions. The most popular ones are TinySec [KSW04] and MiniSec [LMPG07]. TinySec is the first link-layer security architecture for WSNs. Although it achieves low energy consumption, it reduces the level of protection. MiniSec is the

successor of TinySec, and achieves low energy consumption while providing higher security. Both TinySec and MiniSec are implemented on the TinyOS operating system. Casado et al. proposed ContikiSec, a security architecture for the Contiki operating system [CT09].

Various security issues in WSNs like secure routing [CMYP09b], secure data aggregation [OM07, Yu09, BJT12], secure localization [ZCHX09, MSS10], secure code dissemination [LON08, TOZJ09], and broadcast authentication [RYLZ09], have also been addressed. Intrusion detection [KBG+09, WFA09] and anomaly detection [ZB11] methods have been investigated for WSNs. Analyses of memory protection [KKS07, CAE+07], code injection [FC08], buffer overflow [GN08], software-based attestation [CFPS09], and access control [FGS09] have also received attention in the research community [GFN11].

There are also privacy-oriented works, including query privacy [CYS+10], privacy for data aggregation [ILMé10], source-location privacy [SHZ+09, KFLFS11], event-source unobservability [YSZ+08, BGTB11] and sink unobservability [BBT11].

The next sections will discuss the related work in detail for the individual chapters of the thesis: (i) Confidential Data Storage in WSNs; (ii) Node Identification Based on Clock Skew; (iii) Tamper Detection and Node Identification Based on CSI.

2.2.1 Confidential Data Storage in Wireless Sensor Networks

Existing work proposes many novel security mechanisms and approaches for WSNs [CMYP09a]. However, the existing work is mostly focused on securing communication. Still, there are several contributions on secure file storage in WSNs which is most related to the presented work. The existing solutions cover individual aspects of Codo (confidential secure storage framework presented here), such as key management or integration with available flash memory structure, but do not address all aspects.

Bhatnagar and Miller presented a secure and reliable file system [BM07]. A unique seed is assigned to each node before the deployment and nodes use this seed to generate keys. Every encryption operation uses a newly generated key. The key generation process cannot be reversed and, therefore, when an adversary steals the node he cannot decrypt already encrypted data and he can only access a small amount of data which is not

encrypted yet. This approach is very secure but prevents a node from accessing stored data, which is necessary for any in-network processing. In contrast, Codo allows nodes to access stored data for processing.

Systems described by Pietro et al. [PMST08] and Girao et al. [GWMA07] are similar to the previous described solution. All of these solutions use key generation methods that prevent nodes from accessing data locally.

Ren et al. proposed a secure, dependable and distributed storage scheme [RRZ08]. Different to the previous solutions, they suggest public key encryption to ensure data confidentiality. In this case, the stored data can also not be accessed by nodes. In addition, public key cryptography requires more processing than symmetric cryptography as used in Codo.

Considering its speed and better security (e.g. resistant to cold boot attacks) hardware-based storage encryption has been made available lately by many vendors. An example company Ironkey [Iro] manufactures secure USB flash drives in which implemented hardware-based AES 256-bit encryption in CBC mode. Codo is different to these solutions as it does not rely on special storage hardware.

In the second part of this work, the secure storage framework is combined with secure communication solutions in IoT to remove the duplication of security operations on the sensor node. Solutions for secure communication and secure storage of data in the IP-based IoT exist, but these functions are generally designed and operated independently of each other. To the best of our knowledge, this is the first work that aims to combine both aspects. Secure storage solutions are discussed above. The secure communication solutions will be discussed in the next paragraphs.

Communication in the IoT can be secured on different layers. The IoT uses the IEEE 802.15.4 [IEE03] link-layer. IEEE 802.15.4 link-layer security is the current state-of-the-art security solution for the IP-connected IoT; it defines data encryption and integrity verification.

IEEE 802.15.4 security does not provide end-to-end security when connecting an IEEE 802.15.4 network via a gateway router to the existing Internet. Thus, additional

solutions exist that protect data traveling from Internet hosts to the border router. For example, ArchRock PhyNET [Arc08] applies IPsec in tunnel mode between the gateway router and Internet hosts.

To achieve true end-to-end security between Internet hosts and smart objects, an IPsec extension for 6LoWPAN has been proposed [RDH+12]. Unmodified Internet hosts can communicate directly with smart objects. The border router applies 6LoWPAN header compression in order to enable efficient transport of IPsec packets in IEEE 802.15.4 networks. This mechanism is used for the framework proposed in the thesis.

End-to-end security can be provided by using Transport Layer Security (TLS) or its predecessor Secure Sockets Layer (SSL). SSL has been proposed as security mechanism for the IoT by Hong et al. [HKH+10]. Foulagar et al. propose a TLS implementation for smart objects [FMMA06].

2.2.2 Node Identification Based on Clock Skew

Clock skew is the deviation of a clock from the true time. Fingerprinting devices using clock skew is carried out by comparing frequencies of two clocks, one of them generally assumed to represent the true time.

Kohno et al [KBC05] has shown that clock skew of devices can be measured to fingerprint devices. It is shown that the clock skew of each device is unique and stays fairly consistent over time.

Zander et al. [ZM08] improved clock skew measurement by applying a technique called synchronized sampling. They demonstrated that synchronization of samples reduces the quantisation error and hence improves skew determination quality.

Jana et al. [JK08] used clock skew to fingerprint wireless devices. The motivation for their work was the detection of fake wireless access points. Their work demonstrates that emitting timestamped beacons with high frequency allows for precise clock skew calculation. According to their observations, 50 to 100 beacons are sufficient to estimate clock skew accurately enough to identify individual nodes.

Arackaparambil et al. [ABSK10] demonstrated a clock skew spoofing attack in 802.11

networks by using virtual interfaces. In their work they propose methods to combat clock skew spoofing and propose standardised interfaces that would allow network providers to publish clock skew information.

Huang et al. [HTW+08] demonstrated clock-skew-based identification in wireless sensor networks in the context of the Flooding Time Synchronization Protocol (FTSP) [MKSL04]. FTSP provides coarse estimation of clock skew based on current offset and previous skew (it uses linear regression on the past 8 data points).

Uddin et al. [UC10] demonstrated that sensor nodes have a unique clock skew and that the clock skew of a node can easily be monitored.

Murdoch et al. [Mur06] split skew into two components, a constant and a variable part. The variable part is affected by temperature changes and this effect was used to reveal node identities in the The Onion Router (TOR) network by influencing CPU load and hence the temperature of devices leading to measurable clock skew changes.

The proposed mechanism in the thesis differs from existing approaches as clock skew is measured locally on nodes. It is believed that this is a necessary step towards a practical system as variations in communication delays cannot be avoided in any real-world WSN deployment. Furthermore, it is shown how clock skew measurements fit with energy-efficient operations of sensor nodes and investigate the required sampling effort in detail. Existing work with the exception of Huang et al. [HTW+08] calculate clock skew offline after a long sequence of samples are collected, using a linear programming approach. Huang et al. [HTW+08] calculate clock skew online using a linear regression approach which is adapted in this work.

2.2.3 Tamper Detection and Node Identification Based on Channel State Information

Existing work can be grouped into two main categories: transmitter identification and transmitter localisation. Transmitter identification aims to use received signal characteristics to identify the transmitting device (or class of device). Transmitter localisation aims to use the received signal characteristics to determine the location (or

area) of the transmitter. The work presented in this thesis falls into both areas. It differs from existing work in four main ways: (i) Most existing work is not based on 802.11n OFDM and no work so far has incorporated the fact that dynamic adaptation of the number of spatial independent streams in 802.11n must be taken into account. (ii) Existing work is mostly aimed at rejecting individual messages from an attacker, while the aim of this work is to determine a tamper-evidence value based on a number of incoming transmissions. (iii) Existing work does not evaluate the relation between transmitter movement and detection capability of a detection system. (iv) Existing tamper detection systems do not work in practical deployments as they do not address properly the separation of environmental and tamper events.

A number of recent works aim at transmitter localisation using channel characteristics. Li et al. [LXMT06] proposed a method for PHY layer authentication based on measuring the *Channel Frequency Response* (CFR). Three USRP/GNURadio Software Defined Radios (SDRs) are deployed at different locations and used as transmitter, receiver and attacker. Transmitter and attacker send packets alternately to the receiver. By employing a change-point detector, transmissions can be distinguished. Patwari et al. [PK07] used *Channel Impulse Response* (CIR) information to construct link signatures for location distinction. A history of $N-1$ transmission signatures is compared with the Nth transmission signature using the Euclidean distance to decide whether the transmission is from a new location. The method in [PK07] analyses features in the time domain whereas [LXMT06] operates in the frequency domain. Additionally, [LXMT06] uses a complex-valued signature where the phase information is included, while [PK07] uses a real-valued signature where the phase information is excluded. Zhang et al. [ZFPK08] combined the best features of [LXMT06] and [PK07], which are (i) the advantage of operating in the time domain and (ii) the advantage of using complex-valued signatures. Recently, Jiang et al. [JZL+13] proposed a source-authentication method to detect spoofing attacks on 802.11n Management Frames (MFs) by using CSI. Although this work is not aimed at location distinction, it uses the same source of information (CSI) as used in this thesis. It is shown that amplitude of CSI changes for injected frames. To

the best our knowledge, this work is the closest one to the work presented in this thesis in terms of source of information, encoding methods and test devices. However, the work does not consider encoding using independent spatial streams as it is used in practical 802.11n deployments. Faria et al. [FC06] created *signalprints* based on Received Signal Strength (RSS) information to identify wireless devices with respect to their locations. However, the method proposed here uses a richer set of channel characteristics.

In addition to location identification, PHY-layer information is also used for device identification. Brik et al. [BBGO08] used modulation errors to identify 802.11 devices caused by modulator circuitry. The transient part of the RF signal was used to identify 802.11 [US07] and 802.15.4 [DC09] devices. Danev et al. [DHBC09] used RF burst information to identify RFID transponders. Xiong et al. [XJ13] identified Wi-Fi clients by looking at the angle-of-arrival information of clients' incoming signals by leveraging multi-antenna Access Points (APs). More detailed information about device identification based on PHY-layer information can be found in [DZC12].

Chapter 3

Confidential Data Storage for Wireless Sensor Networks

In the first part of this chapter, the Codo framework will be explained. Codo provides confidential data storage in Wireless Sensor Networks (WSNs) with minimal impact on sensor node operation and performance[1]. Combined secure communication and secure storage in WSNs will be explained in the second part of the chapter. This combination removes the duplication of security operations on the sensor node.

3.1 Confidential Data Storage for Wireless Sensor Networks

In many WSNs sensor data is transferred immediately to a sink for analysis and/or storage. In such application cases no data is stored on nodes and confidentiality of stored information is not an issue. However, a number of (recent) applications use available on-node storage space to add new features or to improve network performance. For example, on-node storage may be used as insufficient network capacity is available to transport all gathered data from all nodes to the sink. Instead, stored data is pre-processed by nodes and only processing results are transmitted. The sink may as well request processing or transmission of stored data.

[1]This chapter is based on the papers titled "Codo: Confidential Data Storage for Wireless Sensor Networks", "Combined Secure Storage and Communication for the Internet of Things" and "Fusion: coalesced confidential storage and communication framework for the IoT".

An example application is industrial process monitoring and control [SSW+09]. Sensors may be used to collect and store vast amounts of data on production processes. Nodes process sensor data and transmit results to a sink node. At times, the sink may request nodes to process sensor data over specific time periods (e.g. to calculate the average of a sensor reading over a recent set time period) or request all sampling points in a specific time period. Such specific data requests may be necessary for error diagnosis or to calibrate the overall production processes. As sensors store information about production processes, it is of vital interest to a company to keep such information hidden from competitors. If a node is removed from the facility, it should not be possible to retrieve the stored data.

To secure data stored on nodes it has been proposed to simply encrypt the data before storage using key chains (see for example [BM07, RRZ08]). Such a naive approach ensures confidentiality but at the same time restricts a node's capability and does not consider performance issues. Such existing solutions do not enable nodes to access already stored data for in-network data processing on nodes and do not allow us to balance performance and security concerns. Furthermore, existing solutions are not tailored to hardware specifics such as flash memory layout or available hardware support for cryptographic algorithms.

In the first part of this chapter we present Codo, a framework for efficient confidential data storage on sensor nodes. Codo addresses the aforementioned shortcomings present in existing solutions. Codo allows for confidential data storage while enabling in-network data processing on nodes. Security concerns and performance can be balanced by deciding how much unencrypted data can be present on a node at any given point in time. Encrypted data storage is aligned with flash memory layout and cryptographic hardware support. The specific contributions of this work are:

- *Codo:* We give a design specification of the efficient confidential data storage framework.
- *Codo Implementation:* We detail the implementation of Codo for the Contiki operating system [DGV04] running on a Tmote Sky. In particular, we show the

integration of Codo with the Contiki flash file system Coffee [TDZV09].

- *Codo Evaluation:* We evaluate the different aspects of Codo and quantify performance implications.

Existing confidential data storage solutions have a number of shortcomings and limitations. Rather than addressing each aspect individually it is necessary to tackle all aspects together; as it is then possible to optimize usage of limited node resources. Next we discuss shortcomings of existing solutions and then describe Codo, the confidential storage framework that addresses these.

3.1.1 Limitations of Existing Solutions

Security Issues Most existing solutions encrypt all data with a single key before storage. Only the owner of the key (for example, the sink) is able to decrypt stored data. Such a solution has the drawback that security depends on a single point of failure; if the key is revealed all data can be accessed. A better approach is to store chunks of data using individual keys (for example, as described by Bhatnagar et al. [BM07] where an irreversible key chain starting with a random seed is constructed). With such an approach, a single key loss does not lead to a total loss of confidentiality. However, such a solution still does not address the important WSN aspect of in-network processing on nodes. Many sensor node applications rely on the nodes' ability to process previous stored data. Mechanisms to retrieve keys for previously stored chunks of data must be available. Another approach is to use public key cryptography [RRZ08]. Private keys are stored in the sink, therefore an adversary cannot decrypt the data. However with this approach, the stored data can also not be accessed by nodes. Furthermore, public key cryptography generally requires a lot of processing resources that are not available on sensor nodes.

Performance Issues Existing security solutions focus on optimizing encryption performance (for example, by optimizing encryption algorithms [SOS+08] or by employing specialised cryptographic hardware [HCSO09]). However, performance gains that result

from looking at a secure storage solution from a systems perspective are largely ignored. In existing solutions, data is generally encrypted as soon as it produced (for example, in [BM07, RRZ08, PMST08, GWMA07]). However, for many applications it is from a security perspective possible to cache data unencrypted before performing bulk encryption and storage. Obviously, the amount of data that can be cached unencrypted will depend on the specific application.

Hardware Issues Current solutions are hardware agnostic, which leads to inefficiencies. Flash memories used on sensor nodes are restricted in terms of read and write capabilities. It is often only possible to access chunks of data rather than in individual bytes and it is always more efficient to process data in chunks. Thus, crypto mechanisms should operate on chunk sizes that reflect hardware capabilities. Existing solutions also ignore that hardware support for cryptographic operations exists on WSN nodes. Hardware support is generally available for secure communication but it is possible to re-use these features for secure storage. Again, the cryptographic hardware is optimised for specific data chunk sizes and, if used in the context of storage, these hardware restrictions must be taken into account.

3.1.2 Codo: Confidential Data Storage Framework

Codo tackles the aforementioned limitations and shortcomings of existing confidential data storage solutions. The framework aims to realize confidential data storage with minimal impact on node operation and performance. Nodes should be able to store and access stored data as possible without confidential storage. Nodes should prevent performance degradation due to the additional security functions as much as possible.

In our proposed storage framework data is organised in *DataChunks*. Some unencrypted data is cached to improve performance; depending on application security and performance needs it can be decided how much unencrypted cached data can be present. To improve performance only complete DataChunks are cryptographically processed. The size of DataChunks is matched to the capabilities of storage hardware (e.g. page sizes) and to the capabilities of the encryption hardware (e.g. buffer size of cryptographic

processor).

DataChunk Size The DataChunk size S_D is determined by a number of factors. These are:

- Cryptographic Algorithm: The cryptographic algorithm usually operates on fixed block sizes S_B. The DataChunk size should therefore be aligned with this block size. Thus, the DataChunk size S_D must be a multiple of S_B.
- Cryptographic Hardware Support: If cryptographic hardware support is available it is normally operating most efficiently on a block size of S_C. The transfer of data to and from the crypto processor has a fixed cost element (addressing, loading operations, etc.) and a variable cost element that depends on the data size to be processed. S_C is a multiple of S_B but in many practical setting $S_C = S_B$. Again, S_D must be a multiple of S_C.
- Flash Memory: Flash memory is organized in pages of size S_P. Depending on the flash memory hardware the page size implies different constraints. For example, with some hardware it is only possible to read or write a whole page. Other hardware allows to read or write parts of a page but writing or reading of complete pages is most efficient (as the fixed cost of addressing has to be paid only once for all data associated with the page). It is therefore reasonable to align the DataChunk size with the page size. S_P should be therefore a multiple of S_D.

$$S_P = aS_D = bS_B = bS_C$$

$$b \equiv 0 \ (mod\, a)$$

$$\forall\, a, b\, \epsilon\, 1N^+$$

Data Caching To increase the performance of the system, unencrypted DataChunks are cached. Write operations are cached and encryption is carried out after a certain number of DataChunks are accumulated. Likewise, if previously stored data must be read/modified

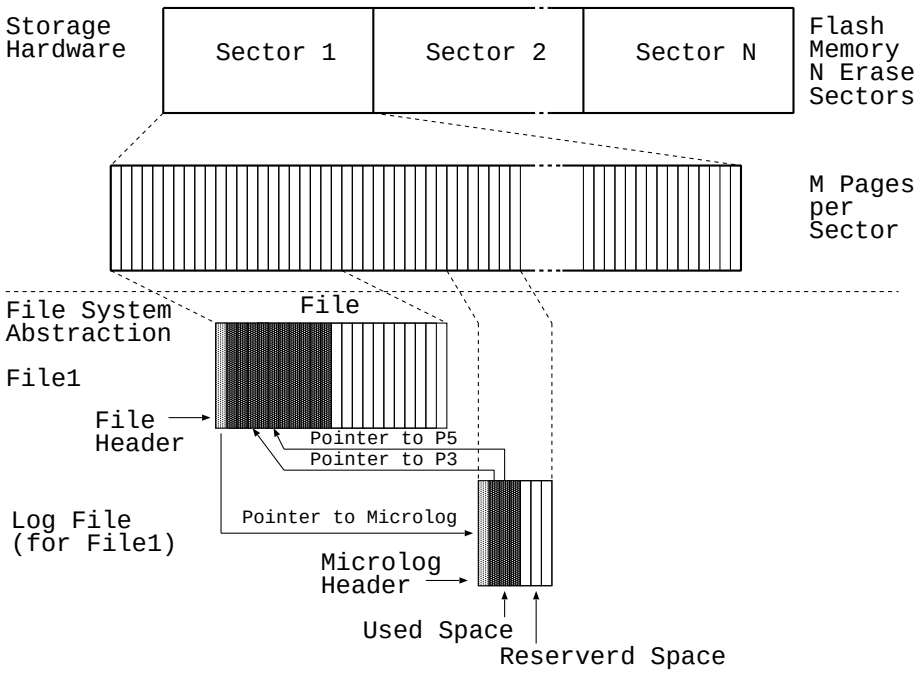


Figure 3.1: The Contiki Coffee File System (CFS)

the complete corresponding DataChunk is decrypted and cached. Increasing the size of the cache leads to better performance; however, with this approach the amount of unencrypted data present in the system would be bigger. Decreasing the size of the cache leads to better security, but at this time the system performance decreases. The number of unencrypted DataChunks N_D allowed in the system at any given time is a configuration parameter.

Key Management In the current implementation of our framework, encryption keys are pre-shared. Future implementations might be enhanced with dynamic key-management protocols such as Internet Key Exchange (IKE) [KHNE10].

3.1.3 Codo Implementation

Codo is implemented as an extension of Contiki's [DGV04] Coffee File System (CFS) [TDZV09]. CFS organizes files as a collection of similar sized pages (see Figure 3.1) that have generally the same size as the underlying flash memory pages[2]. For each new file,

[2]It is possible to map several Contiki file system pages into one flash memory page. However, this only makes sense if the flash memory hardware is able to support operations on parts of a page.

a header is created and a number of consecutive free pages are allocated. New data is directly written to the empty pages in the file. If pages with existing content are modified, a so called micro log file (also simply referred to as log file) is used. For modifications, a micro log file is created and linked with the original file which contains a sequence of log records that have the same size as a page in the original file. Each log record points to the original page in the file and contains the updated information. If data is accessed, the CFS checks first whether newer data is available in the log file before accessing the original file. After a certain number of changes, the log file is filled and it is *merged* with the original file to form a new consolidated file. The old file and micro log file are marked for garbage collection to be recycled. The micro log structure is used as the flash memory hardware does not allow us to overwrite pages directly. Before overwriting a hardware page, it is necessary to format and clear an entire erase sector containing many pages. This would be inefficient when used frequently and the use of a log file reduces erase sector formats to a minimum performed at convenient times by the CFS garbage collection. The CFS exposes standard functions such as `cfs_open()`, `cfs_write()`, `cfs_read()`, `cfs_seek()` and `cfs_close()` to the application for interaction with the file system.

3.1.3.1 CFS Optimisation

Codo makes use of the micro log as cache structure to hold unencrypted data. Thus, every read and write call involves the micro log structure and therefore it is important that it is performing efficiently.

Figure 3.2 shows the measured execution times of 10 consecutive `cfs_write()` using 16*byte* blocks, a log record size of 256*byte* and a log file size of 4 records. CFS_APP shows the execution time of CFS when appending to a file. In this case the microlog structure is bypassed and data is written directly to the file in the flash. CFS_MOD shows the execution time when writing to a file that has been modified at some point; in this case the log structure is used and execution times are significantly higher (13.5 times higher for the first write operation). With CFS_MOD it is checked if a log record for the page that data is written to exists already in the micro log file. If this is the

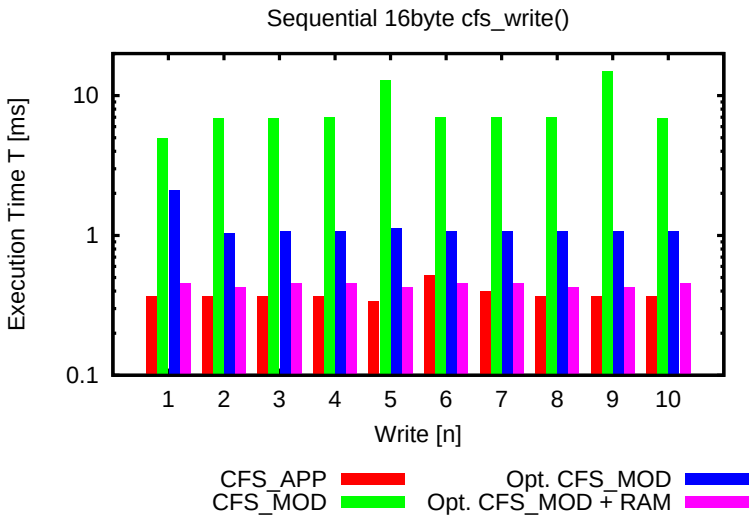


Figure 3.2: Execution time for sequential writes of 16*byte* blocks to CFS using a log record size of 256*byte*.

case the existing log record is read and then merged with the new data that should be written and the result is placed into a new log record in the micro log file. Obviously, when using a large log record size (e.g. 256*byte*) and writing small amounts of data (e.g. 16*byte*) large portions of log records remain unused. Furthermore, copying data from an existing log record to a new log record is time consuming. Finally, frequent costly merge operations are necessary as the log file fills quickly (merging is executed with writes 5 and 9 in the example).

To improve log file efficiency we optimised its execution and modified its behaviour slightly. If data has to be written to a log file but existing data in an existing log record does not have to be overwritten, the new data is simply added to the existing log record. This CFS optimisation is possible if the flash memory hardware allows partial writes to memory pages. The optimisation results in a significant performance gain (now the first write is 5.7 times more costly than CFS_APP).

In a final optimisation step, we place the log file structure in RAM rather than in the flash memory. This results in further improvements as can be seen in Figure 3.2 and write operations involving the log file structure are of comparable speed to direct flash write operations (The first write is now only 1.3 times more costly than CFS_APP). Using RAM for the log file structure is not without problems. Data loss is possible in case of a

Algorithm 1 A simple Contiki application program using CFS with and without Codo extension.

```
[01] PROCESS_THREAD(cfs_test_process, ev, data) {
[02]  PROCESS_BEGIN();
[03]  char buf[100];
[04]  char *filename = "msg_file";
[05]  int fd; int n=0;
[06]  fd = cfs_open(filename, CFS_READ);
[07] #ifdef CFS_CRYPT
[08]  while(n<sizeof(buf)) {
[09]   n+=cfs_read(fd,buf+n, sizeof(buf)-n);
[10]   if(n<sizeof(buf))
[11]        PROCESS_WAIT_UNTIL(ev == KEY_READY);
[12]  }
[13] #else
[14]  cfs_read(fd,buf, sizeof(buf));
[15] #endif
[16]  cfs_close(fd);
[17]  PROCESS_END();
[18] }
```

power failure and some platforms may not have RAM to spare for cache placement.

The outlined file system optimisations lead also to improvements of similar scale for read operations. We use the described optimised CFS variant for our Codo implementation.

3.1.3.2 Codo Extensions for CFS

To implement Codo with Contiki's CFS it is necessary to i) modify and extend function calls provided by the existing CFS library and to ii) modify and extend the behavior of internal CFS components. We detail these necessary modifications in the next paragraphs.

CFS Function Calls

Algorithm 1 shows a simple Contiki program that uses the CFS library. The definition in line 7, 13 and 15 is used to switch the API semantic between CFS with Codo (CFS_CRYPT) and standard CFS.

Without CFS_CRYPT a file is opened in line 6 using `cfs_open()` for reading which is indicated via the flag `CFS_READ`. In line 14 `cfs_read()` is used to read data from the file. `cfs_close()` (line 16) is used to close the open file.

With CFS_CRYPT, if data to be read using `cfs_read()` (line 9) is not yet available

in the micro log file (which acts as cache holding unencrypted DataChunks), the security manager component (see next section for details) is triggered to fetch the key required to decrypt the next DataChunk. The program leaves `cfs_read()` before completion of the read process and blocks on `ev == KEY_READY`. When the key is obtained by the security manager, it sends a signal to the waiting application process. When `cfs_read()` is now called again it will be able to decrypt the next data for which a key is now present. Multiple executions of `cfs_read()` with following `PROCESS_WAIT_UNTIL` may be necessary to complete one read as a sequence of different keys may be required. The call to `cfs_read()` with following `PROCESS_WAIT_UNTIL` (line 8 to line 12) can be combined within one *C* macro to hide the complexity of multiple function entries from the programmer.

`cfs_write()` is used in a similar way to `cfs_read()`.

`cfs_open()` supports the additional flag `CFS_NO_CRYPT` to indicate that a specific newly opened file should not be encrypted. Thus, the file system can hold encrypted and unencrypted files at the same time.

We add also two additional functions to the CFS API: `cfs_read_crypt()` and `cfs_write_crypt()`. These two functions can be used to read and write the encrypted data directly. If data is still in unencrypted form in the cache `cfs_read_crypt()` will perform encryption of the data. The security manager may have to be informed to fetch necessary keys and multiple calls to `cfs_read_crypt()` followed by `PROCESS_WAIT_UNTIL` might be necessary. These two functions are particularly useful for situations in which an encrypted file must be transported over the network to the sink or another sensor node. With these functions it is possible to avoid re-encryption of data for data transport and the already securely stored data can be directly placed in network packets.

We provide a `cfs_merge()`, which can be used to execute the processing costly merge of file and log file at a convenient time (for example, at times the system is idle).

`cfs_close()` is modified to ensure that a merge is executed to ensure that all unencrypted cached data is encrypted and stored in the file. `cfs_close()` may require multiple function calls with `PROCESS_WAIT_UNTIL` as keys may have to be organized by

the security manager for encryption.

CFS Components

Micro Log As described, the CFS uses the micro log files to handle flash memory read/write specifics. For the Codo implementation we modify the log file such that it becomes in addition a cache holding unencrypted DataChunks. In the standard CFS, the log file is used for modifying write operations. In our CFS extension, read and initial write operations also operate on the log file.

Whenever data is read from the file, it is first checked if the data is present in unencrypted form in the log file. If not, the key associated with the data is requested via the security manager component and, upon obtaining the key, the data is decrypted and transferred to the log file. New data is always written to the log file. When the maximum size of the log file is reached, the log file must be cleared and merged with the original file.

Security Manager The security manager is implemented as a Contiki thread that is responsible for i) generating new keys if needed ii) communicating with the sink to store and retrieve keys. The security manager has room to store exactly one key, which may be lost when a node is captured by an attacker. Keys are exchanged between sink and nodes using public key cryptography.

The CFS can ask the security manager via a function call for a key to a specific DataChunk of a file. This request contains three parameters: file_id, DataChunk_id and flags. file_id is the filename which is a unique identifier, DataChunk_id is the number of the DataChunk for which a key is required. flags indicates if the requested key is for a portion of the file that has never been used before. If this is the case, the security manager has two options. First, it can create the requested key, inform the file system and then transmit the key to the sink for storage. Second, it can send a request to the sink for a new key and, when a response arrives, inform the file system. If flags indicate that a key for a previously used DataChunk is needed the security manager must send a request for the key to the sink.

Cryptographic Functions For encryption/decryption we use AES in Counter (CTR) mode with 128*bit* key length provided by either by hardware (e.g. via the CC2420 radio chip present on many sensor node platforms) or the open source MIRACL [Cer] library if hardware support is not available.

3.1.4 Codo Evaluation

We evaluate the Codo implementation based on Contiki's CFS using a Tmote Sky sensor node. To evaluate system performance we analyse the execution times of the CFS function calls. Execution times are important indicators as they are a measure for system responsiveness and are directly proportional to a node's energy consumption. We use the CFS optimisations described in Section 3.1.3 and investigate performance with the log file residing in flash memory and RAM.

The cryptographic hardware support of the Tmote's CC2420 radio chip requires a minimum block size of $S_C = 16byte$. The Tmote provides an ST M25P80 flash memory with a page size of $S_P = 256byte$. We therefore select a DataChunk size of $S_D = 256byte$ to obtain a well matched system (see Section 3.1). For encryption/decryption we use AES in counter mode (CTR) with 128*bit* key length provided by either the CC2420 radio chip (CFS_CRYPT_HW) or the open source MIRACL [Cer] library (CFS_CRYPT_SW). The file system is set to use a log record size of $S_L = 256byte$ to match flash memory page size. Furthermore, the system is configured to use $N_L = 4$ log records which means that $S_L \cdot N_L = 1024byte$ of unencrypted data can be present on the system at any given point in time.

3.1.4.1 cfs_write() Performance

In this first experiment, a file of size 2048*byte* is written using a sequence of `cfs_write()` calls. With each `cfs_write()` call S_W bytes are written to the file system ($S_W \in \{16, 32, 64, 256\}$). The execution time of each `cfs_write()` call is measured. The experiments are repeated using the original CFS in append mode (CFS_APP), the original CFS in modify mode (CFS_MOD) (data is appended, but the file is assumed to be modified

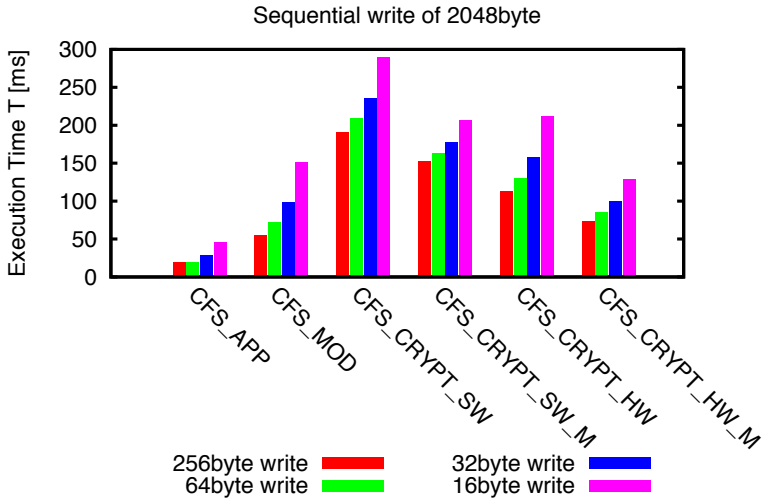


Figure 3.3: Writing of 2048*byte* in blocks of 256*byte*, 64*byte*, 32*byte* and 16*byte* using `cfs_write()`.

and therefore the log file is also used), Codo CFS with software (CFS_CRYPT_SW) and hardware supported (CFS_CRYPT_HW) encryption and Codo CFS with additional RAM supported log file (CFS_CRYPT_SW_M and CFS_CRYPT_HW_M). In this experiment, the Security Manager holds all 8 required keys for the 2048*byte* sized file locally and a performance penalty due to a key exchange protocol is not observed. If keys are exchanged over the network, key exchange times have to be added to the experimental results.

The experimental results are shown in Figure 3.3. Using CFS_APP and $S_W = 256byte$ the time to write all 2048*byte* to the file system is 18.3ms. In this mode, the file system does not make use of the log file structure and data is directly written to the file structure in flash memory. With CFS_MOD the time increases significantly to 54.8ms as the log file structure is involved in the writing process. Each write is directed to a log record in the log file in flash memory, and when all log records are filled a merge is executed to integrate log file and original file. CFS_CRYPT_SW and CFS_CRYPT_HW are functionally identical to CFS_MOD but when the log file is merged with the original file, encryption has to be performed. Thus, with CFS_CRYPT_SW and CFS_CRYPT_HW

write	CFS_APP	CFS_CRYPT_SW	CFS_CRYPT_HW	CFS_CRYPT_HW_M
1	2.29ms	4.06ms	4.12ms	1.43ms
2	2.29ms	3.14ms	3.17ms	1.40ms
3	2.29ms	2.93ms	3.02ms	1.40ms
4	2.29ms	2.93ms	2.93ms	1.37ms
5	2.29ms	168.67ms	90.39ms	63.45ms
6	2.29ms	3.14ms	3.14ms	1.40ms
7	2.29ms	2.99ms	2.96ms	1.37ms
8	2.29ms	2.93ms	2.96ms	1.37ms

Table 3.1: Writing of 2048*byte* in 8 blocks of 256*byte*.

execution times are 190.8ms and 112.7ms. With RAM caching, execution time reduces further to 151.6ms and 73.2ms (CFS_CRYPT_SW_M and CFS_CRYPT_HW_M).

The overall time of writing 2048*byte* to the file is not distributed equally among the 8 separate executions of `cfs_write()` with $S_W = 256byte$ (see Table 3.1). The first `cfs_write()` takes for CFS_CRYPT_SW and CFS_CRYPT_HW slightly more time than the following three as time to create the log file structure in flash memory is needed. The 5th write takes considerably more time than previous writes as the log file of size $N_L = 4$ is full and a merge must be executed before a log record can be written. During merge, encryption is performed, which takes significant processing time. The use of encryption hardware support improves encryption performance by 47%. When using CFS_CRYPT_HW_M, the first 4 write operations are faster then CFS_APP as data is written to the cache located in RAM.

When decreasing the write size S_W to 64*bytes*, 32*bytes* and finally 16*bytes* the overall time necessary to write the file of 2048*bytes* increases (see Figure 3.3). This is not surprising as each `cfs_write()` call is associated with additional overhead. For example, the overall time necessary to write file of 2048*bytes* length increases from 112.7ms to 211.6ms to when switching from $S_W = 256byte$ to $S_W = 16byte$ with CFS_CRYPT_HW. It has to be noted that CFS_CRYPT_HW_M outperforms CFS_MOD for $S_W = 16byte$. This means that under this condition Codo, which performs caching and encryption, outperforms the standard CFS when operating on files that have been modified.

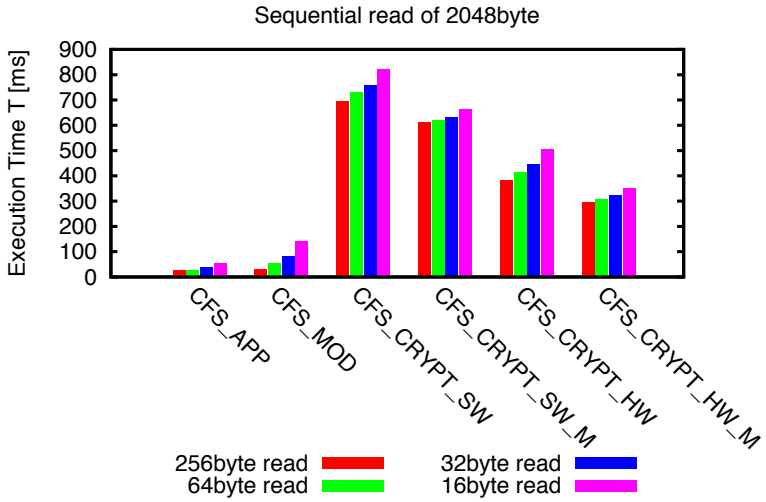


Figure 3.4: Reading of 2048*byte* in blocks of 256*byte*, 64*byte*, 32*byte* and 16*byte* using `cfs_read()`.

Summary Codo CFS is relatively expensive in comparison to CFS. However, CFS does not provide data confidentiality and this feature cannot be implemented at zero cost. For example, overall execution time for writing $S_W = 256byte$ increases with Codo CFS (CFS_CRYPT_HW_M) compared to CFS (CFS_APP) by a factor of 4. However, individual write operations that do not require merging and encryption are faster with Codo (CFS_CRYPT_HW_M) (By a factor of 1.6 for the first write with $S_W = 256byte$). Furthermore, within an application scenario, it might be possible to schedule costly merge and encrypt operations at times the system is idle and, thus, overheads for providing confidentiality may not impact overall system performance.

3.1.4.2 cfs_read() Performance

In this second experiment, the file created in the previous experiment (2048*byte* file size) is read using a sequence of `cfs_read()` calls. With each `cfs_read()` call, S_W bytes are read from the file system ($S_W \in \{16, 32, 64, 256\}$). The execution time of each `cfs_read()` call is measured. The time necessary for reading the complete file is shown for all file system modes in Figure 3.4.

read	CFS_APP	CFS_CRYPT_SW	CFS_CRYPT_HW	CFS_CRYPT_HW_M
1	3.14ms	232.03ms	134.31ms	100.56ms
2	3.11ms	38.24ms	18.77ms	15.66ms
3	3.17ms	38.36ms	18.80ms	15.66ms
4	3.11ms	38.30ms	18.86ms	15.69ms
5	3.17ms	231.45ms	133.88ms	100.22ms
6	3.11ms	38.24ms	18.77ms	15.63ms
7	3.11ms	38.30ms	18.83ms	15.66ms
8	3.14ms	38.33ms	18.86ms	15.78ms

Table 3.2: Reading of 2048*byte* in 8 blocks of 256*byte*.

Using CFS_APP and $S_W = 256byte$ the time to read all 2048*byte* is 25.1ms. With CFS_MOD the time increases to 30.6ms. With CFS_CRYPT_SW and CFS_CRYPT_HW execution times are 693.2ms and 381.1ms; with CFS_CRYPT_SW_M and CFS_CRYPT_HW_M times are 608.8ms and 295.1ms. Again, the overall time of reading 2048*byte* to the file is not distributed equally among the 8 separate executions of `cfs_read()` with $S_W = 256byte$ (see Table 3.2). The first `cfs_read()` of CFS_CRYPT_SW, CFS_CRYPT_HW and CFS_CRYPT_HW_M requires a merge as this first read is performed after the previous experiment in which the file was written and the log file structure was filled. Also the 5th read requires a merge as the log file is filled. All 8 reads require reading from the flash memory followed by decryption followed by placing of the decrypted data in the log file structure.

When decreasing the read size S_W to 64*bytes*, 32*bytes* and finally 16*bytes* the overall time necessary to read the file of 2048*bytes* increases as shown in Figure 3.4. However, the times necessary for individual `cfs_read()` calls have an uneven distribution. For example, for $S_W = 16byte$ with CFS_CRYPT_HW_M the first read requires 99.8ms as it includes a merge, decryption of 256*byte* of data and placement of this data in the cache (the log file). The next 15 read operations require 0.5ms each as the decrypted data is now available in the cache. The 16th operation requires 14.8ms as a new block of 256*byte* is decrypted and moved to the cache.

Summary Reading a securely stored file requires considerable more effort than reading the file from the original CFS. For example, the overall time to read a 2048*byte* file

in 256*byte* blocks with CFS_CRYPT_HW_M increases by a factor of 11.7. However, not every read operation is equally expensive. For example, when using a read size of $S_W = 16byte$ with CFS_CRYPT_HW_M read operations increase by a factor of 1.3; only when merge and/or decryption operations are necessary read operations are much more costly.

3.1.4.3 Cache Performance

Instead of using Codo, which enables caching of unencrypted data, one could use a simple solution (CFS_SIMPLE), which encrypts/decrypts data before calling `cfs_write()`/`cfs_read()` of the original Contiki file system. CFS_SIMPLE would only be usable if it is ensured that data is accessed in whole blocks that can be encrypted/decrypted in full. Hence, CFS_SIMPLE is only useful to provide a baseline for comparison here but it is not a practically usable alternative.

Writing 256*bytes* of data using CFS_SIMPLE_HW takes 11.7ms (9.4ms for hardware supported encryption and 2.3ms for writing to flash memory) when writing to a file that has not been modified yet and hence the log file structure is not in use. In comparison, CFS_CRYPT_HW_M requires only 1.4ms as the data is written to the cache in RAM. A performance penalty only occurs for writes when the log file structure is full and a merge has to be performed (see previous paragraphs).

We note a similar performance difference for read operations. CFS_SIMPLE_HW requires 12.5ms while CFS_CRYPT_HW_M requires only 1.4ms if the data is found in the cache structure.

The performance difference between CFS_SIMPLE_HW and CFS_CRYPT_HW_M diminishes when handling smaller amounts of data in each read and write operation. This is due to the fact that then encryption/decryption times are comparable to times necessary for flash read/write operations. For example, when writing 16*bytes* CFS_SIMPLE_HW requires 1.2ms while CFS_CRYPT_HW_M requires 0.6ms.

Summary The caching functionality provides a performance benefit for individual read and write operations. Sensor network applications that access files sequentially

(e.g. writing a continuous log file) may benefit from the increased read/write speed effective for most operations. However, applications that really benefit from the cache functionality are applications that access the same data in a file multiple times. For example, some applications may record sensor data and then perform periodically complex data processing which requires multiple reads of the previously recorded data.

Following from this work, Codo is combined with secure communication in WSNs, more specifically in Internet of Things (IoT). This new mechanism will be explained in the next section.

3.2 Combined Storage and Communication for Internet of Things

The IoT is becoming a reality and vast numbers of smart objects are interconnected via the Internet Protocol (IP). A number of applications in this context handle sensitive information. For example, smart objects may be used for patient monitoring in hospitals, implementations of security systems in airports or to monitor crucial business processes in factories. Thus, security mechanisms are required to ensure confidentiality, integrity and authenticity of the collected information.

Due to resource limitations of smart objects it is not feasible to use the existing IP protocol throughout the entire IoT. IP header compression, as defined in the IPv6 Over Low Power Wireless Personal Area Networks (6LoWPAN) [MKHC07] framework, is used in wireless IEEE 802.15.4 networks which smart objects generally use for interconnectivity. 6LoWPAN header compression and decompression is carried out by gateway nodes when relaying packets between IEEE 802.15.4 networks and the existing IP network infrastructure.

As the IoT relies on the established and tested IP protocol it is reasonable to also use security mechanisms defined in this context. The Internet Protocol Security (IPsec) [SK05] framework defines security mechanisms for IP networks and it is supported by

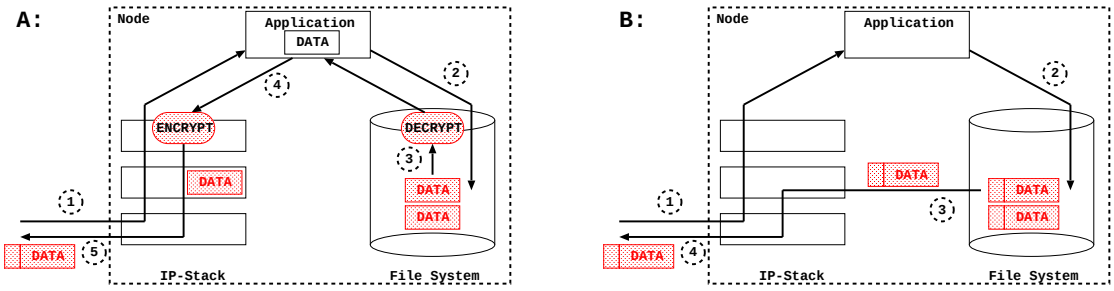


Figure 3.5: *A: Traditional Operation:* 1 - Data is requested from the node. 2 - The application forwards the request to the file system. 3 - The data is decrypted and passed to the application. 4 - The application sends data for transmission to the IP stack which secures the data. 5 - The data is transmitted.
B: Combined Secure Storage and Communication: 1 - Data is requested from the node. 2 - The application forwards the request to the file system. 3 - The secured data is directly passed into the IP stack. 4 - Data is transmitted without cryptographic processing.

nearly all hosts currently in use. A definition of IPsec 6LoWPAN extensions [RDH+12] exists which allows smart objects to participate in IPsec secured communication. Thus, secure communication in the IoT using standardised mechanisms is feasible.

Smart objects now provide vast amounts of storage space due to the recent advances in flash memory technology. IoT applications rely on this storage space in order to improve system performance [TD11]. It is therefore becoming more important to not only secure communication but also to protect sensitive data while it is stored on smart objects. Various secure storage solutions exist that can be used to protect data on nodes.

The previously outlined secure communication and storage solutions have been developed individually. It is not taken into account that tasks such as key exchange or cryptographic processing are executed for both system components. Thus, in many situations, cryptographic work performed by smart objects is unnecessarily carried out twice or more. Given that smart objects are very resource limited devices it is desirable to prevent such process duplication. Freed resources may be used to reduce hardware complexity, improve energy consumption or to add additional application features.

We address the previously outlined shortcoming of existing solutions and provide a design of a combined secure storage and communication framework that allows us to reduce security related processing on smart objects (see Fig 3.5). In particular, we consider the IP, 6LoWPAN and IPsec standards as the base for our work. We believe that

a standard compliant solution is more desirable than a proprietary system. Furthermore, it is safer to build on tested and trusted security mechanisms rather than designing an entirely novel mechanisms. Data is stored securely on the flash file system such that it can be directly used for secure transmission. This is not a trivial task as packet header content of future transmissions must be considered when securing data for storage. We show in this work that an IP-based combined secure storage and communication solution is possible and that this can save up to 71% of a node's security related processing effort. A cost in regards to additional storage space is incurred as a result of the secure storage; however, given that smart objects can now provide ample amounts of storage space we do not see this as limiting factor. The specific contributions of this work are:

- The definition of a framework for combined secure storage and communication for IP/6LoWPAN networks.
- An implementation of the framework for the Contiki operating system.
- A detailed evaluation of the performance gains of the framework.

3.2.1 The Secure Storage and Communication Framework

Our proposed secure storage and communication framework is based on the established IPv6/6LoWPAN protocols. IPv6/6LoWPAN defines IPsec/ESP (Encapsulating Security Payload) that provides encryption and authentication of transmitted data packets. We use the same cryptographic methods and data formats defined by ESP for data processing before storage. This requires us to store not only data but also all header information that is involved in the cryptographic processing. Encrypted data must be stored in ESP compatible form such that requested data can be transmitted over the network without further cryptographic processing. This requires us to anticipate content of communication protocol header fields such as IP destination addresses, sequence numbers and checksums at storage time. As IPsec is the base for communication and storage, the existing key exchange mechanisms defined for IPsec can be reused for the storage element of the framework.

The next subsection describes IPsec/ESP usage in 6LoWPAN networks. This represents the communication element of our framework. More detailed information about 6LoWPAN and IPsec/ESP are given in Chapter 2. Thereafter follows a description of the storage element of the framework. We then briefly discuss application layer protocols that may be used with the framework and describe our Contiki-based implementation. Finally we discuss expected performance gains and cost in terms of storage overhead and provide a security analysis.

3.2.1.1 Communication Component

IPv6 uses IPsec [SK05] to secure IP communication between two end points. IPsec is a collection of protocols that include Authentication Header (AH), which provides authentication services, and ESP, which provides both authentication and privacy services. A suite of encryption and authentication algorithms are also defined. A node keeps track of Security Associations (SA) that specify how IP flows are treated in terms of security. Each SA holds a Security Parameter Index (SPI), which is a 32-bit value used by a receiver to identify the correct SA.

In an ESP [Ken05] packet data (for example, a UDP packet), padding, pad length and next header information are encrypted. All header information may be authenticated using the optional Integrity Check Value (ICV). In an 802.15.4 network an ESP header will not be transmitted directly. Its compressed form as defined by 6LoWPAN is used instead to reduce header overheads.

6LoWPAN defines header-compression mechanisms. LOWPAN_IPHC is used for IP header compression and LOWPAN_NHC for the next-header compression. The NH field in LOWPAN_IPHC when set to 1 indicates that the next header following the compressed IPv6 header is encoded with LOWPAN_NHC. LOWPAN_NHC has a length of 1 or more octets, where the first variable length bits identify the next header type and the remaining bits are used to encode header information. Currently, 6LoWPAN defines LOWPAN_NHC for the IP extension header (LOWPAN_NHC_EH) and the UDP header (LOWPAN_NHC_UDP). A definition for ESP encoding (LOWPAN_NHC_ESP) is

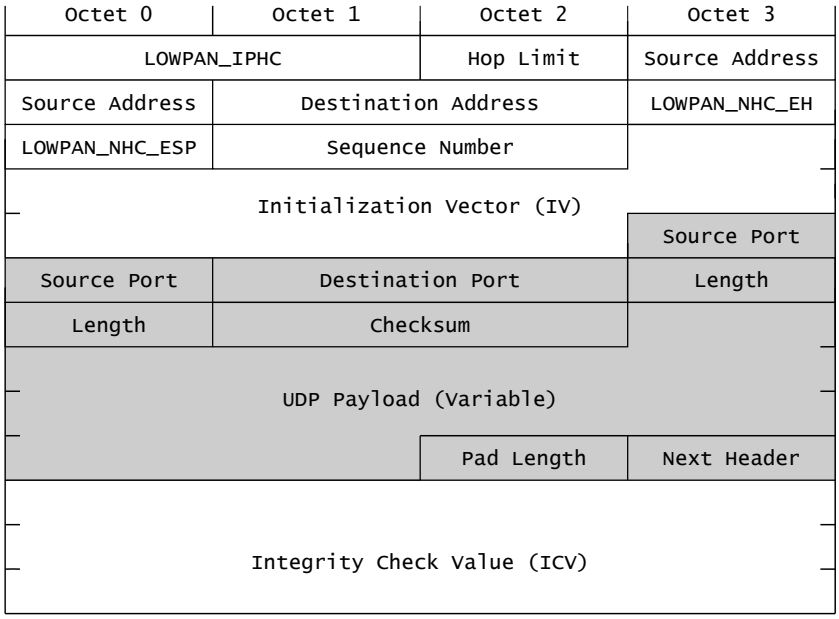


Figure 3.6: A compressed and ESP secured IPv6/UDP packet.

provided in [RDH+12].

Figure 3.6 shows a UDP/IP packet secured with compressed ESP. An Initialization Vector (IV) may be carried in the ESP packet if the selected encryption algorithm requires transmission of this information with every packet. The shaded portion represents encrypted data. Authentication can be provided using the ICV.

3.2.1.2 Storage Component

Data is stored securely such that it can be transmitted as ESP compliant packets on request without additional cryptographic processing. This requires storage of all cryptographically processed elements of the ESP packet within the file system. ESP header elements that are not cryptographically processed and can be constructed with little effort when data is requested and therefore do not have to be stored.

Data is stored as blocks representing the shaded part (and the ICV if authentication is required) shown in Figure 3.6. If data stored within a block is requested the block is read from the file system and the full packet, as shown in Figure 3.6, is assembled and transmitted. The receiver may only be interested in part of the received data and some undesirable transmission overhead may occur. However, typical applications will require

bulk data transfer (large parts of a file) in which case such overheads do not occur. For example, for further data analysis an application may request recorded sensor samples within a particular time frame or, for performance debugging purposes, recorded link quality metrics over a longer time period may be requested.

Some stored information is dependent on the communication relationship. At the time of storage, assumptions regarding the forthcoming communication relationship must be made in order to enable cryptographic processing. Elements to be considered are:

- *UDP Header:* A UDP header is stored within the encrypted ESP payload. Assumptions regarding destination and source UDP port must be made at time of storage. The destination IP address of packets is used within the IPv6 UDP checksum calculation. Thus, IP source and destination address assumptions must be made as well.

- *Initialization Vector (IV):* The IV (if required) is used for ESP encryption. Most protocols allow a counter mode where the IV for each packet is constructed by adding a transmission sequence number to an initial IV.

- *Sequence Number (SN):* The ESP header includes a sequence number. This sequence number is not encrypted but it is included in the ICV calculation. If ESP authentication is used a sequence number must be selected at time of storage in order to generate a ICV for storage alongside the data.

UDP Header Construction: A UDP header has to be prepared at time of data storage. The header consists of 4 fields of $2byte$ length: Source Port, Destination Port, Length and Checksum.

The Length field is defined by the amount of data contained in the UDP packet. To reduce packet overheads the amount of data contained in each UDP packet is selected such that the maximum 802.15.4 frame size of $127byte$ is utilised.

The selection of a Source and Destination Port is not problematical. It can be assumed that well known ports can be used for data retreival.

The calculation of the Checksum field is challenging. The checksum is mandatory in IPv6 and is calculated using a pseudo header. This pseudo header contains the IP Source Address, the IP Destination Address, UDP Length and IP Next Header field. As the IP Destination Address is included assumptions regarding the IP address requesting stored information must be made. The checksum is calculated as the 16-bit one's complement of the one's complement sum of the pseudo header, the UDP header, and the data.

It is a reasonable assumption that a particular host is used most of the time to request information from nodes (e.g. the sink). The IP address of this node may be used for storage preparation.

In some cases data may be requested from a different node and the IP destination address used for checksum calculation does not match the destination of the data requestor. In this situation it is possible to correct the checksum in a way that does not require the decryption and encryption of all of the data again. Thus, performance is reduced as part of the stored data must be cryptographically processed before transmission but it is still beneficial in comparison with a system that does not combine secure storage and transmission (see Section 3.2.2).

ESP can use encryption algorithms which operate on blocks (e.g. AES using 16*byte* blocks). It is possible to decrypt only the first block of a larger stored ESP packet which will contain the UDP header and its checksum. Since the UDP checksum algorithm is a simple summation checksum re-calculation is trivial. By substituting the old destination address for the new destination address, a new checksum can be calculated. Now the first block of the ESP packet can be encrypted and it is ready for transmission to an alternative destination.

IV Construction: The IV does not have to be stored in the file system together with the encrypted ESP fields. An initial IV can be used and the storage block number is added to construct the IV.

Sequence Number Construction: If authentication is required it is possible to also store the ICV. As the ESP header includes a sequence number which is included in the ICV calculation, it is necessary to predict at storage time what sequence numbers will be

used during communication.

Data belonging to a file is stored as sequence of ESP encrypted blocks and we can use the block number as ESP sequence number. IPsec allows us to reset the sequence number counters at the start of a communication relationship by establishing a new SA. Thereafter, data from the file can be delivered sequentially. In this setting we ensure that the communication uses sequence numbers that were selected at time of data storage.

3.2.1.3 Framework Usage

Application Layer Protocol: Nodes store data securely which may be requested by Internet hosts. Stored data has an application specific semantic. For example, sensor values may be stored as a 4*byte* sensor value together with a 4*byte* timestamp and 2*byte* sequence number. Nodes execute a storage application that is able to respond to queries such as "send sensor samples recorded between 12:00:00 and 13:00:00". A host executes a storage application that is able to send these requests and to process arriving data. Host and node storage applications use UDP for communication. Similar to the well known File Transfer Protocol (FTP) protocol, separate flows are used for command and data transfer which makes different IPsec security settings (including keys and security mechanisms) for both channels possible.

Security Configuration: The IPsec SA defines how data flows are protected. The SA holds secret keys, encryption algorithm descriptions and IP addresses to identify flows. Each SA holds an SPI, which is a 32-bit value used by a receiver to identify the correct SA.

If each file should be encrypted with a different key it is necessary to specify distinct SAs that each use a unique SPI. The SPI is transmitted in the 6LoWPAN header in compressed form (See Section 3.2.1.1). However, compression is only possible when the default SPI value is used; otherwise SPI information must be carried within the packet. Thus, the most frequently used file should use the default SPI in order to improve efficiency. In a practical setting this is not an issue as most nodes are using a single large file for storage of sensor data.

3.2.1.4 Implementation

We implemented the outlined framework for the Contiki [DGV04] operating system. The implementation uses Contiki's μIP stack with 6LoWPAN/IPsec extensions as defined in [RDH+12] as the communication component. The storage component uses Contiki's Coffee File System (CFS) [TDZV09] with Codo to provide file system security extensions. The μIP stack was modified in order to enable direct passing of ESP encrypted packets from the file system to the communication stack. On the host side we used a standard Ubuntu Linux host.

For encryption/decryption we used AES in Counter (CTR) mode, with a 128*bit* key, in either hardware (e.g. via the CC2420 radio chip present on many sensor node platforms) or the MIRACL [Cer] library if hardware support is not available. If authentication is required, AES-XCBC-MAC-96 is used to calculate the necessary ICV (provided via cryptographic processor or the MIRACL library).

The maximum 802.15.4 payload is 127*byte* and the available MAC layer payload size is 102*byte*. As seen in Figure 3.6, 7*byte* are required for the compressed 6LoWPAN header, 12*bytes* are required for the compressed ESP header fields, 2*bytes* are required for the ESP trailer fields, 12*bytes* are required for the ICV if it is used and 8*bytes* are required for the UDP header. This leaves a maximum payload of 61*byte*. The AES algorithm requires a minimum block size of 16*byte*. Thus, the maximum feasible amount of data that can be stored per block before fragmentation must occur is 54*byte*. Storage blocks contain 64*byte* of encrypted data (8*byte* encrypted elements of the UDP header, 2*byte* encrypted ESP trailer and 54*byte* payload). Other feasible payload sizes are 6, 22 and 38. To avoid padding, an application should align write operations with these payload sizes.

At this point in time, our Contiki IPsec implementation does not support key exchange mechanisms such as the IKE protocol. Keys are set manually before deployment. However, for most application scenarios this would not be an issue limiting the framework's usability.

3.2.1.5 Security Discussions

In this section we briefly discuss the security of the combined storage and communication system. We consider key management, cryptographic algorithms, message encryption and message authentication. In particular, we determine whether the combination of secure storage and secure communication provides weaker security than systems treating both subsystems individually.

Confidentiality - Communication is secured using IPsec's ESP procedures. The solution does not deviate from procedures defined in the IPsec framework. An attacker with access to the communication channel has access to the same information as an attacker on any other ESP secured communication. If we consider IPsec a secure solution, the provided solution can be considered secure as well.

Our implementation uses AES in CTR mode with 128*bit* keys. The best known Advanced Encryption Standard (AES) attack for this key length is four times better than exhaustive search [BKR11], and does not adversely affect its security.

Integrity and Authentication - When authentication is required, the ICV is calculated and appended to the ESP. Here we have to balance security and performance needs. Storing the ICV along with the ESP will ensure data integrity and authentication for storage and communication. However, when storing ICVs along with the encrypted payload it is necessary to select sequence numbers at the time data is stored. Hence, sequence numbers are predictable and will repeat when stored file content is transmitted repeatedly. Thus, protection against replay attacks in the communication channel is weakened. On the other hand, we will have performance gains as the ICV does not have to be computed at transmission time. If we decide to calculate the ICV before each transmission the replay protection is strongly enforced while the performance gains are reduced and stored data is missing integrity and authentication data.

To provide both strong data integrity and relatively weaker anti-replay functionalies when the ESP authentication field (ICV) is also stored in flash memory, sequence numbers should be in order before calculating ICVs for all the stored packets in a file, and the sender's and receiver's counter should be reset (by establishing a new SA) prior to the

transmission of a file.

Storage - Data is stored in the same format as it is later transmitted. An attacker with access to the file system has the same information available as an attacker with access to the communication channel. If transmitted information secured using ESP is considered to be secure then information stored in the file system must be considered secure as well.

Key Management - Data in flash memory is secured using the same key that is later used for communication. Hence, transmission of the same stored data requires usage of the same key on the communication link. It is not possible to negotiate a fresh key for each communication relationship compared to when IPsec is used on its own. However, many practical IPsec deployments use pre-shared fixed keys so we consider this a secure option.

Similarly, if multiple nodes have to be able to access the same stored information they will also have to use the same key for communication. This is similar to practical situations where IPsec is used with a single pre-shared key.

The proposed system has difficulties with revocation of keys; if a new key is selected data already stored in the flash file system must be re-encrypted, which is costly on resource constrained systems.

3.2.2 Evaluation

In this section we first discuss the costs in terms of storage overhead that are associated with the proposed scheme of combined storage and communication. Thereafter we analyse the processing performance and energy consumption gains associated with our scheme. We use our Contiki implementation for the Telos B platform.

3.2.2.1 Storage Overheads

Storing encrypted data together with the ESP fields that demand cryptographic processing requires additional storage space compared to a solution which would only store encrypted data.

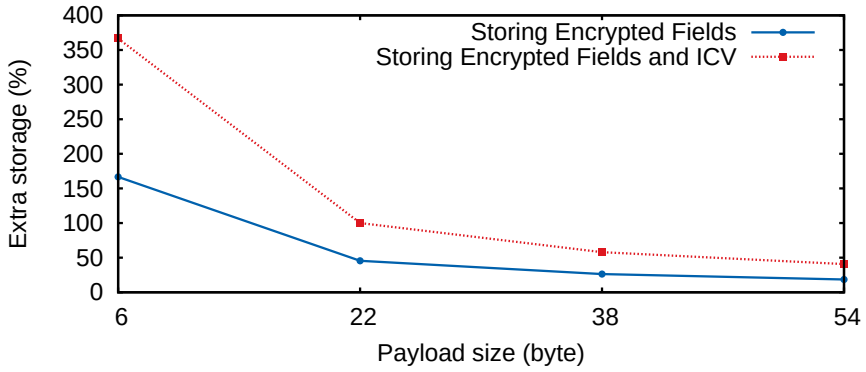


Figure 3.7: Storage overheads for different payload sizes.

If only encryption is used an extra $10byte$ per stored data block is required. Thus, it is better to store large blocks (large ESP packets) as this reduces the overhead. Figure 3.7 shows the overhead in dependency of the payload size. We show overheads for payload sizes which align with the AES encryption block size of $16byte$ and that do not require fragmentation when transmitted.

If authentication is also required then overheads increase as the ICV data of $12byte$ has to be stored alongside the other encrypted information.

The results show that the proposed framework reduces the effective storage size of the available flash storage space on nodes by 40.7% when using a payload size of $54byte$ and use of ICV. However, if we consider a common flash memory size of $16GB$ in which a $10byte$ sensor reading is recorded every minute the time until the storage capacity is exceeded is reduced from $3266years$ to $1329years$. Both values are acceptable in any deployment context and it can be concluded that the necessary storage overhead is not a limiting factor of the proposed framework.

3.2.2.2 Performance Gains

The combined storage and communication framework provides performance improvements. To analyse performance benefits in detail we use 4 different experiments. In all 4 experiments, ESP encryption and authentication is provided. The different experiments are used to show increasing performance benefits with increasing integration of storage and communication. *Experiment A* uses a system without the combining storage and

communication. In these experiments (*Experiment B, C, D*) the framework is used in different configurations.

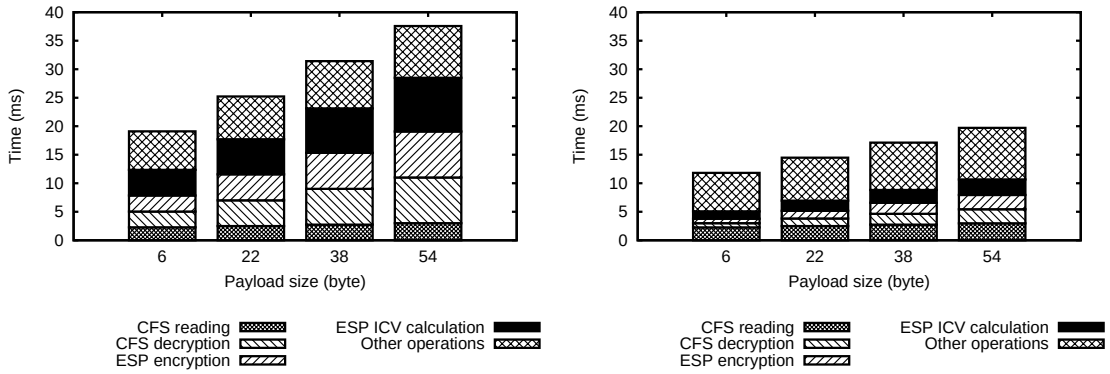
	Encryption	Authentication	UDP checksum re-calculation
Experiment A	individual	individual	-
Experiment B	combined	individual	not required
Experiment C	combined	individual	required
Experiment D	combined	combined	not required

Table 3.3: Experiment setup details used for evaluation. All experiments use ESP encryption and authentication. The combined storage and communication framework is used for different aspects. UDP checksum re-calculation is assumed in some settings.

Experiment A is the baseline experiment where data is read from flash memory, decrypted and re-encrypted for IPsec conformant transmission. In addition, authentication is provided and an ESP ICV is constructed. In *Experiment B*, ESP encrypted fields are stored in the flash memory and can be directly transmitted upon request. The ICV for authentication is still constructed at transmission time. *Experiment C* differs from *Experiment B* in terms of UDP checksum calculation. A non-matching IP address was used at storage time and a re-calculated before transmission is necessary. In *Experiment D*, ESP encrypted fields and the ESP authentication field (ICV) are stored in flash memory and can be transmitted directly upon request. All of the experiments are carried out with both software and hardware encryption. Table 3.3 summarises the experiment settings.

Experiment A: Baseline experiment

In this experiment, data is read from a file and sent using conventional methods. ESP conformant AES encryption is used for the storage component and communication component to allow for comparison with the other experiments. Payload data is read in blocks of 6, 22, 38 and 54$bytes$. The payload is decrypted and then re-encrypted for IPsec transmission. ICV authentication data is constructed before transmission. Decryption is carried out within the Contiki CFS; encryption and ICV calculation is carried out within the Contiki μIP stack.



(a) With software encryption. **(b)** With hardware encryption.

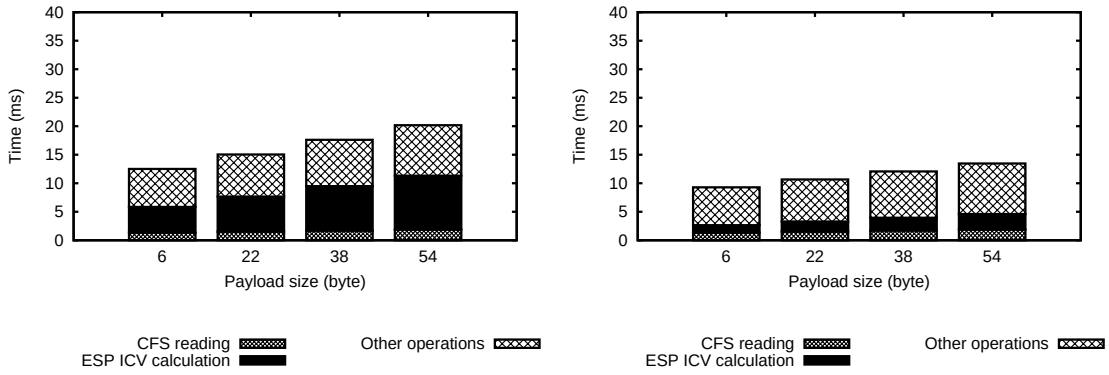
Figure 3.8: Duration of different operations involved in preparing single packet for transmission with software and hardware encryption.

Figure 3.8 shows the time that is necessary on a node to process one payload. The processing time is measured from the start of the file system read operation to the completion of the packet transmission. The total processing time is broken down to show the contribution of significant individual operations:

- *CFS reading* is the time required to read data from the file system.
- *CFS decryption* represents the time necessary to perform data decryption.
- *ESP encryption* represents the time necessary to encrypt the ESP payload.
- *ESP ICV calculation* is the time required to produce authentication data.
- *Other operations* summarises the duration of all other operations.

Figure 3.8a shows the processing duration breakdown when cryptographic processing is carried out in software. The total time to prepare a single packet is 19.1ms, 25.2ms, 31.4ms and 37.6ms for 6$byte$, 22$byte$, 38$byte$ and 54$byte$ payload data, respectively. CFS reading time is 8%, CFS decryption time is 21.3%, ESP encryption time is 21.5% and ESP ICV calculation time is 25.1% of the overall processing time for 54$byte$ payload.

Figure 3.8b shows the duration of operations when using hardware supported cryptographic processing. Total times for preparing a single packet are 11.8ms, 14.5ms, 17.1ms



(a) With software encryption. **(b)** With hardware encryption.

Figure 3.9: Duration of different operations involved in preparing single packet for transmission with software and hardware encryption when storing ESP encrypted fields.

and 19.7ms for the different payload sizes. CFS reading time is 15.1%, CFS decryption time is 12.5%, ESP encryption time is 12.7% and ESP ICV calculation time is 13.8% of the overall time when preparing 54$byte$ of data.

Enabling hardware support improves performance by 38.1%, 42.6%, 45.5% and 47.5% for 6$byte$, 22$byte$, 38$byte$ and 54$byte$ payloads, respectively.

The experiments show that when a 54$byte$ payload is transmitted processing the node spends 67.9% of the preparation time on cryptographic processing (software supported). This cryptographic processing time can be avoided by the proposed framework as we show in the following experiments.

Experiment B: Storing ESP fields

In this experiment, ESP encrypted fields are stored in flash memory with the payload data. 6, 22, 38 and 54$byte$ payloads are used. As additional stored ESP fields are 10$byte$ length 16$byte$, 32$byte$, 48$byte$ and 64$byte$ must be read from the file system. Compared to *Experiment A* CFS decryption and ESP encryption is now not necessary, so processing time is saved.

Figure 3.9a shows the duration for different operations when preparing a single packet for transmission with software encryption. Total times for preparing a single packet are

12.5ms, 15ms, 17.6ms and 20.2ms; and improvements in system performance when it is compared to the baseline experiment with software encryption are 34.6%, 40.4%, 43.9% and 46.3%.

Figure 3.9b shows processing times when using hardware encryption. Total times for preparing a single packet are 9.3ms, 10.7ms, 12.1ms and 13.5ms; and improvements in system performance when it is compared to the baseline experiment with software encryption are 51.3%, 57.7%, 61.6% and 64.1%.

It is notable that the CFS read time is less than in *Experiment A* even though more data has to be read from the file system (e.g. instead of 54$byte$, 64$byte$ are read as ESP information is included). This is due to the fact that elements of the CFS can be bypassed when directly reading encrypted data for transmission.

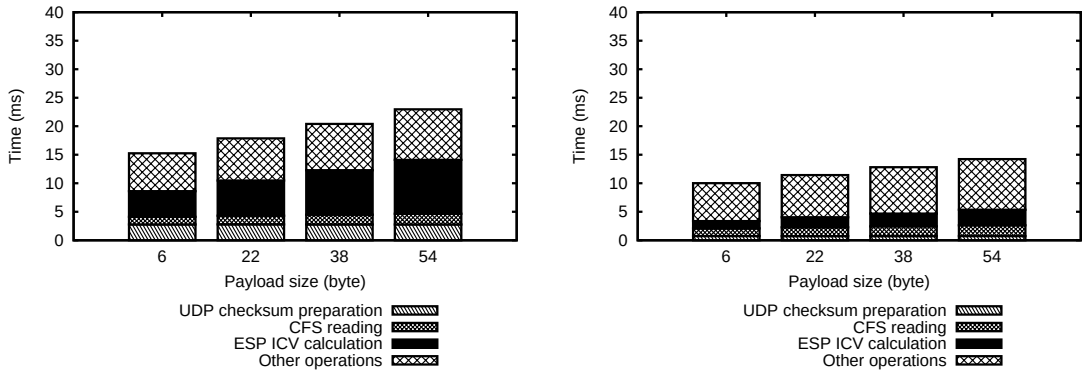
Experiment C: Using a non-matching IP address

This experiment is similar to *Experiment B*. The difference is the IP address of the destination when carrying out ESP encryption for storage. The UDP checksum enclosed in ESP packets must be corrected before transmission. The time necessary to perform decryption of the 16$byte$ block containing the checksum, its correction and encryption of the 16$byte$ block containing the corrected checksum is referred to as *UDP checksum preparation*.

Figure 3.10a shows results using software encryption. The total time for preparing a single packet is 15.3ms, 17.9ms, 20.4ms and 23ms; and improvements in system performance when it is compared to the baseline experiment with software encryption are 20.1%, 29.1%, 35.1% and 38.9% for the different payload sizes.

Figure 3.10b shows results when using cryptographic hardware support. Total times are 10ms, 11.4ms, 12.8ms and 14.2ms; and improvements in system performance when it is compared to the baseline experiment with software encryption are 47.6%, 54.7%, 59.2% and 62.1% for 6, 22, 38 and 54$byte$ data.

The correction of the UDP checksum, which may be necessary in cases we cannot anticipate the endpoint to which stored data must be delivered, is not very costly. For a



(a) With software encryption. **(b)** With hardware encryption.

Figure 3.10: Duration of different operations involved in preparing single packet for transmission with software and hardware encryption when using a non-matching IP address.

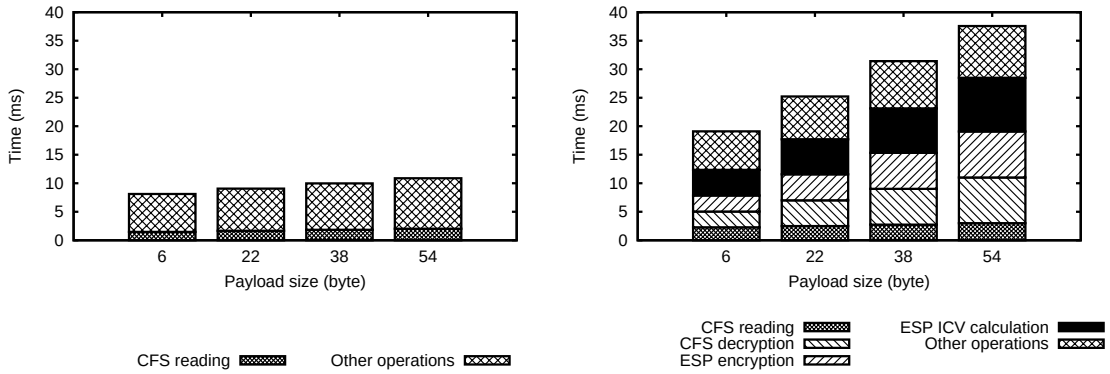
54*byte* payload using hardware support the performance gain is only reduced from 64.1% to 62.1%.

Experiment D: Storing ESP and ICV fields

In this experiment all options of the proposed framework are in use. Data is stored in ESP compatible form alongside the ICV authentication data. In this case no encryption processing is required when data is requested, and thus processing times are independent from cryptographic algorithm implementations (hardware or software). The results are shown in Figure 3.11a. For direct comparison we again show the results of *Experiment A* (software encryption) in Figure 3.11b.

In this experiment, ESP encrypted fields and ESP authentication field (ICV) are stored in flash memory together with the payload data. As encrypted fields have a length of 10*bytes* and the authentication field (ICV) is 12*bytes* long, blocks of 28*byte*, 44*byte*, 60*byte* and 76*byte* have to be read for different payload sizes.

Compared to the previous two experiments, CFS read times increase as the additional ICV has to be read. Total time for preparing a single packet is 8.1ms, 9.1ms, 10ms and 10.9ms. Improvements in system performance when compared with the baseline experiment with software encryption are 57.5%, 64.1%, 68.3% and 71% for 6, 22, 38 and



(a) Combined storage and communication

(b) Individual security processing

Figure 3.11: Duration of different operations necessary to prepare a single packet for transmission when using the combined storage and communication framework and when using individual storage and communication security solutions.

54*byte* payload data.

These results show that the proposed framework of combined storage and communication can achieve significant performance gains. When the framework is used, requested data can be delivered approximately 3 times faster as cryptographic processing is not required at the time when data is prepared for delivery.

3.2.2.3 Energy Consumption

We have shown that combining secure storage and communication reduces processing time on sensor nodes. However, it is not immediately clear if savings in processing time translate to energy savings as the proposed mechanism changes usage patterns of hardware components such as flash memory and hardware encryption.

We therefore compare energy consumption of the conventional storage method with our combined storage and communication method. We use the setups previously described as *Experiment A* and *Experiment B*. We use energy consumption values for CC2420 radio operations and ST M25P80 flash operations from the Tmote Sky datasheet [Sky06], and for CC2420 hardware cryptographic support from [ZDC11].

If 54*byte* data is required to be stored and transmitted later using the conventional method, 54*byte* data has to be encrypted and written to the flash memory for storage;

54*byte* data has to be read from the flash memory and has to be decrypted; 64*byte* data has to be encrypted in IPsec; 80*byte* data has to be authenticated in IPsec and the packet has to be transmitted, respectively. In case that 54*byte* of data is required to be stored and transmitted using combined storage and communication method, 64*byte* data has to be encrypted and written to flash memory for storage; 64*byte* data has to be read from the flash memory; 80*byte* of data has to be authenticated and the packet has to be transmitted.

The system is able to skip two cryptographic operations when using combined secure storage and communication. Therefore, the energy consumption decreases by 32.1%, even when additional 10*byte* have to be written to and read from flash memory. We do not detail energy savings for all other experiment combinations as discussed in the previous section. However, in all cases our proposed method leads to energy savings. In the worst-case, energy consumption decreases by 18.7% (in *Experiment D* with 6*byte* data size).

3.3 Chapter Summary

In the first part of this chapter we present Codo, a novel framework for confidential data storage on sensor nodes. We have described and evaluated a Codo implementation for Contiki. As described, Codo addresses a number of shortcomings in existing secure storage solutions. Codo matches hardware capabilities with security requirements; in-network processing capabilities are preserved while providing confidentiality; already encrypted data in the file system can be used directly for communication.

Codo uses AES with a 128 bit key length. Given this key length, the best known AES attack is four times better than an exhaustive search [BKR11], which requires a brute force attack with 2^{126} keys. This is considered infeasible within a reasonable time frame and, thus, we consider Codo to be secure. Security requires a processing overhead which is considerable. This processing overhead is proportional to the increase in energy consumption of a node as the CPU is active for longer. However, the CPU is generally the smallest energy consumer (radio and sensors consume far more energy) and the

overall increase in energy consumption is reasonable given the added security. The exact reduction of node lifetime depends on the particular CPU type and sensor platform used.

In the second part of this chapter, we have shown that combined secure storage and communication can reduce security related real-time processing on nodes dramatically (up to 71% reduction). As shown, this can be achieved while decreasing as well a node's power consumption (up to 32.1%). Furthermore, we have shown that this is possible within the context of the IP protocol family which we believe will be used in the future IoT. The described solution requires additional storage space on nodes. However, we believe that currently available flash memory sizes can absorb these overheads.

Data on nodes must be secured when stored *and* transported in order to implement a comprehensive security solution. As resource-constrained embedded systems are limited in resources it is necessary to find efficient solutions. As shown in this chapter, the proposed framework combining security aspects of storage and communication can help to achieve this goal.

Chapter 4

Node Identification Based on Clock Skew

All clocks on Wireless Sensor Network (WSN) platforms experience a natural drift. This drift is unique to a node as it depends on the clock hardware[1]. For example, the drift of a node's real-time clock is defined by unique properties of the quartz crystal used. For most WSN applications, clock drift is a nuisance and mechanisms such as time synchronisation protocols are put into place to combat it. However, in this work, we take advantage of a node's unique clock drift pattern and use it to uniquely identify nodes.

Besides other security requirements, it must be possible to authenticate nodes and sensor data provided by them. For example, a sink node must be able to verify that data is provided by genuine nodes and not by an adversary. Classical cryptographic methods can be used to implement authentication. In this case, shared keys are used to identify nodes. As keys may become compromised (i.e. someone obtaines a copy) which would allow an adversary to impersonate a node, methods have been developed which bind authentication to a node's hardware. For example, a crypto chip such as the Atmel ATSHA204 [Atm12] can be included in a node's design which holds cryptographic material for authentication. In this case an attacker must obtain the crypto chip in order to impersonate a node. However, crypto chips are expensive and require an additional

[1]This chapter is based on the paper titled "Node Identification Using Clock Skew".

component to be integrated in the node design. Thus, we use an already present hardware characteristic to derive a unique node identification; we use a node's unique clock skew characteristic for identification.

In addition to the outlined security application, clock-skew-based identification is useful for other tasks. For example, when commissioning nodes, unique identifiers such as node IDs and communication addresses must be determined. Clock skew can also be used in this broader context to generate unique identifiers.

Clock skew has been previously used as means of node identification. For example, Kohno et al. [KBC05] have shown that clock skew is unique and can be used to identify classical PCs in the Internet. Uddin et al. [UC10] have shown that this method can be used in principle in the context of WSNs. Existing work determines clock skew by comparing clocks on separate nodes (or nodes and a sink). For this process, it is necessary to distribute timestamps over the underlying communication network and a *constant* network delay is required. In a WSN context, this is an impractical requirement as duty-cycled communication induces large network delay variances.

In this part of the thesis we move away from this limitation of existing work and we measure clock skew locally on nodes. We discuss how this can be achieved in general, describe an implementation of this method and provide a detailed analysis describing clock sampling requirements and achievable quality. More specifically, the contributions of the work are:

- *Local Skew Determination:* We describe how the clock skew of a node's crystal-based real-time clock can be measured locally using the high-precision clock available on modern transceivers to create unique node identifiers. An implementation of this method for the Zolertia Z1 platform using Contiki is described.

- *Analysis of Sampling Requirements:* The achievable quality of the local clock skew determination is compared with the quality of state of the art distributed methods. The dependency of clock sampling effort and skew calculation quality is analysed in detail. In this context, it is also shown how clock sampling can be aligned with general transceiver operation in order to avoid a transceiver duty cycle increase.

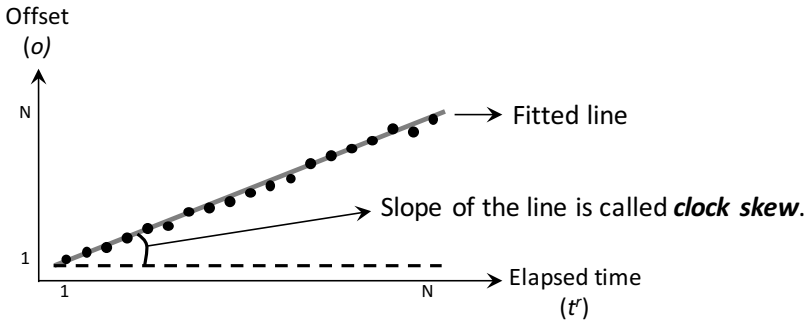


Figure 4.1: Clock skew estimation.

4.1 Clock Skew

Clock skew could be determined by analysing the drift of a clock C_m with the help of a stable reference clock C_r. However, in a practical setting, a stable reference clock is generally not available and it is only possible to monitor one drifting clock with another drifting clock. Hence, a measured clock skew value for a node reflects drift of the measured clock and the used reference clock. Nevertheless, the determined clock skew value is unique and dependent on the hardware characteristics of the clocks used.

4.1.1 Definition of Clock Skew

The measured clock C_m runs at frequency f_s and the reference clock C_r runs at frequency f_r. To determine clock skew, timestamps of the measured clock and the reference clock are taken periodically. T_1^m and T_1^r are the first timestamps of both clocks taken at the first sample point, T_i^m and T_i^r are timestamps taken at the i^{th} sample point. The elapsed time of the measured clock C_m at the i^{th} sample point is $t_i^m = (T_i^m - T_1^m)/f_m$; the elapsed time at the reference clock C_r is $t_i^r = (T_i^r - T_1^r)/f_r$. The *offset* – the difference between measured and reference clock – at the i^{th} sample point is $o_i = t_i^m - t_i^r$. If we sample N pairs of (t_i^r, o_i) for $i \in \{1, ..., N\}$ and plot these pairs (the so called offset-set), we obtain an approximately linear graph. It is possible to fit a linear function of the form

$$\delta \cdot t_N^r + \varphi \tag{4.1}$$

to these obtained measurement points. The slope δ of the fitted linear function is called the *clock skew*. Figure 4.1 demonstrates the clock skew estimation.

4.1.2 Clock Skew Determination

There are a number of methods available to fit a linear function. Depending on the exact method used the definition of *clock skew* is also slightly altered. In the literature mainly two methods are used in the context of clock skew calculation which we detail next and use in the remainder of the work.

4.1.2.1 Linear Programming

Moon et al. [MST99] have shown that clock skew can be accurately estimated from a set of samples using a Linear Programming (LP) method. LP finds a line $\delta \cdot t_i^r + \varphi$ that upper bounds the offset-set. The problem constraint of LP is given as

$$\delta \cdot t_i^r + \varphi \geq o_i \tag{4.2}$$

and the following function is minimized (object function):

$$\frac{1}{N} \cdot \sum_{i=1}^{N} (\delta \cdot t_i^r + \varphi - o_i) \tag{4.3}$$

LP delivers accurate results but has drawbacks when considering a WSN context. Samples must be collected and stored before the calculation can begin. Furthermore, a relatively complex solver for the LP must be available. If considering to execute the calculation on a resource constrained node storage and calculation requirements may be too excessive.

4.1.2.2 Linear Regression

Maróti et al. [MKSL04] estimate clock skew using a form of Linear Regression (LR) in their Flooding Time Synchronization Protocol (FTSP). The algorithm uses the average elapsed time at the reference clock $\overline{t^r}$ and the average offset $\overline{o}$ up to the i^{th} sample point.

Then the skew is estimated by calculating:

$$\delta = (o_i - \overline{o})/(t_i^r - \overline{t^r}) \tag{4.4}$$

For the FTSP 8 sample points are used to estimate the clock skew. We use this algorithm with a varying number of sample points in order to have control over the achievable quality of the skew estimation. Compared to LP the implementation is much simpler and therefore more suitable for usage on resource constrained nodes.

4.1.3 Clock Skew Quality

In this work we aim to use clock skew to uniquely identify nodes. Hence it is important that clock skew measurements on two nodes can be attributed to the individual nodes. The collision probability (the likelihood that two nodes are seen as the same even though they are different) should be as small as possible.

Clock skew measurement, for example using the previously described LP or LR method, is subject to variation. Thus, as two nodes may have a skew values close to each other, variance of the measured skew may make it hard to clearly attribute measurements to individual nodes.

We use a t-test to compare the means of the measured clock skews of two nodes. The t-test returns a test decision for the null hypothesis that the nodes are the same. The alternative hypothesis is that the two nodes are not the same. The probability value (p-value) returned by the t-test is the probability of rejecting the null hypothesis (two nodes are the same) when the null hypothesis is true. Therefore, a small p-value indicates that the means of clock skew measurements of two investigated nodes are unlikely to be the same.

For the experimental evaluations described in later sections we use p-values to describe quality of determined clock skew measurements.

4.2 Remote Clock Skew Determination

In existing work (for example, [UC10] and [HTW+08]) measured clock C_m and reference clock C_r are located on different nodes in the network. For example, the reference clock is the real-time clock of the sink node and the measured clock is the real-time clock of the node to be identified. This remote clock skew determination is carried out as a node generally provides only one accurate clock in its default configuration which is suitable for skew calculations.

4.2.1 The Impact of Network Jitter

As shown in Section 4.1, timestamps T_i^m and T_i^r are the i^{th} sample taken at the same point in time. When using remote clock skew determination T_i^m is taken first and the timestamp is transmitted via the network to the reference node which then takes the corresponding timestamp T_i^r. Thus, T_i^r is taken Δ_i after the sample T_i^s. Δ_i is the network delay associated with transmitting the i^{th} timestamp of the measured clock. The clock skew calculation as presented in Section 4.1 is still valid if the network delay is constant ($\Delta_i = \Delta \, \forall i$). Variations in Δ_i (network jitter) reduce the quality with which the clock skew can be determined. In a practical WSN network jitter is high due to the nature of duty-cycled communication. For example, when using a protocol such as ContikiMAC [Dun11] the forwarding delay is dependent on when a sender transmit request occurs relatively to the point in time when a receiver node enters its periodic listen phase. Using standard ContikiMAC configuration settings, forwarding delays vary between $0ms$ and $125ms$ (ContikiMAC uses channel check rate of $8Hz$). This is too high to measure the characteristics of clock skew.

Existing work uses remote clock skew determination in specific setups where nodes are only one hop distance away and no duty cycling MAC protocols or significant network traffic is present. In such a specific case network jitter is low and remote skew determination is possible. However, in any practical setting this method has its limitations.

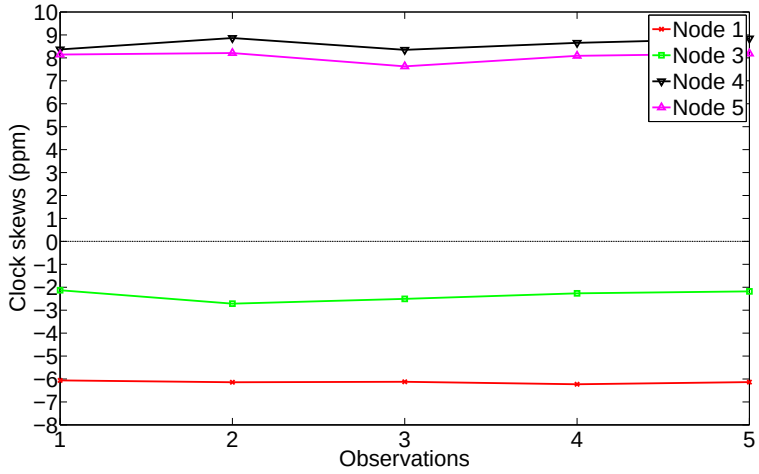


Figure 4.2: The measured clock skew using five Zolertia Z1 nodes with remote clock skew determination. 5 observations are carried out using 2500 timestamp samples.

4.2.2 Experimental Evaluation

We use five Z1 nodes from Zolertia [Zol10] to carry out a baseline experiment using remote clock skew determination. We use the results obtained in this experiment as comparison for the local clock skew determination method we introduce later.

The Z1's external $32,768Hz$ watch crystal is used for clock skew profiling. We label the nodes from Node 1 to 5 and select Node 2 as the sink node. Nodes are one hop away from a sink node and its external crystal clock is used as reference clock C_r. The nodes run the Contiki operating system and use a NullMAC (no duty cycling MAC is present). No other network traffic than the transmission of timestamps from the 4 measured nodes is present in the network.

Timestamps T_i^m are transmitted every $4s$ from the nodes to the sink. When the sink node receives a timestamp T_i^m it records the corresponding timestamp T_i^r. The total transmitted packet count is 2500 for every node. We repeat this operation 5 times (5 observations) and estimate the clock skew for each node. The clock skew is estimated using the previous described LP method. This calculation is carried out offline after collection of all timestamp samples. The result is shown in Figure 4.2; clock skew is shown in "part per million" (ppm).

Figure 4.2 shows that nodes can be clearly identified by measuring clock skew. The

	Node 1	Node 3	Node 4	Node 5
Node 1	-	$4.25E-06$	$3.19E-08$	$2.36E-08$
Node 3	$4.25E-06$	-	$3.91E-07$	$1.69E-07$
Node 4	$3.19E-08$	$3.91E-07$	-	$3.23E-03$
Node 5	$2.36E-08$	$1.69E-07$	$3.23E-03$	-

Table 4.1: Obtained p-values describing how clearly nodes can be distinguished from each other node. The smaller the value the more clearly nodes are distinguishable.

Node 1	Node 3	Node 4	Node 5
$1.14E-01$	$9.43E-01$	$4.49E-01$	$3.54E-01$

Table 4.2: Obtained p-values when comparing a node with itself. Values are larger (2 magnitudes) then the ones shown in Table 4.1.

clock skew is stable enough over several observations. For some nodes it is easier to distinguish them (for example, Node 1 and Node 4 are clearly separate nodes), others have skew values close together (for example, Node 4 and Node 5). However, even though some nodes are close together measured skew values can be clearly attributed to nodes. How clearly nodes can be identified as separate (the quality of the skew values) can be expressed using the previously outlined t-test. Table 4.1 shows the results of this analysis. To provide some means of judging p-values we provide Table 4.2. Here, the means of the first two observations of a node are compared with the mean of the third and forth observation of a node; effectively p-values are generated where a node is compared with itself and values are generated that are not distinguishable. As it can be seen, worst case p-values in Table 4.1 are two orders of magnitude lower then in Table 4.2. This indicates that the investigated set of nodes in this experiment is clearly uniquely identifiable via measured clock skews.

4.3 Local Clock Skew Determination

To overcome the previously outlined limitations of remote clock skew determination (i.e. the need of a constant network delay), it would be beneficial to use two local clocks on a node for skew determination.

4.3.1 Local Clock Sources

Most sensor node platforms provide two clock sources, the crystal-based real-time clock and a processor internal Digitally Controlled Oscillator (DCO). However, these two available local clocks cannot be used for skew determination as the DCO clock has a much lower precision then the real-time clock. Thus, no stable clock skew values can be determined using this setting.

However, most node platforms have a radio transceiver chip which has internally a high-precision clock which is necessary for timing of data transmissions. In most cases it is possible to access this resource and use it within the platform for other purposes than transmission and reception of data.

The Zolertia Z1 platform we use for our work provides an $8MHz$ clock within the CC2420 radio transceiver. We use the same approach that Pettinato et al. [PWEV12] used to make use of the clock source of the radio. The Clear Channel Assessment (CCA) pin of the radio is alternatively configured to output the internal radio clock signal [Tex13]. We connect the radio CCA pin with the external timer sources pin (TBCLK) of the MSP430 processor. This allows us to use the radio clock as alternative clock source. As this clock is of better quality than the crystal-based Real-time Clock (RTC) it is now possible to use these two clocks to determine locally on the node a stable clock skew value.

4.3.2 Experimental Evaluation

In this experiment we use the same 5 Z1 nodes as used for the previously described baseline experiment. The Z1's external $32,768Hz$ watch crystal (the measured clock C_m) is used again for clock skew profiling. The CC2420 radio clock source is used as reference clock C_r. Obviously, all nodes have their individual transceiver clock and, thus, for each node skew profile an individual reference clock is used. Timestamps T_i^m and T_i^r are collected with an interval of $1s$. The results determined using the LP method are shown in Figure 4.3.

As can be seen, nodes are clearly identifiable in terms of observed clock skew. The

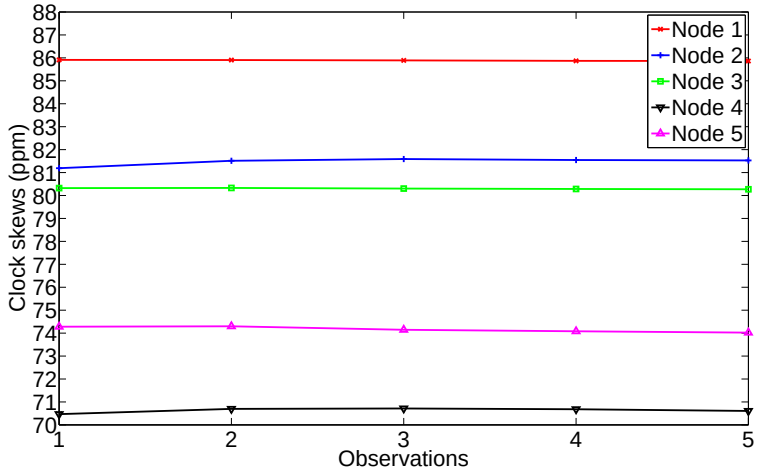


Figure 4.3: The measured clock skew using five Zolertia Z1 nodes with local clock skew determination. 5 observations are carried out using 600 timestamp samples.

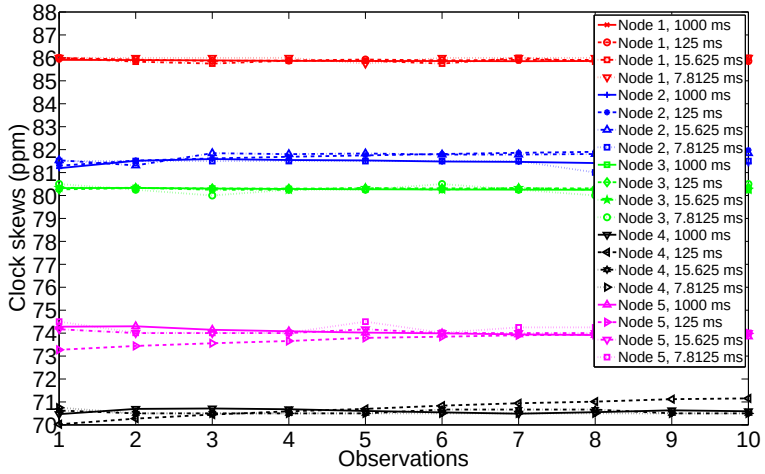
skew values are different to the values obtained remotely as shown in Figure 4.2. This has to be expected as different reference clocks are used.

The calculated p-values for the five nodes are shown in Table 4.3. As can be seen here, p-values are several magnitudes lower compared to the remote skew determination. This means that individual nodes can be distinguished much more clearly. It has to be noted that this significant improvement is achieved even though fewer sample points (600 compared to 2500) are taken and a shorter sample durations (1s compared to 4s) are used.

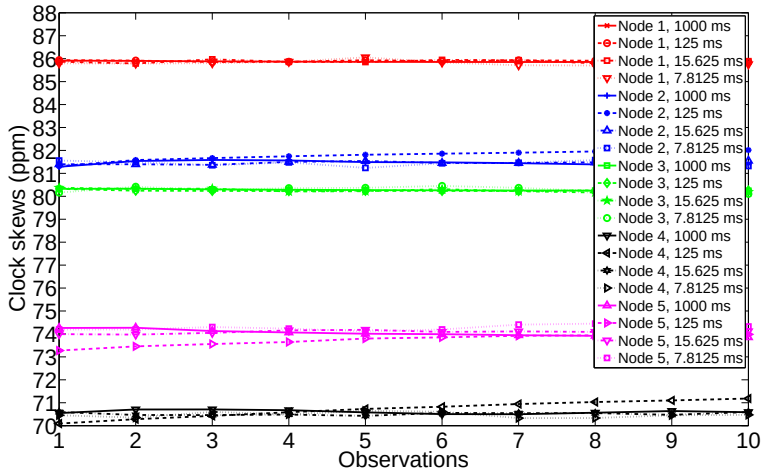
We conclude that the local clock skew determination is much better than remote clock skew determination as nodes can be more clearly identified. The collision probability (the likelihood that two nodes are seen as the same even though they are different) is greatly reduced. This is an important factor for the design of security mechanisms.

4.3.3 Processing Optimisation

So far, we have used the LP method to determine clock skew. This method is useful and provides sufficiently accurate results (as previously shown) for local and remote clock skew determination. However, LP is too complex to execute directly on resource constrained nodes. We therefore use the LR method which is computationally more



(a) Clock skews using LP.



(b) Clock skews using LR.

Figure 4.4: The measured clock skew of five Zolertia Z1 nodes with remote local clock skew determination. 10 independent observations with 600 timestamp samples are used.

feasible. LR is expected to deliver results of lesser quality compared to the LP method. It is therefore necessary to analyse if the quality of the results is still sufficient to distinguish individual node clock skews.

Figure 4.4 shows a comparison of local skew determination using the LP and LR skew calculation method (The figure also contains lines showing the effect of reducing the sample period from $1s$ down to $7.8125ms$; we discuss the effect of reduced sample period in the next section). The corresponding p-values are given in Table 4.3. Interestingly,

	Method	Node 1	Node 2	Node 3	Node 4	Node 5
Node1	LP	-	$1.30E-15$	$1.6E-24$	$8.13E-22$	$6.58E-19$
	LR	-	$1.92E-16$	$6.12E-25$	$1.74E-22$	$3.12E-19$
Node2	LP	$1.30E-15$	-	$1.63E-10$	$8.32E-21$	$1.21E-15$
	LR	$1.92E-16$	-	$1.59E-11$	$8.79E-21$	$1.46E-16$
Node3	LP	$1.36E-24$	$1.63E-10$	-	$4.13E-20$	$8.29E-17$
	LR	$6.12E-25$	$1.59E-11$	-	$7.01E-21$	$3.98E-17$
Node4	LP	$8.13E-22$	$8.32E-21$	$4.13E-20$	-	$2.03E-13$
	LR	$1.74E-22$	$8.79E-21$	$7.01E-21$	-	$4.36E-14$
Node5	LP	$6.58E-19$	$1.21E-15$	$8.29E-17$	$2.03E-13$	-
	LR	$3.12E-19$	$1.46E-16$	$3.98E-17$	$4.36E-14$	-

Table 4.3: p-values using LP and LR with 10 observations, 600 samples, $1s$ sample period.

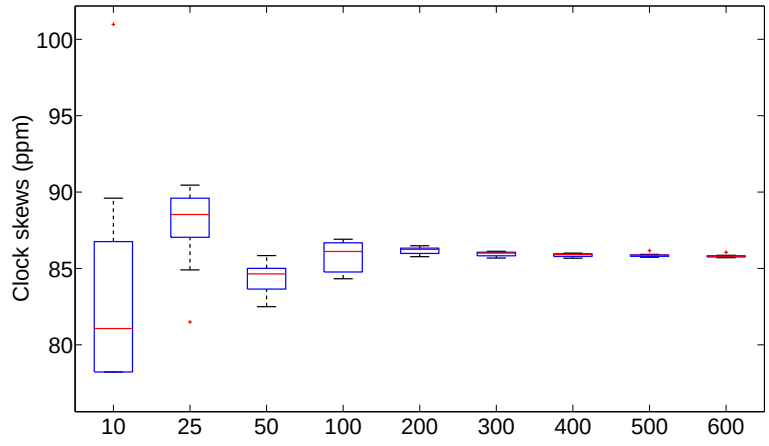
the LR method fares sightly better in terms of producing clearly distinguishable clock skew values. This is contrary to what we would have expected. Clearly, LR therefore represents a feasible option for clock skew calculation.

We have implemented the LR method of calculation for the Contiki operating system (LP was executed in Matlab after data had been collected). The run-time complexity of the LR algorithm is $O(n)$. For the calculation of skew using 200 sample points an execution time of $2.01ms$ is required which corresponds to an energy consumption of $0.027mJ$.

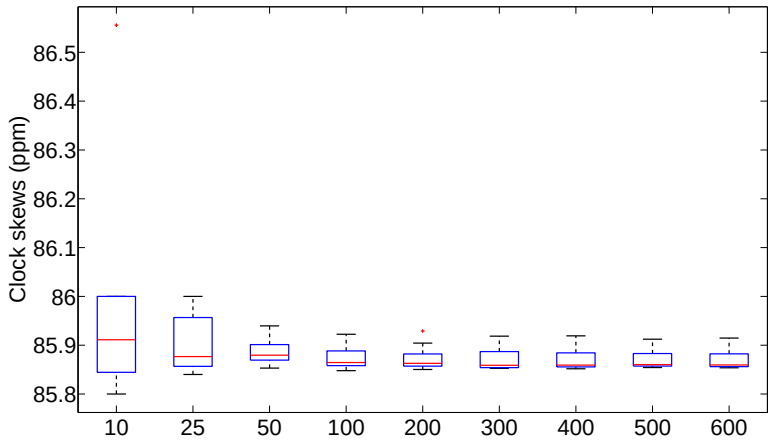
4.3.4 Sampling Optimisation

So far we have demonstrated the feasibility of local clock skew determination using relatively long sampling periods of $1s$ and and a relatively large sample size of 600 samples. It is beneficial to reduce as much as possible sample period and sample size. Reducing the sample period and size allows for more energy-efficient operations and enables us to obtain skew values faster. When reducing the sample period, less time to observe skew is available and resolution of the used clocks can be a limiting factor. When reducing the sample size less points are available to reduce variance of the obtained result.

For local clock skew determination as presented in this work the transceiver clock is required. This clock is only available if the transceiver chip is active. A duty-cycled MAC protocol will aim to put the transceiver into an energy-efficient sleep state for as



(a) Node 1 using a 7.8125ms sample period.



(b) Node 1 using a 1s sample period.

Figure 4.5: Node 1 skew for different sample sizes and sample period of 7.8125ms and 1s.

long as possible and only wake the transceiver for short durations for transmissions and receptions. If short sample periods are feasible it is possible to align communication and clock sampling and no additional transceiver chip wake times must be introduced. For example, many slotted MAC protocols have transceivers periodically on for durations of 10ms to facilitate reception or transmission.

The required sample size should be as small as possible as this would allow us to obtain a skew measurement fast. If we assume that sample periods are aligned with natural transceiver activity we still have to wait until the transceiver was used sufficiently

	Node 1	Node 2	Node 3	Node 4	Node 5
Node 1	-	$6.66E-10$	$6.14E-12$	$3.14E-16$	$3.37E-15$
Node 2	$6.66E-10$	-	$1.21E-04$	$3.25E-12$	$2.08E-12$
Node 3	$6.14E-12$	$1.21E-04$	-	$3.79E-12$	$1.92E-11$
Node 4	$3.14E-16$	$3.25E-12$	$3.79E-12$	-	$1.69E-09$
Node 5	$3.37E-15$	$2.08E-12$	$1.92E-11$	$1.69E-09$	-

Table 4.4: p-values using LR with 10 observations, 200 samples, $7.8125ms$ sample period.

often before a skew measurement can be obtained.

We record 600 samples with four different sampling intervals: $1s$, $125ms$, $15.625ms$ and $7.8125ms$. These results are shown for 10 observations in Figure 4.4, using LP and LR for analysis. Figure 4.5a and Figure 4.5b show changes in skew variance of 10 observations when modifying sample size and sample period.

It is clearly visible that skew variations experienced from one measurement to the next are reduced when increasing sample period and/or sample size. The question is what a feasible combination of sample size and sample period is. The answer will depend on the application situation. However, generally it can be assumed that the shortest feasible sampling period should be used as this helps in aligning sampling with general transceiver activity. Then, the number of samples should be reduced up to a point a sufficient quality (expressed as p-values) of skew calculation is ensured. For example, if we assume that a sample period of $7.8125ms$ and a sample size of 200 is chosen, we obtain p-values as shown in Table 4.4. These p-values are better than the p-values obtained via remote skew determination. Thus, with these settings local skew detection can replace remote skew detection without a loss in skew quality.

4.4 Chapter Summary

A node's unique clock skew can be used for node identification purposes. It is useful to use this approach because node identification is bound to the hardware and no additional components have to be incorporated in the node design. We have demonstrated that clock skew can be determined reliably locally on nodes. Existing methods rely on jitter free network communication which is unachievable in most practical WSN deployments. Thus,

the presented work takes an important step towards practical clock skew identification.

We have shown that clock skew of a node's RTC can be determined locally using the transceiver clock present in most WSN systems. We have shown that locally determined skew values for nodes can be as unique and distinguishable as skew values determined in a distributed fashion. A sample period of $7.8125ms$ and a sample size of 200 are sufficient to determine clock skew locally with the same quality as a remote technique with a sample period of $4s$ and a sample size of 2500. Also, the possible short sample period of $7.8125ms$ allows us to take clock skew measurements during times the transciever is active for communication. Additional transceiver active periods do not have to be scheduled to achieve local clock skew determination.

Chapter 5

Tamper Detection and Node Identification Based on Channel State Information

A large number of Wireless Sensor Network (WSN) systems are using Wi-Fi as a means of communication [BKR15][1]. Wi-Fi devices are now commonplace and are often deployed in application scenarios with strict security requirements. For example, wireless devices are often part of physical intrusion detection systems used to protect critical infrastructure. Wireless surveillance cameras might be used within a physical intrusion detection system of an airport. An attacker may aim to change the camera's area of observation. Transmitted image data would still be cryptographically authenticated and tamper detection would require the inspection of the visual data. An attacker may also obtain key material and replace a node in the deployment to inject false observation data. For these reasons, it is desirable to provide an additional layer of defence that is able to indicate node tampering. In this work, we use wireless channel characteristics to achieve this.

Communication environment changes can be observed via changes in channel characteristics and this effect can be exploited for security purposes. Previous work exploited

[1]This chapter is based on the papers titled "Short Paper: Gathering Tamper Evidence in Wi-Fi Networks Based on Channel State Information" and "Using Channel State Information for Tamper Detection in the Internet of Things".

channel characteristics to detect location changes [ZFPK08] or message injection [JZL+13]. In this work, we use those characteristics to detect tamper events for each node. In particular, and in contrast to existing work, we consider transmissions over a varying number of spatial dimensions in multi-antenna configurations of 802.11n Wi-Fi systems (see Section 2.1.6 for details), and propose a tamper detection algorithm that works in practical deployments.

In 802.11n, the frame preamble on the physical layer is used to compute Channel State Information (CSI) for each incoming packet. We use this measurement for tamper detection. Moving or replacing a transmitter changes the observed channel characteristics, which results in an increasing tamper-evidence value. For example, a high tamper-evidence value may be used to dispatch security personal to check integrity of a device.

Unfortunately, not only tamper events lead to CSI fluctuations; modifications of the communication environment have an impact too. In particular, movement of people in the communication environment has a noticeable influence on the CSI. As we will show in Section 5.2.1, tamper detection based on CSI analysis is generally possible but it is hard to distinguish tamper events from natural changes in the communication environment. Thus, it is difficult to construct a practical security system based on CSI analysis as a high number of false positives and a low number of true positives are detected. Figure 5.1 illustrates this challenge. The changing CSI amplitude value over time for just one subcarrier is shown. As can be seen, tamper events and environmental changes (in this case a person walking between sender and receiver) manifest in very similar ways.

To address this problem we propose to analyse CSI values from a single transmission simultaneously at multiple receivers to improve distinction of tamper and movement events. A moving person is expected to have an impact on some but not all communication links while a tamper event is noticeable at all receivers. We describe the necessary algorithms for the proposed CSI tamper detection method using multiple receivers. We analyse the tamper detection capability in practical deployments with varying intensity of people movement.

In this part of the thesis, we describe algorithms for tracking CSI values of Wi-Fi

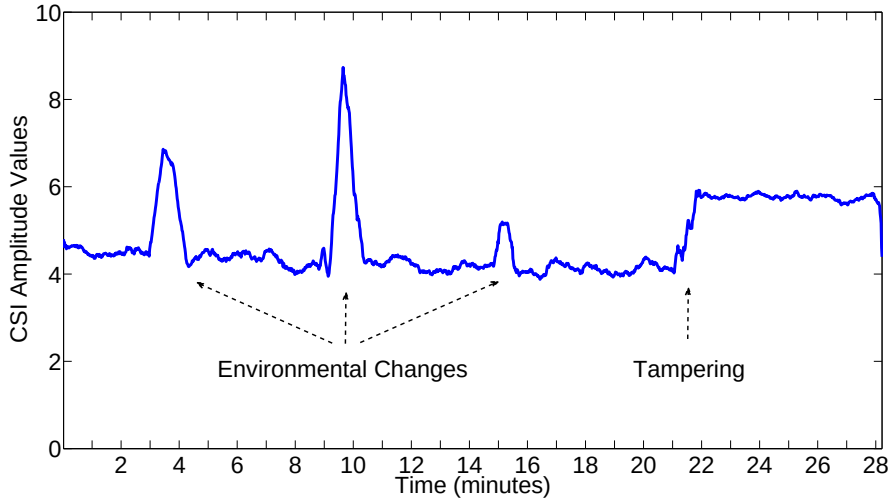


Figure 5.1: Effects of environmental changes and tampering on CSI amplitude values of the 9^{th} subcarrier of the 2^{nd} antenna. Environmental changes and tamper events have a similar effect on CSI amplitude values.

transmissions and show how to compute the tamper-evidence value. In particular, we account for 802.11n's varying number of used spatial streams, and analyse detection capability in practical deployments. To extract CSI information from off-the-shelf hardware, we use the *Linux 802.11n CSI Tool* [HHSW11]. No additional transmissions or protocol modifications are necessary for tamper detection. The contributions of this chapter are:

- *Tamper Detection:* We describe a method for device-specific tamper detection based on per packet 802.11n CSI. The method handles varying numbers of spatial streams; a feature used in 802.11n which is not yet considered by previous work. We also describe a tamper detection algorithm that can be used in practical deployments using multiple receivers.

- *Detection Analysis:* We demonstrate that device movements of just 1 cm and hardware replacements are clearly detectable. We analyse the tamper detection capability of the proposed method using realistic deployment environments. We describe system capability in these environments in terms of achievable False Positive Rate (FPR) and True Positive Rate (TPR).

- *Tamper Detection Implementation:* We describe how the proposed algorithms can

be put to action in practical deployments. We discuss their application in the future WSNs.

5.1 Tamper Detection

We use CSI of 802.11n packets in this work. As explained in Section 2.1.6, CSI is the estimation of the channel conditions and represented as $\mathbf{M}^{sc}_{R\times S}$. It gives the amplitude changes and phase shifts for each subcarrier *sc*.

We analyse the characteristics of a wireless channel at the receiver using CSI to detect tampering on the transmitter. CSI values changes with the transmitter location or hardware changes. However, CSI is also affected by tamper-unrelated events, like changes in the communication environment due to moving people. These changes have to be addressed in order to reduce false alarms. In the case that transmitter uses spatial multiplexing, each transmission might be sent with different number of spatial streams, as it is explained in Section 2.1.6.1. Different spatial streams have different CSI values, and this aspect has to be addressed during the design of a tamper detection mechanism.

We first show the feasibility of tamper detection based on CSI using a single receiver. In this setting, number of spatial streams S may vary between one and the number of transmission antennas T. Then we provide a more reliable tamper detection mechanism for practical deployments using multiple receivers. The transmitter must send broadcast packets in this setting, because the receivers use the same link for tamper detection. If the transmitter sends the packets in broadcast mode (i.e., receiver agnostic) it only uses one spatial stream. Nodes transmit periodic broadcast beacons which can be picked up by multiple receivers for CSI-based tamper detection.

Using single receiver or multiple receivers for tamper detection requires slightly different algorithms to be able to handle different number of spatial stream settings. We will show these algorithms in the following subsections.

5.1.1 Single Receiver Tamper Detection

In the single receiver tamper detection algorithm, receivers collect and store τ CSI measurements $\mathbf{M}^{sc}_{R\times S,i\in 1...\tau}$. These CSI measurements are collected while the transmitter is in a tamper free state and will be used for comparison with newly received CSI measurements $\mathbf{M}^{sc}_{R\times S,i>\tau}$. A distance metric is used as existing works show that such algorithms work well in this context (see related work [PK07, ZFPK08]). If the distance is above a certain threshold, the algorithm decides there is a tamper event and triggers an alarm.

A transmitter with T antennas can choose from one to T spatial streams. We create T different models $Mod_{m=1...T}$, because each model requires its own training for the corresponding spatial stream setting. Additionally, each model has one to m spatial streams. Therefore, we need to generate models $Mod_{m=1...T,s=1...m}$ for each spatial stream s.

The algorithm uses only the amplitude information of a CSI measurement and omits the phase information. The amplitude information is normalized by taking the Euclidean norm of all values in dimensions sc and r. The Euclidean distance $D^i_{m,s}$ for a model $Mod_{m,s}$ is obtained by calculating the Euclidean distance between the stored τ CSI measurements and a new CSI measurement $\mathbf{M}^{sc}_{R\times S,i}$:

$$D^i_{m,s} = \frac{1}{\tau}\sum_{j=1}^{\tau}\sqrt{\sum_{r=1}^{R}\sum_{sc=1}^{SC}\left(\frac{|M^{sc}_{r,s,i}|}{||M^{sc}_{r,s,i}||_2^{sc,r}} - \frac{|M^{sc}_{r,s,j}|}{||M^{sc}_{r,s,j}||_2^{sc,r}}\right)^2}$$

$D^i_{m,s}$ can have outliers as shown in Section 5.2. We are applying sample-wise and time-wise moving average filter to $D^i_{m,s}$ in order to remove the outliers. The sample-wise moving average filter takes the mean of k samples:

$$D^i_{m,s,\text{sampMA}} = \sum_{j=i-k+1}^{i}\frac{D^j_{m,s}}{k} \text{ for } i \geq k$$

We take the time points t_i when a packet arrives and the CSI of the packet is measured,

and average these time points over a time window t_w for time-wise moving average filter:

$$D^i_{m,s,\text{timeMA}} = \frac{1}{\text{count}(t_j)} \sum_{\substack{j \\ t_j \in (t_i - t_w, t_i)}} D^j_{m,s} \text{ for } t_i \geq t_w$$

Finally, we take the time-wise moving average of all models $Mod_{m=1...T,s=1...m}$ where $D^i_{m,s,\text{timeMA}} \neq 0$ and average them to get the combined model Mod_c:

$$D^i_c = \left[\underset{\forall s,m}{\text{count}} \left(D^i_{m,s,\text{timeMA}} \neq 0 \right) \right]^{-1} \sum_{m=1}^{T} \sum_{s=1}^{m} D^i_{m,s,\text{timeMA}}$$

The resulting value D^i_c is compared against a threshold to make a decision whether there is a tampering or not. Here, we use the maximum and mean pair-wise distances between all training CSIs $\mathbf{M}^{sc}_{R \times S, i \in 1...\tau}$ for the threshold:

$$D^\tau_{i,j} = \sqrt{\sum_{r=1}^{R} \sum_{sc=1}^{SC} \left(\frac{|M^{sc}_{r,s,i}|}{||M^{sc}_{r,s,i}||_2^{sc,r}} - \frac{|M^{sc}_{r,s,j}|}{||M^{sc}_{r,s,j}||_2^{sc,r}} \right)^2}$$

$$\text{mean}(D^\tau) = \underset{\forall i,j}{\text{mean}} \left(D^\tau_{i,j} \right) \qquad \max(D^\tau) = \underset{\forall i,j}{\max} \left(D^\tau_{i,j} \right)$$

A further discussion about using those metrics, as well as applying the tamper estimation to channel measurements is presented in the Evaluation section.

5.1.2 Multi-Receiver Tamper Detection

Here we provide a more reliable tamper detection mechanism for practical deployments using multiple receivers. As the transmitters send their packets in broadcast mode in the multi-receiver tamper detection, only one spatial stream is used ($S = 1$). Therefore, we simplify the representation of the CSI matrices and use $\mathbf{M}^{sc}_R$ when we are explaining the multi-receiver tamper detection algorithm.

In the multi-receiver tamper detection algorithm, receivers collect and store τ CSI measurements $\mathbf{M}^{sc}_{R,i \in 1...\tau}$, as when using a single receiver. These CSI measurements are

collected while the transmitter is in a tamper free state and will be used for comparison with newly received CSI measurements $\mathbf{M}^{sc}_{R,i>\tau}$. A distance metric for comparison is used again. If the distance is above a certain threshold, the algorithm decides there is a tamper event and triggers an alarm.

We consider three distance algorithms this time: (i) *Euclidean distance*, (ii) *Mahalanobis distance* and (iii) *Earth Mover's distance*. The Euclidean distance gives the distance between two points; in our case the distance of two CSI vectors. Mahalonobis distance gives the distance between a point and a distribution, in our case the distance of a new CSI vector and a distribution of τ CSI vectors. And lastly, Earth Mover's distance gives the distance between two distributions, in our case the distance of the distribution of two CSI vectors.

The algorithm uses only the amplitude information of a CSI measurement and omits the phase information. The amplitude information is normalized by taking the Euclidean norm of all values in dimensions sc and r. Euclidean distance D_i is obtained by calculating the Euclidean distance between the stored τ CSI measurements and a new CSI measurement $\mathbf{M}^{sc}_{R,i}$.

$$D_i = \frac{1}{\tau}\sum_{j=1}^{\tau}\sqrt{\sum_{r=1}^{R}\sum_{sc=1}^{SC}\left(\frac{|M^{sc}_{r,i}|}{||M^{sc}_{r,i}||_2^{sc,r}} - \frac{|M^{sc}_{r,j}|}{||M^{sc}_{r,j}||_2^{sc,r}}\right)^2}$$

Similarly, we omit the phase information in $\mathbf{M}^{sc}_{R,i}$ and use normalized amplitude information when computing Mahalanobis and Earth Mover's distance. We omit the details of Mahalanobis and Earth Mover's distance algorithms here as they are well documented in the literature [Mah36, RTG00]. We show in the evaluation Section 5.2.2.1 that the simplest distance computation method (Euclidean distance) is sufficient and that the more sophisticated methods of Mahalanobis and Earth Mover do not provide significantly better results.

To decide if tampering occurred, we need to set a threshold γ. If D_i is greater than γ the algorithm will detect tampering ($q_i = 1$), and otherwise a tamper free state ($q_i = 0$)

is assumed:

$$q_i = \begin{cases} 0 & \text{if } D_i < \gamma \\ 1 & \text{if } D_i \geq \gamma \end{cases}$$

As we will show in Section 5.2.1, when using one link (one receiver) to evaluate tampering it is not possible to distinguish tamper situations and environmental changes. Both events can push the distance value above the threshold. We will demonstrate this effect in Section 5.2.2.1. To overcome this limitation, we aim to use multiple receivers for CSI analysis. The assumption is that a tamper event will push the distance observed at all receivers above the set threshold while a change in the environment will not result in a sufficiently significant distance change at all receivers. We use an approach where a majority vote is used (i.e. the distance is above the threshold at the majority of receivers to declare tampering). Tampering with the transmitter device clearly must influence all links and not just some.

Formally, the overall tampering decision Q^i with q_i^n being the tampering decision at the individual receivers for a new received frame is:

$$Q^i = \begin{cases} 0 & \text{if } \sum_{n=1}^{N} q_i^n < N \\ 1 & \text{if } \sum_{n=1}^{N} q_i^n = N \end{cases}$$

where N is the number of receivers. Here, the packet i must be received by all the receivers for a decision to be made. This situation can occur in practice as the transmitted beacon used for tamper detection may not be received at all stations.

5.1.2.1 Threshold Selection

The performance of the tamper detection largely depends on the selected threshold γ. If the threshold is selected too high, some tamper events might be missed. If the threshold is too low, the system is too sensitive and a large number of false detections may occur. In this work, we consider 3 methods for threshold selection: (i) *Maximum Distance*, (ii)

Equal Error Rate and (iii) *Zero False Negative*.

A simple and straightforward approach is to use the maximum distance of all captured CSIs during the training phase $\mathbf{M}^{sc}_{R,i\in 1...\tau}$ as the threshold:

$$D^{\tau}_{i,j} = \sqrt{\sum_{r=1}^{R}\sum_{sc=1}^{SC}\left(\frac{|M^{sc}_{r,i}|}{||M^{sc}_{r,i}||_2^{sc,r}} - \frac{|M^{sc}_{r,j}|}{||M^{sc}_{r,j}||_2^{sc,r}}\right)^2}$$

$$\max(D^{\tau}) = \max_{\forall i,j}\left(D^{\tau}_{i,j}\right)$$

This threshold works well when the environment is static, as we will show. However, the algorithm does not provide good performance in terms of false alarms when the environment is dynamic.

We use False Positive Rate (FPR), False Negative Rate (FNR) and True Positive Rate (TPR) metrics to assess the detection mechanism. False Positive (FP) occurs when the algorithm falsely decides there is tampering but actually there is not, False Negative (FN) occurs when the algorithm misses a true tampering, and True Positive (TP) occurs when the algorithm catches a tamper situation. We can create a Receiver Operating Characteristic (ROC) curve by evaluating all possible thresholds. The ROC curve displays the trade-off between FPR and TPR for a given system. A balanced threshold can be found for a system by looking at the Equal Error Rate (EER). EER is the point where FPR equals FNR on the ROC curve.

CSI data collected during a training phase can be used to calculate a ROC. The threshold that gives the EER on the ROC curve, γ_{EER}, can be used as a threshold. This threshold gives a balanced result in terms of FPRs and TPRs. However, as we will show in Section 5.2.2.2, γ_{EER} is often too high. It gives very good FPR results, but at the same time it causes very low TPRs. We thus need to decrease the threshold, and to this end we propose to pick the maximum threshold where the ROC curve gives FNR = 0, $\gamma_{FNR=0}$. We will show in Section 5.2.2.2 that this threshold gives fair TPRs and, unfortunately, provides FPR > 0. We propose to apply time-wise filtering to the overall decision to

address this issue, which we detail next.

5.1.2.2 Time-Wise Filtering

We apply a time-wise filter to the overall decision Q^i to reduce the FPRs. We consider the points t_i—when a packet i was received by all the receivers—and make a decision over a window t_w. The overall time-wise filtered decision Q^{i,t_w} is considered tampered if all individual decisions in the window t_w were tampered:

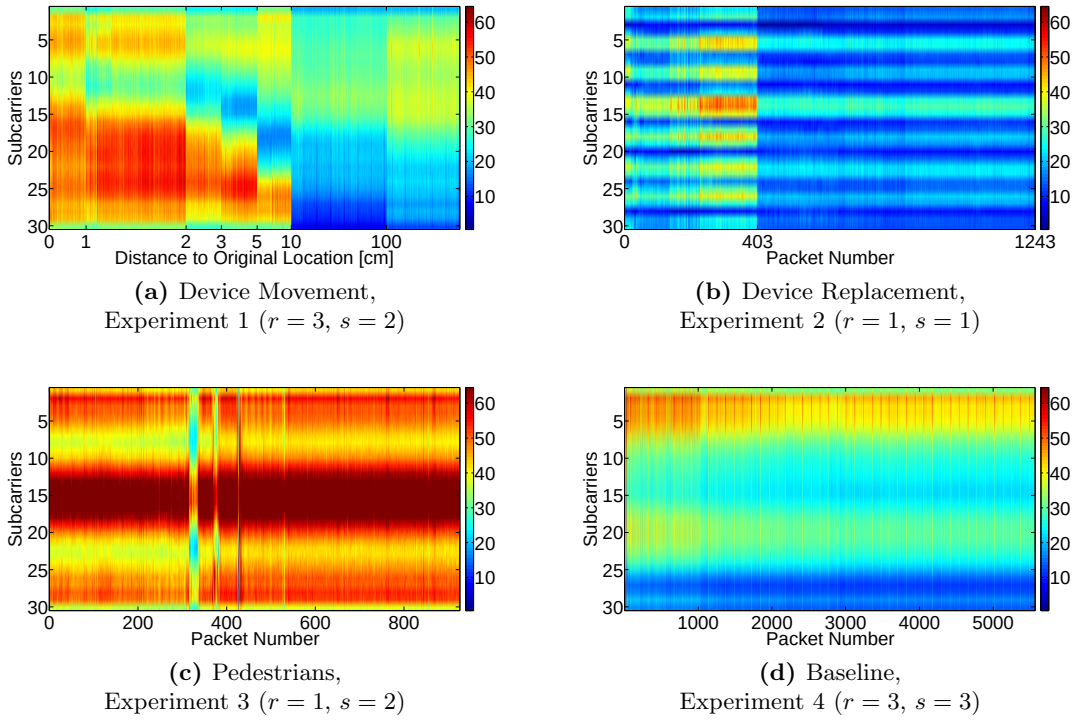
$$Q^{i,t_w} = \begin{cases} 0 & \text{if } \sum_{\substack{j \\ t_j \in (t_i - t_w, t_i)}} Q^j < \text{count}(t_j) \\ 1 & \text{if } \sum_{\substack{j \\ t_j \in (t_i - t_w, t_i)}} Q^j = \text{count}(t_j) \end{cases}$$

Note that *time-wise filtering* used here is different than *time-wise moving average filter* used in single receiver tamper detection. We average distance measurements over a window t_w in the single receiver setting. Here, we apply the filtering to overall decision instead of applying it to distance measurements, and also we do not average the results.

Although time-wise filtering helps to reduce the FPRs, it will also reduce the TPRs. This is a trade-off that needs to be considered. Also, before a decision can be made, data points covering a full window must be collected. This increases the time until a decision can be made (tamper detection is delayed).

5.2 Evaluation

We first evaluate the tamper detection using a single receiver. We show that device movement and device replacement can both be detected. But we also show that false alarms can be triggered by changes in the environment instead of tampering. Then we evaluate multi-receiver tamper detection, of which we propose to overcome false alarms caused by environmental changes.



(a) Device Movement, Experiment 1 ($r = 3$, $s = 2$)

(b) Device Replacement, Experiment 2 ($r = 1$, $s = 1$)

(c) Pedestrians, Experiment 3 ($r = 1$, $s = 2$)

(d) Baseline, Experiment 4 ($r = 3$, $s = 3$)

Figure 5.2: Amplitude CSI values for different experiments and antenna/stream combinations. Warmer colors represents higher amplitude values. Events such as device movement/replacement and environmental changes are visible.

5.2.1 Single Receiver Tamper Detection

For evaluation, we use as receiver an Intel 5300 Network Interface Card (NIC) (3 antennas) which allows us to extract CSI measurements via the *Linux 802.11n CSI Tool* [HHSW11]. As sender, we use either an Intel 5300 NIC or an Apple Mac (3 antennas). Depending on the transmitter decision, each packet is received encoded with up to three spatially independent streams. In all experiments, the transmitter is set to send a packet once a second.

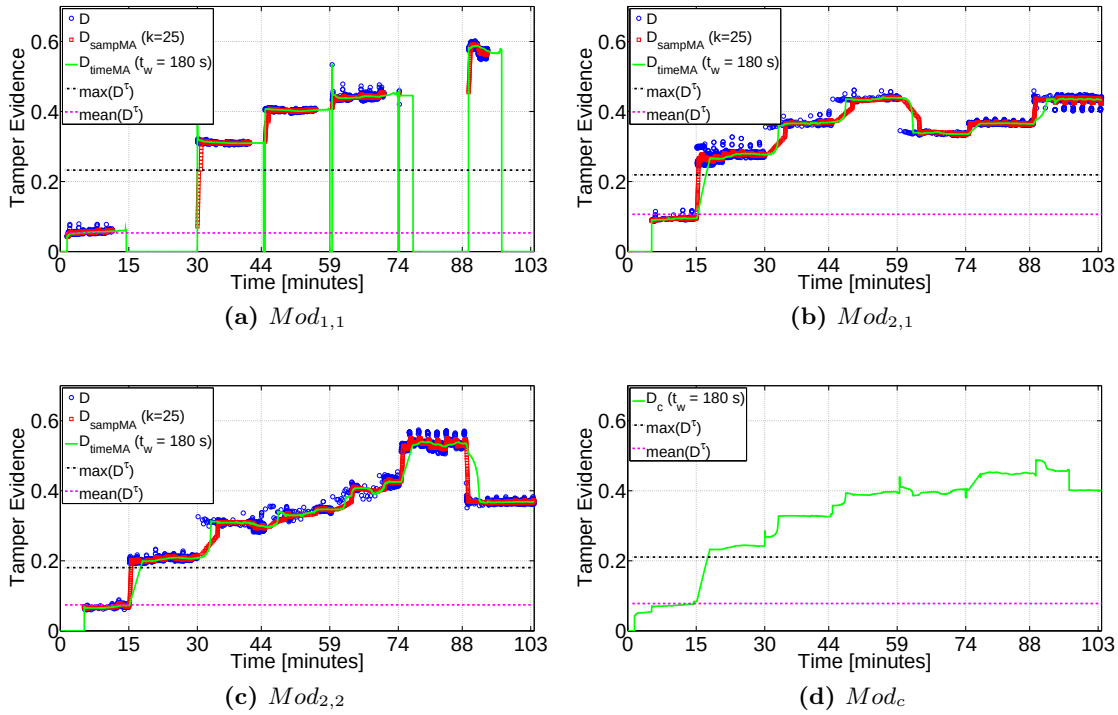
In the first experiment (Experiment 1), we use an iMac as sender with an initial distance of 1.5 m between sender and receiver. After 15 min we start to move the transmitter by 1 cm; after 30 min by 2 cm; after 44 min by 3 cm; after 59 min by 5 cm; after 74 min by 10 cm and finally after 88 min by 100 cm. This experiment is to analyse the detectability of device movement. In the second experiment (Experiment 2), we

analyse detectability of device replacement. An Intel 5300 NIC is used as sender; the distance is 240 cm between sender and receiver. After 10 min the sender NIC is replaced while not altering antenna positions. The third experiment (Experiment 3) is used to analyse the impact of moving pedestrians. An iMac is used as sender with distance 1.5 m between sender and receiver. After 15 min, 16 min and 17 min a person is moving slowly through the communication path (taking about 10 s each time). In the last experiment (Experiment 4), we leave an iMac sender at night transmitting over a distance of 4 m. This experiment serves as baseline without tampering or environmental changes.

Figure 5.2 gives a visual impression of the recorded CSI amplitude values for different experiments for a subset of the overall CSI data. For example, Figure 5.2d shows 30 CSI amplitude values for each incoming packet received at the third antenna $r = 3$ using spatial stream $s = 3$ and being transmitted using $S = 3$ independent spatial streams. Data for other antenna/stream combinations are not shown here but are later used in the tamper detection mechanism. As can be seen in Figure 5.2, the different events within the 4 experimental setups are clearly visible.

5.2.1.1 Experiment 1: Device Movement

As described in Section 5.1.1, we use individual models for each spatial stream setting to analyse the CSI data of each incoming packet. For a received packet using one spatial stream, one model is used ($Mod_{1,1}$). For a received packet transmitted using two spatial streams, two additional models are used (one for each received spatial stream; $Mod_{2,1}$, $Mod_{2,2}$), and for a packet received using three spatial streams, three additional models are used ($Mod_{3,1}$, $Mod_{3,2}$, $Mod_{3,3}$). Thus, 6 spatial models can be created to analyse incoming packets; however, in practice we did not observe a useful number of packets (less than 50 packets during the entire experiment) using 3 spatial streams and, thus, only 3 models become active in this experiment ($Mod_{1,1}$, $Mod_{2,1}$, $Mod_{2,2}$). Figure 5.3 shows the model output for $Mod_{1,1}$, $Mod_{2,1}$, $Mod_{2,2}$ and of the combined Model Mod_c over the duration of the experiment. The output of Mod_c is the tamper-evidence level that we use to decide if a node has been tampered with. Mod_c is created by combining



(a) $Mod_{1,1}$ **(b)** $Mod_{2,1}$

(c) $Mod_{2,2}$ **(d)** Mod_c

Figure 5.3: Experiment 1 - Tamper evidence over time with device movement. Model outputs for packets received with different spatial streams are combined into the overall model to produce a single tamper evidence value.

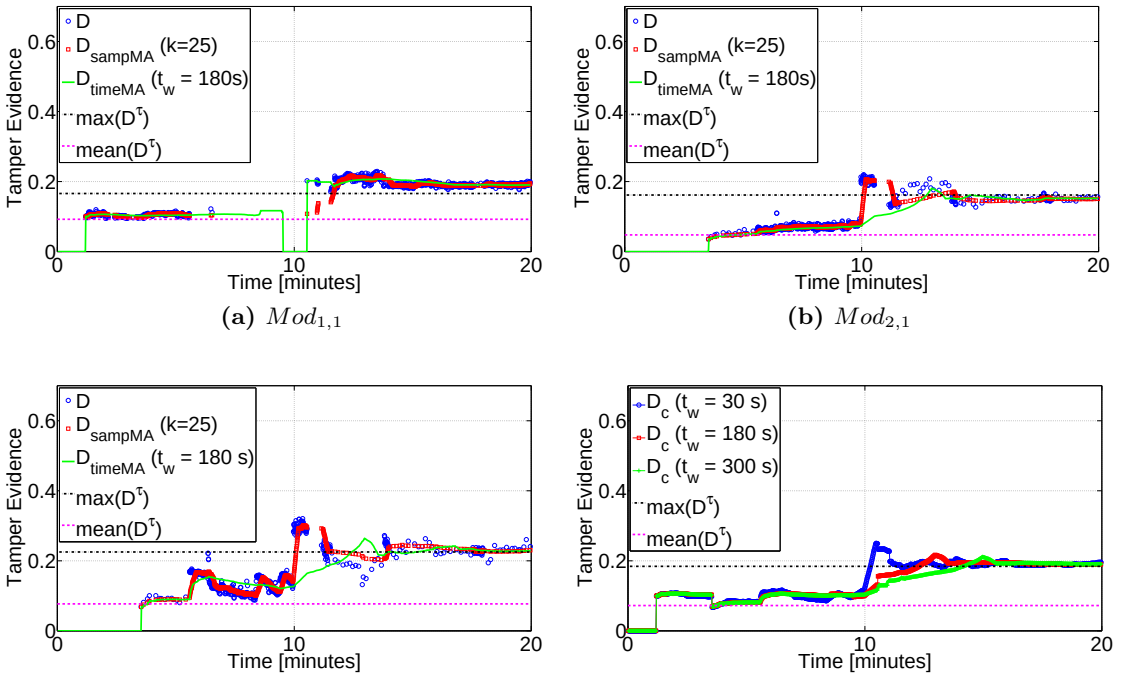
the output of models $Mod_{1,1}$, $Mod_{2,1}$, $Mod_{2,2}$ (see Section 5.1.1).

Before a model becomes operational, it requires a history size of $\tau = 50$ CSI data, which means that, for each model, 50 packets with the appropriate number of spatial streams must be received. A history size of 50 was chosen for the training phase as this provided stable results for our experiments. In a different scenario a shorter training phase might be possible. As the number of streams may change for each packet, a different amount of time is required for each model before it is operational. In this experiment, $Mod_{1,1}$ becomes active first after $t = 88\,\text{s}$, while $Mod_{2,1}$ and $Mod_{2,2}$ become active at $t = 314\,\text{s}$. The three models used for packets with three spatial streams never become active as, throughout the entire experiment, not enough packets of this type are received to fill the history. During this training phase, it must be ensured that no tamper situation is present, which is the case in this experiment. (Tampering starts at $t = 15\,\text{min}$.)

As can be seen in Figure 5.3a, model $Mod_{1,1}$ produces far fewer data points than the other two models. The reason is that the 802.11n transmitter uses two spatial streams for most transmissions, and transmissions using only one spatial transmission are rare. As a result, there are time periods (e.g. between $t = 14\,\text{min}$ and $t = 30\,\text{min}$) where the model output is 0, which means that no recent data for this model is available and, thus, this model in these periods is not useful as a contributor to the overall model Mod_c. However, as long as data packets are received, there is at least one model active contributing to Mod_c and, therefore, a tamper-evidence value is always provided. Only in situations where no packet is received at all would no model be active; however, in a practical setting, a lack of received data (or heart-beat messages) for a period of time may be considered as a tamper situation anyway.

Tampering in the form of movement of the transmitter by 1 cm starts at $t = 15\,\text{min}$. This event can clearly be seen in the tamper-evidence value provided by model Mod_c (see Figure 5.3d). The tamper-evidence value increases from $t = 15\,\text{min}$ onwards with a gradient determined by the window size t_w. A smaller window size would lead to faster increase of the tamper-evidence value. However, as we will show in Experiment 3, it is not always desirable to use a small window size, as a large window helps us to suppress high tamper-evidence levels caused by movement instead of tampering. A threshold could now be selected to decide a tamper-evidence level, beyond which the transmitter would be no longer considered to be trustworthy. The *max* and *mean* values shown in Figure 5.3d might help for threshold selection. For example, a threshold set to twice the value of *mean* might be a useful selection.

As shown in Figure 5.3d, all tested positions of the transmitter different to the original position are clearly visible as a tamper situation. It has to be noted that the tamper-evidence value, which depends on movement distance, is not a monotone increasing function. An increase in distance to the original device position may lead to a reduction of the tamper-evidence value. However, no movement in our experiment set leads to a tamper-evidence value that is close to the value associated with the un-tampered situation.



(a) $Mod_{1,1}$ **(b)** $Mod_{2,1}$

(c) $Mod_{2,2}$ **(d)** Mod_c, window size $t_w = 30\,\text{s}$, $t_w = 180\,\text{s}$ and $t_w = 300\,\text{s}$

Figure 5.4: Experiment 2 - Tamper evidence over time in an environment with device replacement (tampering) at $t = 10\,\text{min}$. Tamper evidence levels are rising faster for smaller window sizes t_w.

From $t = 74\,\text{min}$ onwards, we see a small periodic change in the output of $Mod_{2,1}$ and $Mod_{2,2}$. This periodic change is potentially caused by an interferer (such as a heating system) active during the experiment.

5.2.1.2 Experiment 2: Device Replacement

The results of this experiment are shown in Figure 5.4. We did not observe a useful number of packets using 3 spatial streams and, thus, only 3 models become active in this experiment ($Mod_{1,1}$, $Mod_{2,1}$, $Mod_{2,2}$). Figure 5.4d shows the tamper-evidence value which is the output of the combined model Mod_c. The output of Mod_c is shown for different window sizes t_w. $Mod_{1,1}$ becomes first after $t = 72\,\text{s}$, while $Mod_{2,1}$ and $Mod_{2,2}$ become active after $t = 213\,\text{s}$ when enough packets using 2 spatial streams are received to fill the history of $\tau = 50$ values.

At $t = 10\,\text{min}$ the device is tampered with and the NIC is replaced. In this experiment, we use external antennas connected to the NIC to ensure that only the device is replaced while all antennas remain at exactly the same position.

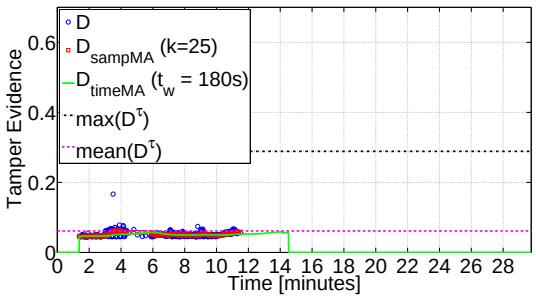
We can clearly see in Figures 5.4b and 5.4c that the tamper-evidence value D changes immediately. The output D_{timeMA} is changing more slowly, as an average of values of D over the past duration of $t_w = 180\,\text{s}$ is used. Figure 5.4d shows the output of the combined model for different window sizes t_w (see Section 5.1.1). Obviously, using a smaller window size leads to faster detection, but the algorithm would be more prone to false alarms as outliers would make a greater contribution to the tamper-evidence value.

A threshold at twice the average tamper-evidence value, *mean*, could be used as alarm threshold. In this case, the device replacement would be indicated after $t = 605\,\text{s}$ when using a window size of $t_w = 30\,\text{s}$. With a window size of $t_w = 300\,\text{s}$, the replacement would be indicated substantially after $t = 678\,\text{s}$. However, using a smaller window size is not without cost, as we will show in the next experiment.

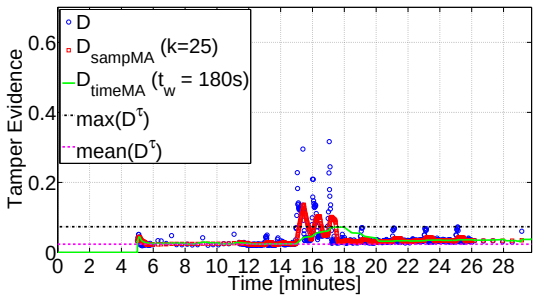
5.2.1.3 Experiment 3: Pedestrians

In this experiment, a person walks slowly 3 times through the line of sight between sender and receiver. The first walk is at $t = 15\,\text{min}$, the second at $t = 16\,\text{min}$, and the third at $t = 17\,\text{min}$. Figure 5.5 shows the output of the models. Figure 5.5g shows the tamper-evidence value, which is the output of the combined model Mod_c (see Section 5.1.1). This is shown for different window sizes t_w.

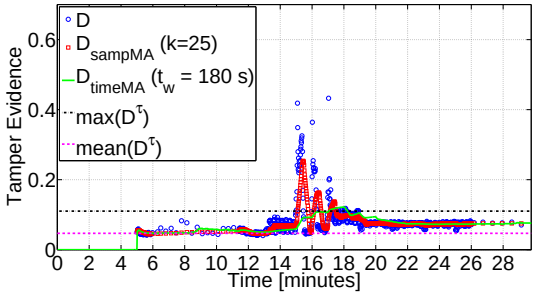
The walks are visible in the tamper-evidence value. However, for all selected window sizes the tamper-evidence value returns after a while to the base line, indicating no tamper situation. This is different to the previous two situations of device movement and device replacement where the tamper-evidence value does not return to the original value. As can be seen in Figure 5.5g, the window size influences the maximum tamper-evidence value that is reached. Large windows suppress high tamper-evidence values for environmental changes such as those caused by a pedestrian. When using a threshold at twice the average tamper-evidence value, *mean*, an alarm situation can be avoided with



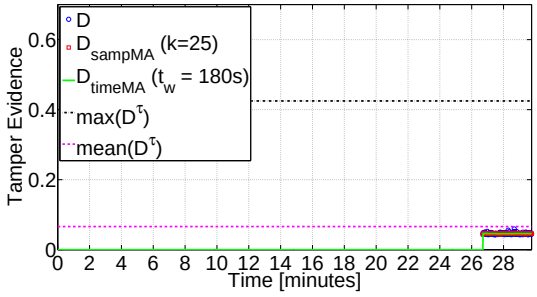
(a) $Mod_{1,1}$



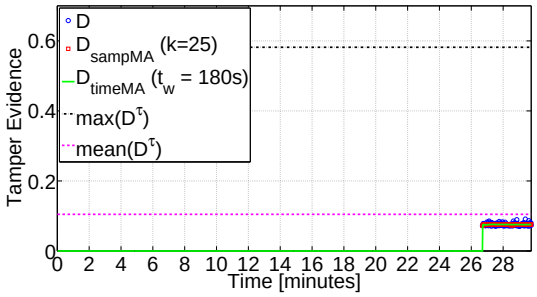
(b) $Mod_{2,1}$



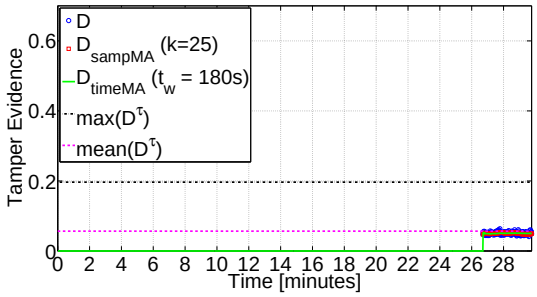
(c) $Mod_{2,2}$



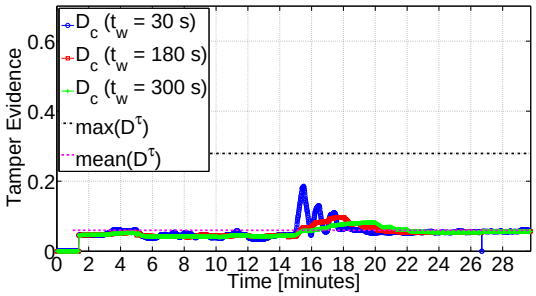
(d) $Mod_{3,1}$



(e) $Mod_{3,2}$

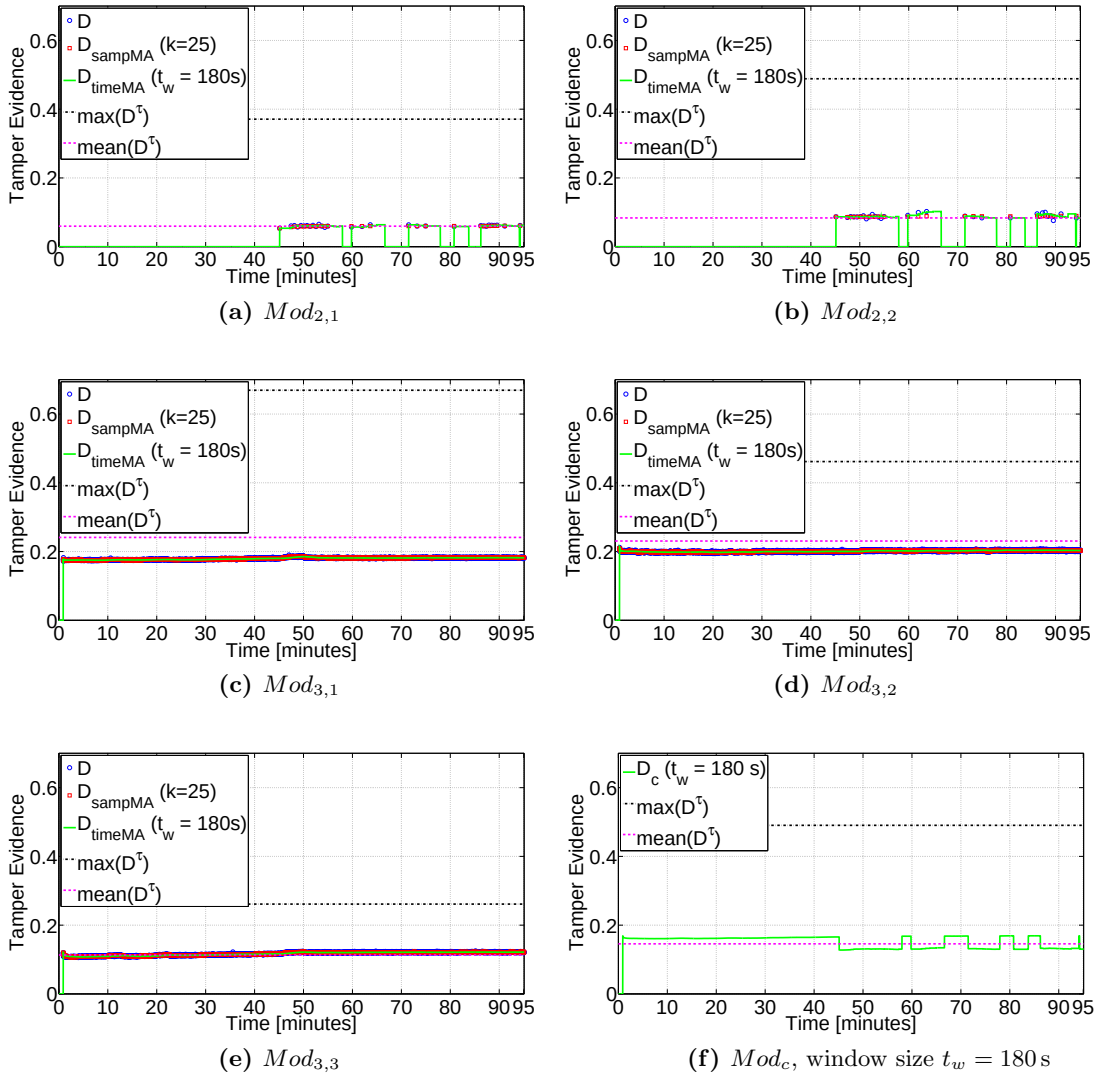


(f) $Mod_{3,3}$



(g) Mod_c, window size $t_w = 30\,\text{s}$, $t_w = 180\,\text{s}$ and $t_w = 300\,\text{s}$

Figure 5.5: Experiment 3 - Tamper evidence over time in an environment with movement at $t = 15\,\text{min}$, $t = 16\,\text{min}$ and $t = 17\,\text{min}$. Tamper evidence levels are lower for larger window sizes t_w.



(a) $Mod_{2,1}$ **(b)** $Mod_{2,2}$

(c) $Mod_{3,1}$ **(d)** $Mod_{3,2}$

(e) $Mod_{3,3}$ **(f)** Mod_c, window size $t_w = 180\,\mathrm{s}$

Figure 5.6: Experiment 4 - Tamper evidence in a tamper free environment. The tamper evidence value remains static at a low level over a long period of time.

a window size of $t_w = 300\,\mathrm{s}$, but not for a window size of $t_w = 30\,\mathrm{s}$.

Comparing the results of this experiment with the results of the previous two experiments, it becomes clear that the selection of the window size over which analysis results are averaged has an influence on i) detection time and ii) suppression of false alarms.

5.2.1.4 Experiment 4: Baseline

This experiment was carried out at night to avoid fluctuations of channel characteristics due to moving people and interference by electronic devices. Figure 5.6 shows the output

of the models. We did not observe a useful number of packets using 1 spatial stream and, therefore, $Mod_{1,1}$ was not active in this experiment. The output of Mod_c when using a window size of $t_w = 180\,\text{s}$ is, as seen in Figure 5.6f, a flat line without significant variation; the maximum tamper-evidence value for a window size of $t_w = 180\,\text{s}$ is 0.17, and the minimum is 0.15 within the 1.5 h duration. This demonstrates that a continuous upgrade of the used history data (a retraining of the model) is not required. It can be expected that, in a static and tamper-free environment, the tamper-evidence value is stable.

5.2.1.5 Discussion

Detection: Our experiments have shown that device movement and device replacement can both be detected. Even small device movements of just 1 cm are clearly visible in the computed tamper-evidence value.

False Alarms: False alarms can be triggered by changes in the environment instead of tampering. As shown, by selecting a large window size t_w, false alarms can be avoided in silent environment while reducing the detection speed. We will investigate false alarms more in the following section where we evaluate multi-receiver tamper detection.

5.2.2 Multi-Receiver Tamper Detection

We start our evaluation for multi-receiver tamper detection with a controlled movement experiment. In this experiment only one person is present and movement of the person in the deployment area is known. This experiment is used to analyse how distance values extracted from the CSI are affected by movement. Furthermore, the experiment shows that the use of multiple receivers is an effective measure for distinguishing movement and tamper events.

Thereafter an experiment with uncontrolled movement is carried out. Nodes are deployed in an office environment in which office workers move around during the day. At night there is less activity but occasionally people are present. In this experiment we do not record the number of people or their movements. The purpose of this experiment

Figure 5.7: A laptop and an antenna used in the experiments. Only the antenna is tampered (moved or rotated) during the experiments.

is to see how different configurations of the tamper detection algorithm handle realistic environments with different levels of activity. We use this experiment as well to evaluate several different tamper situations to see how likely it is that different tamper situations are to be detected.

We use off-the-shelf Toshiba NB250-108 laptops equipped with an Intel 5300 NIC for our experiments. The laptops run Ubuntu 14.04 LTS with the 3.5.7 kernel. We use the Linux 802.11n CSI Tool [HHSW11] to extract CSI from the Intel 5300 NICs. Each NIC is equipped with a triple TP-Link TL-ANT2403N 802.11n omni-directional antenna. Figure 5.7 shows one of the laptops with antenna. To induce tampering, the antenna is moved or rotated.

5.2.2.1 Controlled Movement

We use 4 receivers and one transmitter deployed in an office building as shown in Figure 5.8. The transmitter sends broadcast beacons with a 1 packet/second interval. All receivers listen for these packets and extract the CSI from incoming packets.

A history size of $\tau = 100$ CSI readings is required before receivers calculate distance values. Figure 5.9 shows as an example how the CSI amplitude values develop over time. Values as received by the 2^{nd} antenna of Receiver 3 are shown; similar data is available

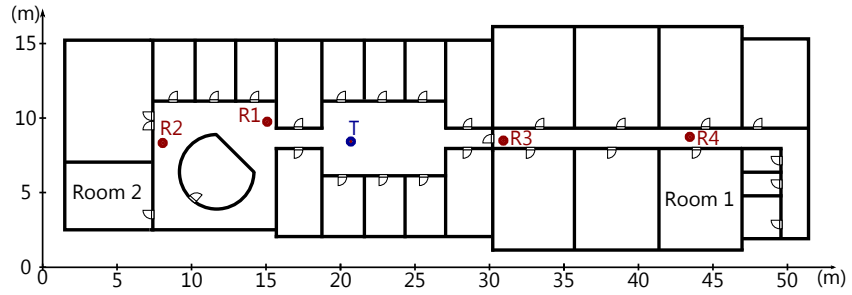


Figure 5.8: Controlled movement experiment layout. Receivers are shown as R1-4, and transmitter is shown as T. The environment is static during the experiment. A person is walking occasionally or waiting in Room 1 or Room 2.

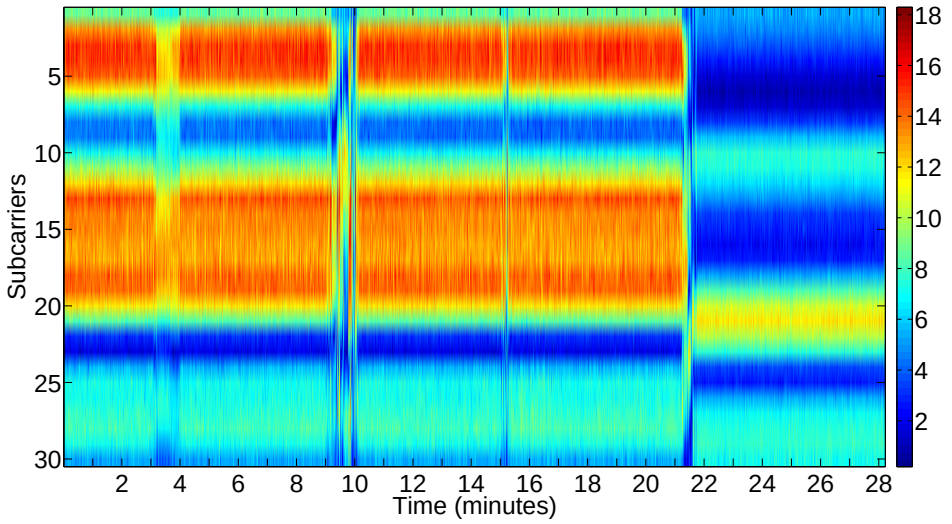
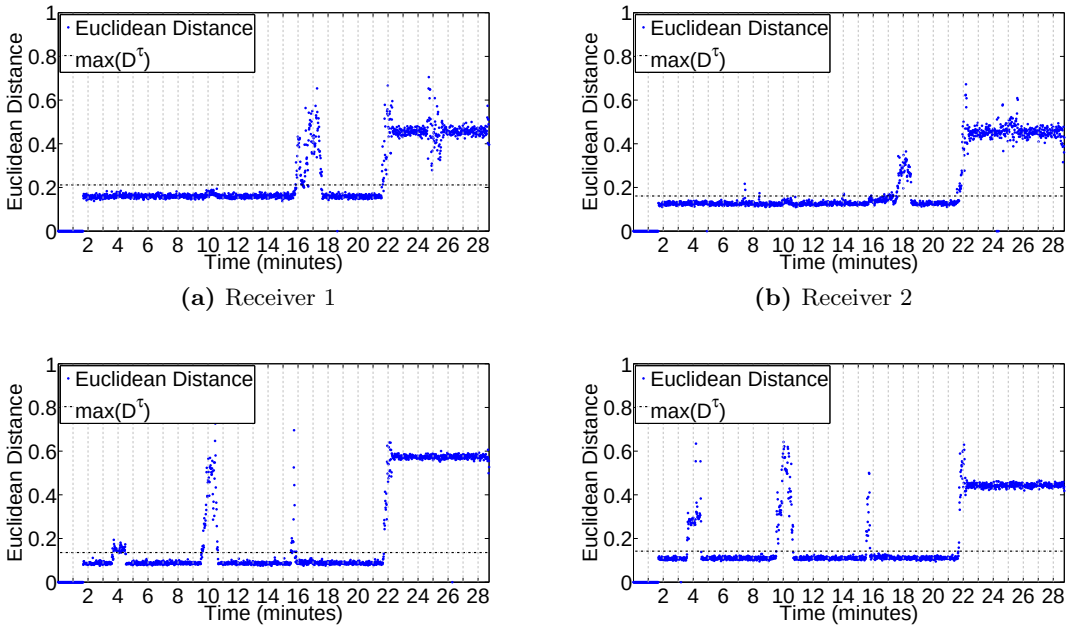


Figure 5.9: CSI amplitude values of 2^{nd} antenna of Receiver 3 during the controlled movement experiment. Amplitude values change occasionally due to movement until a tamper event at time $t = 21.5\,\text{min}$.

for the other 2 antennas. The x-axis shows the time in minutes, and the y-axis shows the amplitude at each OFDM subcarrier. It can be seen that amplitude values change occasionally due to movement of a person until tampering happens at time $t = 21.5\,\text{min}$ which is clearly visible. We detail movement patterns and tampering events in the next paragraphs.

Figures 5.10, 5.11, and 5.12 show the distance value development over time at all 4 receivers using Euclidean, Mahalanobis, and Earth Mover's distance algorithms. We show the Maximum Distance threshold $\max(D^\tau)$ in the figures. Figure 5.13 shows the resulting tamper detection decision q_i considering each receiver individually when using



(a) Receiver 1 **(b)** Receiver 2

(c) Receiver 3 **(d)** Receiver 4

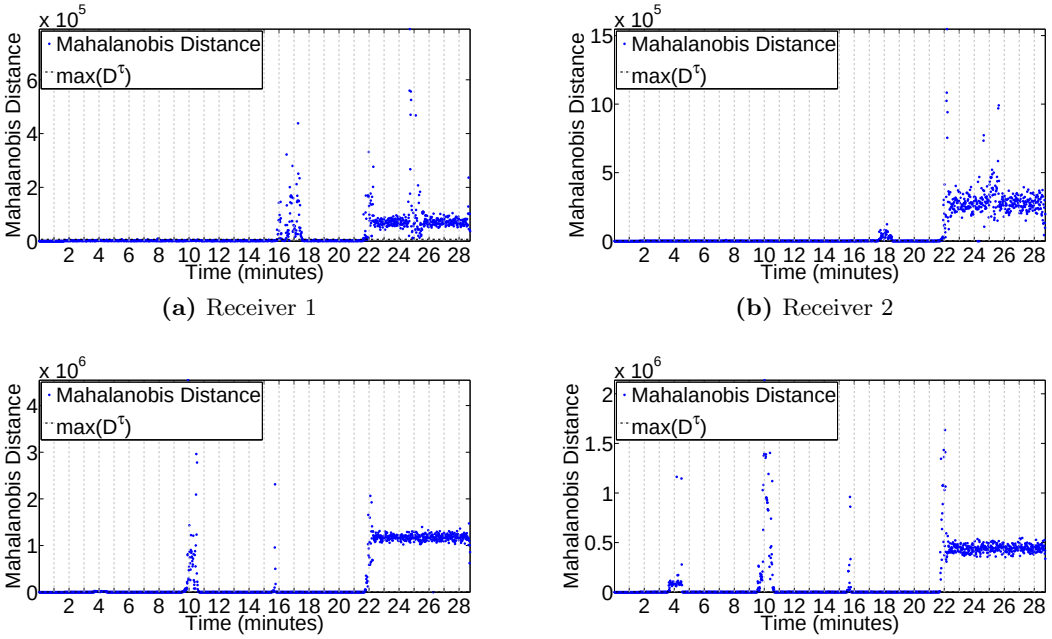
Figure 5.10: Euclidean distance in the controlled movement experiment. Tampering is induced at $t = 21.5$ min. Distance values at each receiver increase occasionally until this time due to movement.

the Euclidean distance (We do not show the result for Mahalanobis, and Earth Mover as these are very similar). The overall tamper decision Q^i considering all 4 receivers combined is shown in Figure 5.14 for Euclidean, Mahalanobis, and Earth Mover's distance algorithms. The tamper event at $t = 21.5$ min is clearly identified; movement events before this time do not lead to a tampering report.

All the receivers start to report distance values after 100 seconds. At the beginning of the experiment, a person is waiting in Room 1.

At time $t = 3.5$ min, the person walks close to Receiver 4, waits next to Receiver 4 for a minute, and then enters Room 1. We can see in the figures that this affects distance values at Receiver 4 and very slightly at Receiver 3. Since we do not see any increase on distance values of Receiver 1 and Receiver 2, the system will not create a false alarm when considering data from all receivers.

At time $t = 9.5$ min, the person walks very close to Receiver 3, waits next to Receiver 3 for a minute, and then enters Room 1. This affects the distance values of Receiver 3 and



(a) Receiver 1 **(b)** Receiver 2

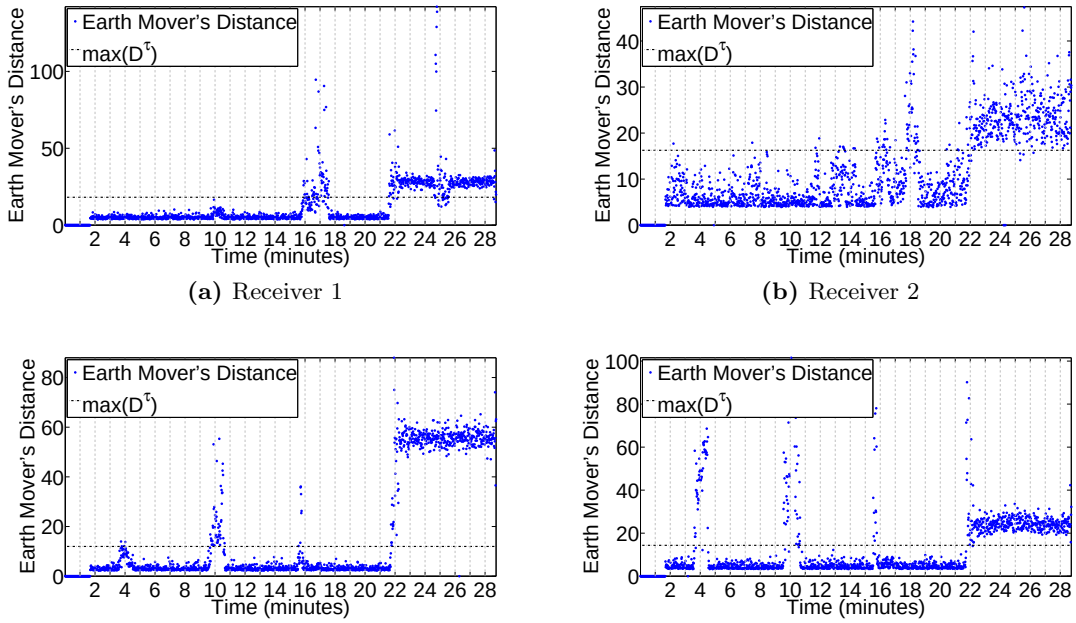
(c) Receiver 3 **(d)** Receiver 4

Figure 5.11: Mahalanobis distance in the controlled movement experiment. Tampering is induced at $t = 21.5$ min. Distance values at each receiver increase occasionally until this time due to movement.

Receiver 4. Again, there is no significant change on the distance values of Receiver 1 and Receiver 2.

At time $t = 15.5$ min, the person walks very close to Receiver 1, waits next to Receiver 1 for two minutes, then moves to Receiver 2 at time $t = 17.5$ min and waits next to Receiver 2 for a minute, and finally enters Room 2. We can see from the figures that distance values increase for Receiver 3 and Receiver 4 at time $t = 15.5$ min. This is because the person needs to pass by Receiver 3 and Receiver 4 to go to Receiver 1. Distance values for Receiver 3 and Receiver 4 go back to normal as distance values for Receiver 1 increase. The system does not create a false alarm when considering all receivers together as it does not see distance value increases above the threshold at all receivers at the same time.

At time $t = 21.5$ min, the person rotates the antenna of the transmitter 90° clockwise, and enters Room 2. Now the system creates an alarm, since distance values of all the receivers increase.



(a) Receiver 1 **(b)** Receiver 2

(c) Receiver 3 **(d)** Receiver 4

Figure 5.12: Earth Mover's distance in the controlled movement experiment. Tampering is induced at $t = 21.5$ min. Distance values at each receiver increase occasionally until this time due to movement.

At time $t = 24.5$ min, the person walks very close to Receiver 1, waits next to Receiver 1 for a minute, and enters Room 2. Distance values of Receiver 1 and Receiver 2 slightly change during this period but remain above the threshold. Thus, the tamper situation remains. The experiment terminates around time $t = 28.5$ min.

Figure 5.13 shows decisions q_i for each receiver (considering individual receiver results) when using Euclidean distance algorithm and $\max(D^\tau)$ as the threshold. The x-axis shows the time in minutes, and y-axis shows the decisions. Decision 0 means there is no tampering, and decision 1 means there is tampering.

From Figure 5.13 we can see that false alarms exist when considering individual receivers. FPRs are 0.08, 0.06, 0.11, and 0.12 for Receivers 1, 2, 3, and 4, respectively.

However, from Figure 5.14 we can see that multi-receiver tamper detection provides the desired results. All the distance algorithms provide FPR = 0 result. Both Euclidean and Mahalanobis distance algorithms have a TPR = 1, however, the Earth Mover's distance algorithm provides only TPR = 0.94.

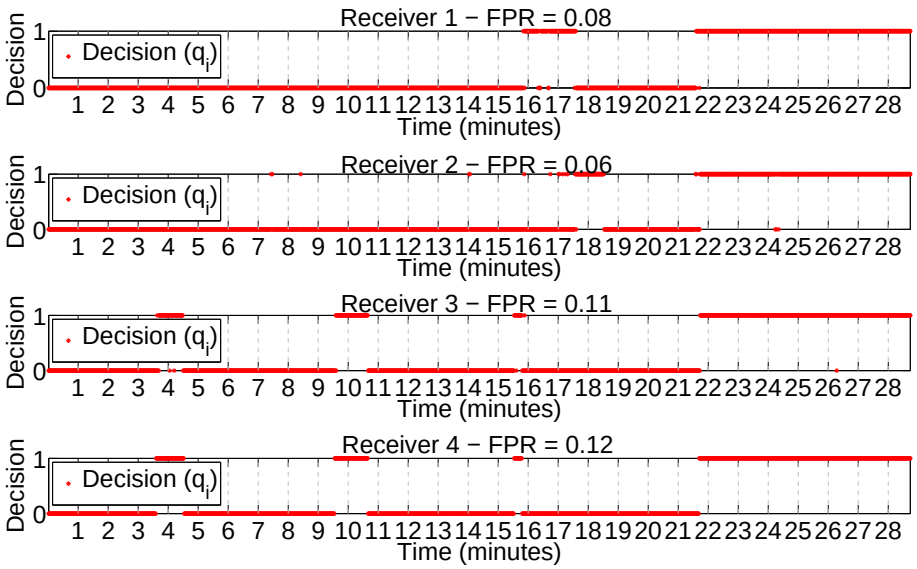


Figure 5.13: Tamper decisions (q_i) for each individual receiver in the controlled movement experiment using Euclidean distance and $\max(D^\tau)$ as threshold. False alarms due to movement are present before the tampering event at $t = 21.5\,\text{min}$.

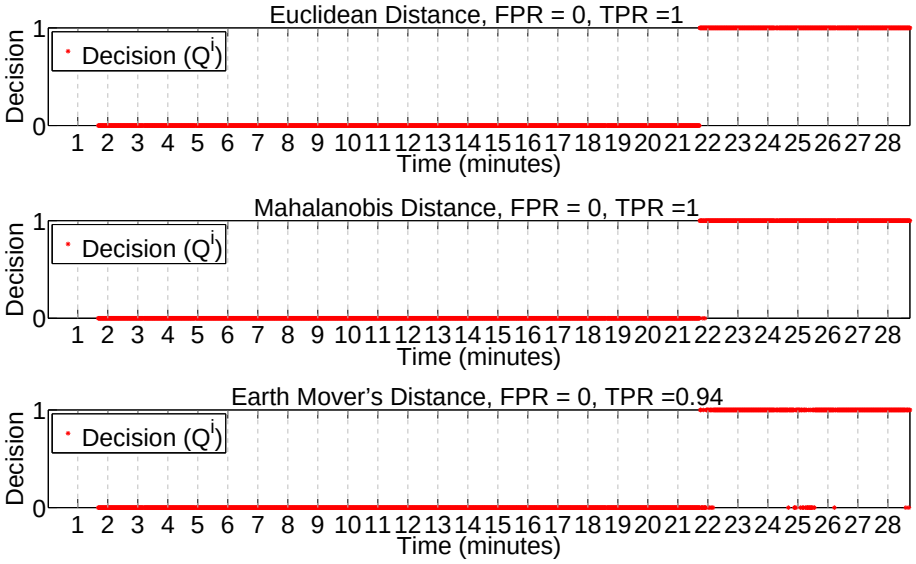


Figure 5.14: Multi-receiver tamper decisions (Q^i) for the controlled movement experiment. All receivers are taken into account and $\max(D^\tau)$ is used as threshold. False alarms are avoided (FPR = 0) while the tamper event is correctly identified (For Euclidean and Mahalanobis distance algorithms with TPR = 1 and for the Earth Mover's distance algorithm with TPR = 0.94).

Euclidean, Mahalanobis, and Earth Mover's distance algorithms perform similarly. This is somewhat surprising as they operate quite differently and one would have expected that more complex distance algorithms capturing more information would provide better

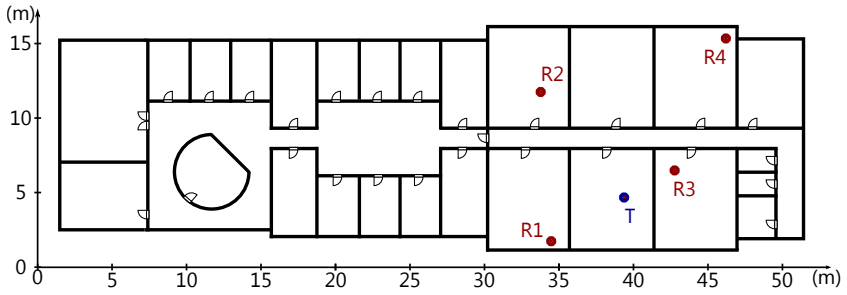


Figure 5.15: Uncontrolled movement experiment layout. People are moving in the rooms and in the corridor inducing CSI variations.

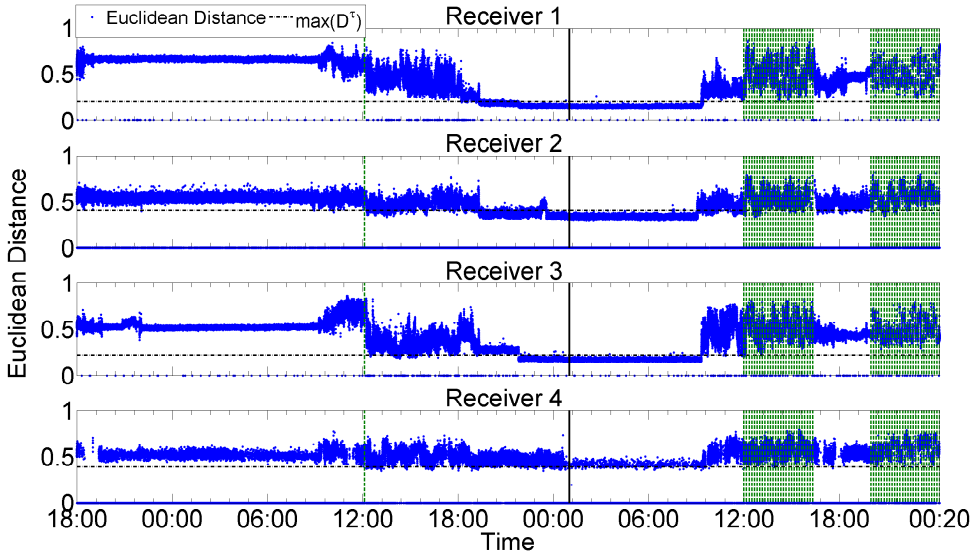


Figure 5.16: Euclidean distance for the uncontrolled movement experiment. Tamper events are indicated with vertical lines. Distance values of each receiver show tampering and also movement during office hours.

results.

5.2.2.2 Uncontrolled Movement

The uncontrolled movement experiment is carried out in the same environment as the controlled movement experiment. However, the transmitter and the 4 receivers are located in different rooms as shown in Figure 5.15. People use the offices and corridors and may also move chairs and other objects. Movement events of people and objects may lead to false tamper detection which we aim to avoid. The transmitter sends broadcast beacons for tamper detection with a 1 packet/second interval.

The experiment starts while the transmitter is in a tampered state. The transmitter

Day				Night			
Time	**Type**	**Time**	**Type**	**Time**	**Type**	**Time**	**Type**
12:00	90° cw	12:10	180° cw	20:00	90° cw	20:10	180° cw
12:20	270° cw	12:30	360°(0°) cw	20:20	270° cw	20:30	360°(0°) cw
12:40	1 cm up	12:50	1 cm right	20:40	1 cm up	20:50	1 cm right
13:00	1 cm down	13:10	1 cm left	21:00	1 cm down	21:10	1 cm left
13:20	2 cm up	13:30	2 cm right	21:20	2 cm up	21:30	2 cm right
13:40	2 cm down	13:50	2 cm left	21:40	2 cm down	21:50	2 cm left
14:00	3 cm up	14:10	3 cm right	22:00	3 cm up	22:10	3 cm right
14:20	3 cm down	14:30	3 cm left	22:20	3 cm down	22:30	3 cm left
14:40	4 cm up	14:50	4 cm right	22:40	4 cm up	22:50	4 cm right
15:00	4 cm down	15:10	4 cm left	23:00	4 cm down	23:10	4 cm left
15:20	4 cm up	15:30	5 cm right	23:20	4 cm up	23:30	5 cm right
15:40	5 cm down	15:50	5 cm left	23:40	5 cm down	23:50	5 cm left
16:00	30 cm left	16:10	60 cm left	00:00	30 cm left	00:10	60 cm left
16:20	Original			00:20	Original		

Table 5.1: The different tamper events and their times.

antenna is rotated by 90° anticlockwise from its intended position. After a while the antenna is rotated to its intended position, i.e., it is transited to the untampered state. In both states (tampered and untampered) data for threshold selection as described in Section 5.1.2.1 is collected. A history size of $\tau = 100$ CSI data is also collected during the untampered state when we can ensure that the environment is free of movement. Then, the setup is left for a long time in an untampered state. Then different tamper states are induced during day time and later as well at night. The Euclidean distance metric is used for evaluation. Figure 5.16 shows this distance value over time for all 4 receivers.

The experiment starts at 18:00 on 18th of May when the antenna of the transmitter is rotated 90° anticlockwise from its intended position. The antenna is rotated to its intended position at 12:06 on 19th of May. Several tamper situations are induced from 12:00 until 16:20 on May 20th (see Table 5.1 for details). To compare tamper detection in busy and more quiet periods, the same tamper events are applied again starting 20:00 on May 20th. Training CSI values are collected at 01:00 on May 20th (shown as vertical black line in Figure 5.16). Maximum distance values within the training data are also shown in the figure as $\max(D^\tau)$. The distance values of the packets not received by a receiver are shown as 0 in Figure 5.16. Tamper events are shown in Figure 5.16 with vertical dashed lines at their corresponding times.

FPR		TPR	
Night time (01:02 - 08:00)	0	Night time (20:00 - 00:30)	0.999
Day time (08:00 - 12:00)	0.831	Day time (12:00 - 16:00)	0.966
Overall (01:02 - 12:00)	0.769		

Table 5.2: FPRs and TPRs when using $\max(D^{\tau})$ as the threshold.

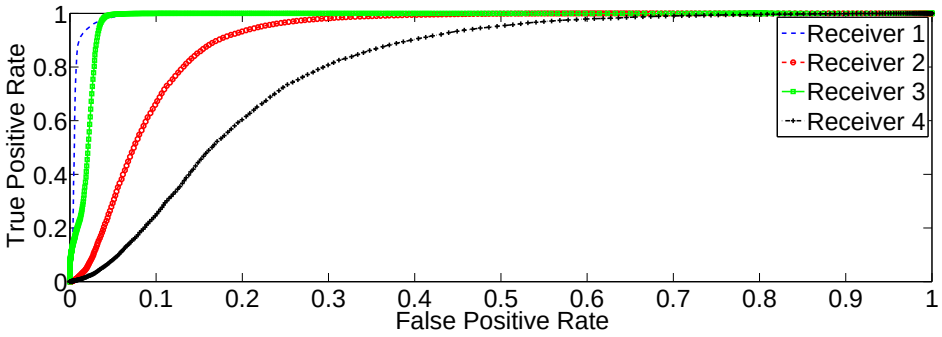


Figure 5.17: ROC curve of the 4 receivers.

Maximum Distance Threshold Figure 5.16 shows that distance values are above $\max(D^{\tau})$ during office hours when there is no tampering. In a busy environment, $\max(D^{\tau})$ is too sensitive and a high FPR is the consequence. Using $\max(D^{\tau})$ as the threshold results in FPRs and TPRs as shown in Table 5.2. FPR results are divided into three time regions: (i) Night time between 01:02 and 08:00 when the experiment environment is less dynamic, (ii) Day time between 08:00 and 12:00 when the experiment environment is dynamic, and (ii) Overall (day and night combined). TPR results are also divided into two time regions: (i) Night time between 20:00 and 00:30 when the experiment environment is less dynamic, (ii) Day time between 12:00 and 16:00 when the experiment environment is dynamic. TPRs from different tamper events are averaged resulting in a single TPR value (We will analyse TPR of individual tamper states later). Although we obtain a high TPR for both time durations and also 0 FPR during night time, we observe a high FPRs during day time. We thus conclude that using the maximum distance threshold $\max(D^{\tau})$ is not suitable for busy environments.

Equal Error Rate Threshold The experiment starts while the transmitter is in a tampered state, then it is transited to untampered state after a while. We can use the

Tampered state	18:00 18th of May - 12:06 19th of May
Untampered state	12:06 19th of May - 01:00 20th of May

Table 5.3: Time ranges of tampered and untampered states for ROC calculation.

Receiver 1	Receiver 2	Receiver 3	Receiver 4
0.032	0.148	0.036	0.258

Table 5.4: EERs for all the receivers in the uncontrolled experiment.

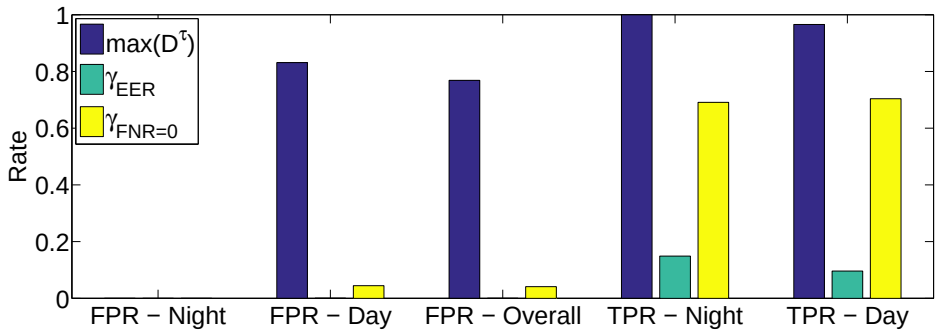


Figure 5.18: FPRs and TPRs with different thresholds. FPRs are always 0 during the night time. $\max(D^{\tau})$ gives high FPRs. γ_{EER} reduces both FPRs and TPRs. $\gamma_{FNR=0}$ gives more balanced results.

distance values from the initial tampered and untampered state to calculate the ROC curve. The ROC curve can then be used to select a threshold based on the desired FPR and TPR rates as previously discussed in Section 5.1.2.1. Table 5.3 shows the time ranges of tampered and untampered states for ROC calculation. Figure 5.17 shows the ROC curve of our 4 different receivers during this training time. Thresholds ranging from 0.001 to 1 with 0.001 intervals are applied to the distance values.

We calculate the EER for each receiver. The EER is the rate where the FPR and FNR are equal. Table 5.4 shows the EERs for all the receivers. Figure 5.18 shows the achievable FPRs and TPRs for the overall decision Q^i using this threshold γ_{EER}.

γ_{EER} reduces the FPRs dramatically. The FPR during the day is 0.00024 and it averages 0.00022 over the entire experiment. Unfortunately, the resulting detection capability of the system is low; TPRs of only 0.148 during night time and 0.096 during day time are achieved. However, for some applications it might be acceptable to have such a low TPR.

	Receiver 1	Receiver 2	Receiver 3	Receiver 4
$\max(D^\tau)$	0.204	0.406	0.219	0.390
γ_{EER}	0.577	0.519	0.500	0.507
$\gamma_{FNR=0}$	0.419	0.386	0.355	0.396

Table 5.5: Threshold values for each receiver.

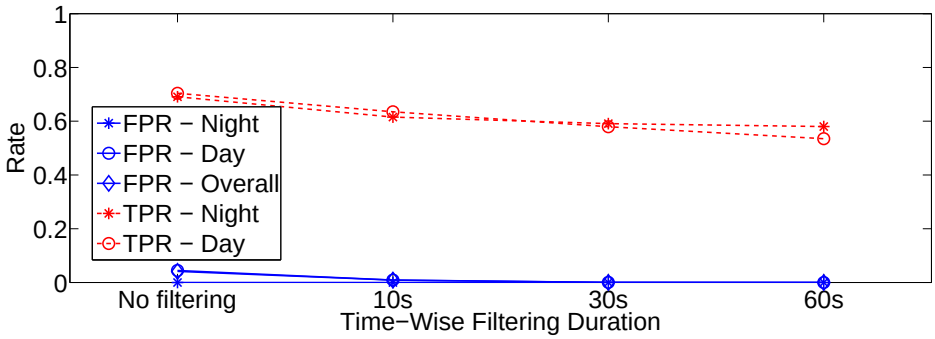


Figure 5.19: Effect of time-wise filtering on FPRs and TPRs when using $\gamma_{FNR=0}$ as the threshold. $t_w = 60\,\text{s}$ reduces FPRs to 0, but it also reduces TPRs.

Zero False Negative Threshold If we select a threshold lower than the EER threshold, we may have a chance to increase the TPR. We select the maximum threshold where the ROC curve gives a FNR = 0, $\gamma_{FNR=0}$. Table 5.5 shows the all thresholds for each receiver, and Figure 5.18 shows the result Q^i based on the threshold $\gamma_{FNR=0}$. This threshold helps to increase TPRs (0.69 during night and 0.7 during the day), but we also increase the FPRs (0.045 during day and 0.041 overall). This threshold achieves a good TPR but the FPR is too high for many practical applications. We apply time-wise filtering to address this issue.

Time-Wise Filtering We use window sizes of $t_w \in \{10, 30, 60\}$ seconds, and the decision is made that there is tampering when it was decided there was tampering for all individual packets within the window. Figure 5.19 shows the effect of time-wise filtering over FPRs and TPRs when using $\gamma_{FNR=0}$ as the threshold. Increasing t_w decreases the FPR. The FPR is reduced from 0.045 to 0 during day time, and it is reduced from 0.04 to 0 overall when $t_w = 60\,\text{s}$. However, using a 60 s window reduces the average TPR from 0.69 to 0.58 during night time, and from 0.7 to 0.53 during day time.

Clearly, it is possible even in a busy environment to use CSI-based tamper detection without risking false alarms while detecting a reasonable number of tamper situations.

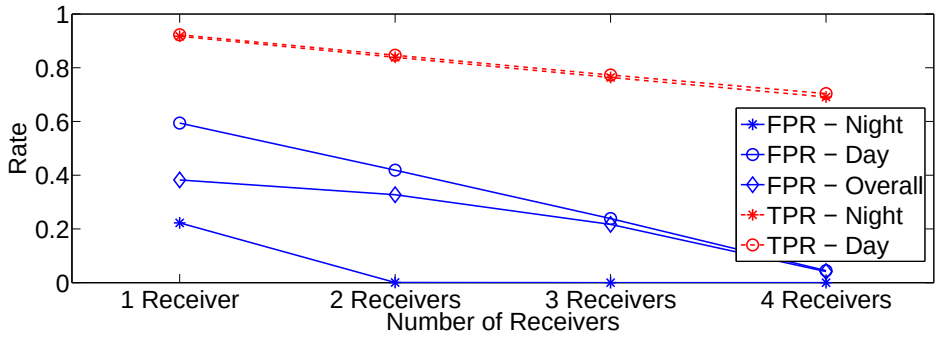


Figure 5.20: Effect of number of receivers to make a decision on FPRs and TPRs, when using $\gamma_{FNR=0}$ as the threshold and without using time-wise filtering. Increasing the number of receivers reduces both FPR and TPR.

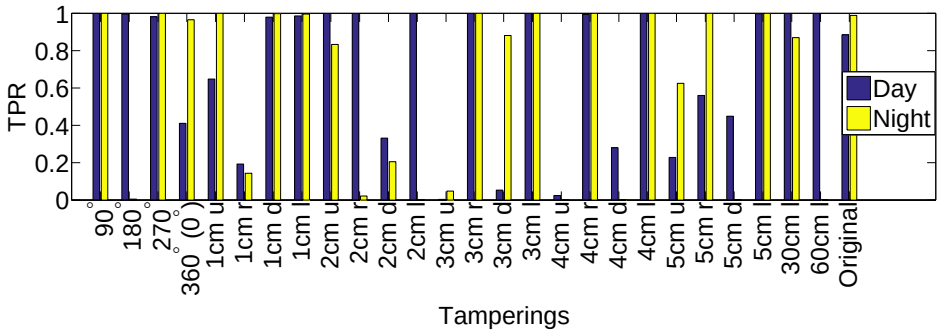
Using Multiple Receivers Until now, all of the decisions were made by using reports from all the receivers. All of the receivers needed to receive the packet, and that packet had to provide a distance above the thresholds for all receivers. The question is how many receivers should be used and what the contribution is of each additional receiver to FPRs and TPRs.

We used a total of $N = 4$ receivers in the experiment. We can use 1, 2, 3, or all of them to make a tamper decision. Figure 5.20 shows the tamper decision results when using $\gamma_{FNR=0}$ as the threshold and without using time-wise filtering. Results are averaged for different combinations of receivers for a given number of receivers, n. For example, if we use $n = 3$ we calculate the results using receivers {1, 2, 3}, {1, 2, 4}, {1, 3, 4}, and {2, 3, 4} in groups and take the average.

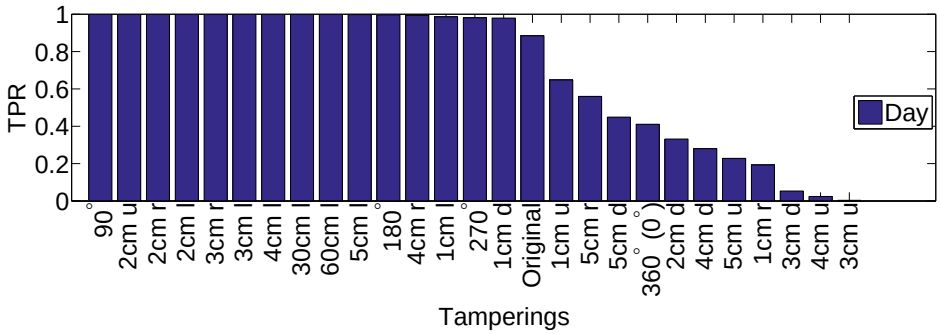
Figure 5.20 shows that using an increasing number of receivers decreases the FPRs. If we use more than 4 receivers to make a decision, we might obtain even better FPRs. However, using more receivers also decreases TPRs.

Tamper Event Detectability Previously, all the TPRs from different tamper events were averaged and shown as a single result. In this section, we analyse individual TPRs for the different tamper states. We applied different tamper events during the day and night. These tamper events are shown in Table 5.1.

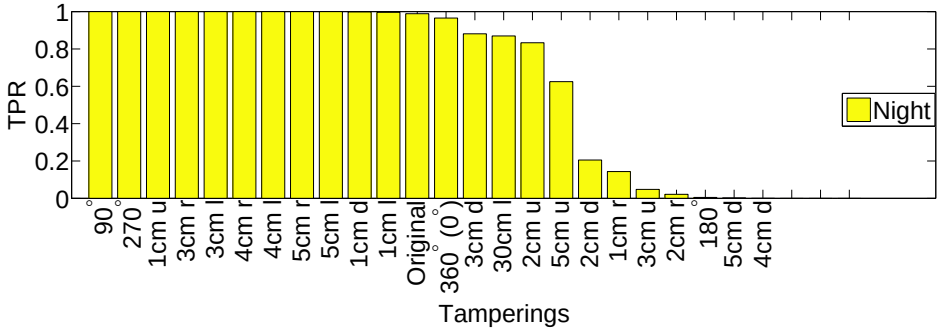
Figure 5.21 shows the results when using $\gamma_{FNR=0}$ as the threshold and without using time-wise filtering. All the receivers are used to make a decision. Some receivers could



(a) Tamperings during the day and the night times.



(b) Tamperings during the day time, sorted by TPRs.



(c) Tamperings during the night time, sorted by TPRs.

Figure 5.21: TPRs for different tamper events when using $\gamma_{FNR=0}$ as the threshold and without using time-wise filtering.

not receive beacons during the events 2 cm left at night, 4 cm up at night, and 60 cm left at night. These results are not shown in the figure.

Figure 5.21a shows TPRs for tamper events in order of their execution while Figure 5.21b and Figure 5.21c show them ordered by TPR night and day values. We can see from this that the magnitude of tampering (moved distance) does not correlate with detectability. It can also be seen that most tamper events can clearly be detected while

a minority are hard to detect.

The minimum TPR value during the day time is 0.0034, the maximum is 1, and the average is 0.704. During the night time, the minimum TPR value is 0, the maximum is 1, and the average is 0.691.

5.2.2.3 Discussion

In the previous sections we have described and analysed multi-receiver CSI tamper detection. Now we discuss how these methods can be put to work in practice to secure a Wi-Fi based WSN setup. We first discuss potential capabilities of such a system based on the previously shown evaluations. We then discuss how a practical Wi-Fi based system can incorporate the methods described and, finally, we describe how such system can be operated in a practical setting.

System Capability Many WSN systems are used to monitor and control critical infrastructure such as airports, refineries, hospitals or military installations. Additional layers of defence guarding the installation against tampering and other attacks are desirable. CSI tamper detection is a valuable building block in this context.

The evaluation has shown that in different deployment settings different FPR and TPR rates are achievable. The level of tolerable FPR and required TPR depends very much on the application scenario. The question is: in which application scenarios could the described system be employed?

A large number of WSN systems are deployed in areas with little variation in the environment. For example, systems are used to monitor and control production processes in refineries or power plants. Workers move throughout these installations but movement is limited; times of high activity are often known in advance (e.g. scheduled maintenance). In these settings a minimal FPR is required (ideally zero) as false alarms require costly investigation of the situation. On the other hand a high TPR is required to ensure that tampering with devices is detected. Our evaluation shows that requirements of such a setup can be fulfilled.

Other WSN setups are in relatively busy environments. For example, a wireless camera

system used to monitor an office building would experience high levels of movement. In such a setting a zero FPR can also be achieved but only when making some sacrifices on the achievable TPR. However, if CSI tamper detection is used as an additional layer of defence and not the only security mechanism a TPR below 1 may already bring significant added value to the overall security of the system.

From our evaluations we conclude that CSI tamper detection is feasible with (i) perfect FPR ($FPR = 0$) and perfect TPR ($TPR = 1$) in settings with limited movement; and (ii) perfect FPR ($FPR = 0$) and good TPR ($TPR = 0.53$) in settings with high levels of movement.

System Design A system using Wi-Fi enabled nodes would likely operate in an infrastructure mode. Devices such as cameras would transmit data via access points interconnected by a fixed wired backbone. It can be assumed that multiple access points are in communication range of a device. We therefore suggest extending the functionality of access points to collect CSI information. The collected CSI information is then forwarded to a central system which carries out CSI-based tamper detection. To enable CSI collection at multiple access points nodes must transmit broadcast frames. This can be achieved by having nodes transmit periodic beacons for the purpose of tamper detection (in our experiments a one second interval was used). Using regular data transmissions of the nodes is less suitable as 802.11n adjusts the number of spatial streams and CSI is dependent on the number of spatial streams (see 5.2.1). However, existing beaconing of nodes can be reused for the purpose of tamper detection. The system must allow secure collection of CSI data at access points which we believe is not trivial but achievable.

Deployment and Operation A system using multi-receiver CSI tamper detection requires a training phase when it is deployed. Depending on the application requirements, different thresholds might be used ($\max(D^\tau)$, γ_{EER} and $\gamma_{FNR=0}$) to achieve the desired FPR and TPR rates. The different thresholds have different complexity in terms of deployment and training.

Using $\max(D^\tau)$ requires a very short training phase. In our experiments we used

100 packets transmitted over a period of 100s. However, during the transmission of these packets it must be ensured that the system is in an untampered state and that the environment is quiet.

Using γ_{EER} and $\gamma_{FNR=0}$ requires training data which contains a tampered state. This can be achieved by treating the initial placement location of a node as a tampered reference state in which data is collected over a period of time. Thereafter the node is placed in its operation location which is the untampered state. During the tampered and untampered state training data are collected to calculate the thresholds as previously described.

We believe that the described setup procedure is feasible for many application scenarios. For example, when a security system is deployed in a restricted area it is possible to ensure no movement in this area during installation.

Once a system is deployed it may happen that the communication environment changes naturally and the initial selected thresholds are invalidated. For example, smaller building alterations would change the observed CSI. These situations can be identified by an increase in the FPR rate and a re-initialisation of the system is required.

5.3 Chapter Summary

We have shown that 802.11n CSI information measured at a receiver for each incoming packet can be used to create a tamper-evidence value which indicates potential tampering with the sender. In particular, our method is able to handle changing spatial stream configurations of 802.11n transmission. We have shown that tampering in the form of device movement or device replacement are clearly detectable. Detection of device replacement can be used for node identification. Furthermore, we have shown that alarms from tamper-unrelated environment changes, such as pedestrians, can be avoided by using multiple receivers. We provided more reliable tamper detection mechanism that can be used in practical deployments.

Although many WSN devices make use of 802.11n for communication, other transceiver types are also in use. The proposed methods can directly be applied to other commu-

nication systems based on OFDM. Other systems will describe the wireless channel differently. However, we believe that any information describing the communication channel is useful input for a tamper detection mechanism.

Chapter 6

Conclusion and Future Work

As Wireless Sensor Networks (WSNs) are being deployed in critical applications, their secure and reliable operation is of great importance. WSNs operate with scarce resources, so traditional security solutions cannot be directly adapted to WSNs. Existing solutions must be modified or new solutions must be proposed. This thesis has contributed to the development of new security mechanisms for WSNs on areas lacking much attention from the research community. It is important to optimise the usage of limited node resources. To this end, this thesis introduced frameworks that will improve the performance of security protocols on WSNs, and contributed methods for node identification without using costly cryptographic mechanisms.

6.1 Contributions

Chapter 3 presented Codo, a framework for efficient secure data storage in WSNs. Codo provides fast and energy-efficient data storage and retrieval, addressing the required level of protection. A design specification and a detailed implementation of Codo for the Contiki operating system running on a Tmote Sky node were included. Different aspects of Codo were also evaluated. The confidential data storage framework was then combined with secure communication, removing the duplication of security operations on the sensor node. This combined secure-storage and communication framework increases the performance and decreases the energy expenditure of the sensor node. The prototype

implementation showed that combined secure storage and communication can reduce security-related processing on nodes by up to 71% and energy consumption by up to 32.1%. A detailed definition of the framework for IP/6LoWPAN networks was presented in the chapter. An implementation of the framework for the Contiki operating system and a detailed evaluation of the performance gains were also given.

In Chapter 4, sensor node identification based on clock skew was explained. Clock skew calculation is done locally on the sensor node with the usage of local clocks. Local clock skew calculation can overcome the constant network delay required for remote clock skew calculation, which is problematic in duty-cycled WSNs. It has been shown that a sample period of $7.8125ms$ and a sample size of 200 are sufficient to determine clock skew locally with the same quality as a remote technique with a sample period of $4s$ and a sample size of 2500. A description of the method for local clock skew calculation of a node's crystal-based real-time clock using the high-precision clock available on modern transceivers was provided. The implementation of this method for Zolertia Z1 nodes using the Contiki operating system was also explained. Additionally, the dependency of clock skew calculation quality and clock sampling effort was analysed.

In Chapter 5, the CSI of a wireless channel was used for node identification and tamper detection. 802.11n networks use the OFDM modulation scheme. In OFDM, receivers estimate the channel conditions and send them to the transmitters. The estimation is called CSI, and provides rich information about the wireless channel. Any change in the wireless channel affects the CSI, and it has been shown that these effects can be used to identify nodes and to detect tampering with them. Any tampering on the wireless device will cause changes on CSI values. However, tamper-unrelated events, like movement of people in the communication environment, also lead to CSI fluctuations and cause false alarms. Analysing CSI values of incoming packets simultaneously on multiple receivers can help to distinguish tamper and movement events. It has been shown that the proposed system deployed in a busy office environment achieved 53% tamper detection while raising zero false alarms. Necessary algorithms for tamper detection using single and multiple receivers were described in the chapter. The tamper detection capability of

the proposed algorithm using realistic deployment environments were also analysed.

6.2 Threat Models and Limitations

In this section we discuss about the threat models and limitations of the proposed methods in each chapter.

Confidential Data Storage for Wireless Sensor Network In Chapter 3, we combined secure storage and communication for WSN. In this work, we used pre-deployed keys. Key management and in particular the usage of keys are dependent on the application scenario. The main aspect to consider is who will consume a data packet. In typical scenarios, a host requests data from a node that then generates a data packet, for example a temperature reading, protects it with a communication security protocol, and transmits. This work also target applications where nodes generate data, store it in the file system, and transmit data on request at a later stage. This stored data may be requested by a single host or by multiple hosts.

Data readings are stored in a file and a specific key is used for encryption (and potentially for authentication if used). In this case the key used to protect the file is shared with the remote hosts who consume this data. If the *same* data file is consumed by multiple hosts there is no need of confidentiality among these hosts for this data. In rare cases, if it is considered problematical to use one shared key among a group of hosts it is possible to store data readings of the same type in multiple files with individual keys. This allows us to issue individual keys to accessing hosts while still retaining the performance benefits in requesting pre-encrypted data. However, in this case data is stored multiple times which leads to a significant increase in storage requirements. This increase in storage requirements may not be problematical if nodes are equipped with enough flash storage space. If a node is compromised [ZYN08] it can be added in a blacklist maintained by an intrusion detection system [RWV13] and all keys held by this node must be invalidated.

Node Identification Based on Clock Skew In Chapter 4, we are considering an attacker who is trying to impersonate a sensor node. The attacker can do this by obtaining the key that used for identification and authentication. Our aim is to introduce an additional layer of protection, which we do by binding the identification process to the node hardware. In this work we assume the attacker does not have physical access to the sensor node, and he is unable to modify the software running on the device.

Initial experiments have shown that the measured clock skew depends on temperature. A node would need to be profiled in terms of skew over the expected temperature range. Skew values would have to be transmitted together with a temperature reading in order to allow identification in deployments with varying temperature. An attacker can use this temperature effect maliciously and prevent the proposed system from working properly by changing the temperature of the environment that the sensor operates in.

Tamper Detection and Node Identification Based on CSI In Chapter 5, we are considering an attacker who is capable of physically tampering with a device. We assume that the attacker is able to move or rotate the device. Our aim is to detect such device movements with a high detection reliability and a low false alarm rate.

In this work, we do not consider physical tampering with internal components of the device. We do not aim to detect replacement of device components such as the micro-controller. Additionally, we do not assume the attacker is able to change the software running on the device. To protect against internal device tampering, other protection methods than the one discussed in this work have to be used.

Depending on the selected threshold, this work might require a long training phase. Moreover, initial thresholds might be invalidated due to the natural changes in the communication environment. In this case, re-initialisation of the system is required.

6.3 Discussions

In this section we talk about the generalizability and the applicability of the works presented in the thesis. We also discuss the cost and scope of the proposed techniques.

Avoiding duplication of cryptographic operations on a sensor node is crucial as it has limited battery life and processing power. This thesis has shown in Chapter 3 that combining secure storage and communication solutions removes the redundant cryptographic operations, and dramatically reduces the security-related processing and energy usage. Considering security of a system as a whole rather than concentrating on individual parts helps to optimise the usage of the resources. The framework presented here uses only the IPsec protocol to secure the communication link, however, it would still be applicable to other protocols such as DTLS.

IPsec protocol is widely adapted to existing operating systems, and there is no extra cost on the host side to use the framework. Proposed method does not require extra hardware or any hardware modifications on sensor nodes. The framework only requires modification on sensor node operating system. We believe that the sensor node platforms can use the framework for many years, because both secure storage and secure communication are essential components of WSNs security.

The thesis has also shown that hardware and channel characteristics can be used for node identification and tamper detection. The methods proposed here do not require cryptographic operations and, therefore, require fewer resources.

Node identification method based on clock skew presented in Chapter 4 would work on any sensor node that has two clock sources. At least one of these clocks must be precise enough to get clock skew results. Fortunately, radio transceivers have precise clock sources and can be used for clock skew calculation. The method only requires small modification (a wiring) on the sensor node. Since the sensor nodes will have at least two clock sources (including a stable radio transceiver clock), we believe that the proposed method will work many years.

The tamper detection algorithm presented in Chapter 5 conforms with OFDM-based systems. Although other systems may use different modulation techniques and describe the channel differently, any information describing the communication channel can be used for a tamper detection mechanism. The method uses off-the-shelf 802.11n Wi-Fi cards without requiring any hardware modification. Additionally, existing beaconing packets

of Wi-Fi devices can be used to detect tamper events, which makes the deployment of the method into the existing systems easier. The tamper detection algorithm does not require extra hardware. The method can be used in Wi-Fi protocol as long as a version of 802.11 that adapts Orthogonal Frequency Division Multiplexing (OFDM) is used.

6.4 Future Work

In this section we recommend and discuss possible future work for each of the key contributions presented in the thesis.

Confidential Data Storage The combined secure storage and communication framework presented in Chapter 3 uses the IPsec protocol to secure end-to-end communication. DTLS is an alternative solution to IPsec. An instantiation of the combined framework for DTLS is described in [BRRV15], but its evaluation is not presented. An evaluation of the framework that quantifies performance enhancements and resource saving can be examined. Additionally, the framework uses pre-shared keys. Automatic key management support can be added, and its evaluation can be investigated.

The work in Chapter 3 has shown the importance of considering security of a system as a whole. We focused at secure storage and communication steps here. However, similar approaches can be investigated for different levels of security on WSNs.

Node Identification Based on Clock Skew Initial experiments in Chapter 4 have shown that the measured clock skew is affected by temperature. An investigation of the temperature effects can be carried out as future work. Additionally, more environmental effects (such as humidity) that alter the clock behaviour can be investigated.

In this work we used crystal-based real-time clock of a node and high-precision clock of a transceiver. Additional clock sources that have more precision than the one on the transceiver can be investigated to achieve more stable clock skew numbers.

Tamper Detection and Node Identification Based on CSI The tamper detection method proposed in Chapter 5 uses Wi-Fi networks. We showed its feasibility with a

prototype application. Its integration to existing Wi-Fi operation can be investigated. Additionally, integration of the technique with other existing communication protocols, especially ones that use OFDM, can be investigated.

We have shown in the thesis that hardware characteristics can be used for identification, which provide an additional layer of protection. Our methods use clock skew or wireless channel information, and they are implemented on sensor nodes. We believe that there are many more hardware characteristics that can be used for an additional layer of protection. These are not only limited to sensor nodes, they can be used on other platforms as well. For example, in our another work, we use quantum confinement to uniquely identify devices [RBZ+15]. This method is a lightweight solution for device identification and can be used on any device. Similarly, different sources and methods for additional layer of protection for different platforms can be investigated as future work.

Bibliography

[ABSK10] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the third ACM conference on Wireless network security*, WiSec '10, pages 169–174, New York, NY, USA, 2010. ACM. [Cited on page 20]

[AQR07] Ortal Arazi, Hairong Qi, and Derek Rose. A Public Key Cryptographic Method for Denial of Service Mitigation in Wireless Sensor Networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pages 51–59, June 2007. [Cited on page 17]

[Arc08] ArchRock Corporation. Phynet n4x series, 2008. [Cited on page 20]

[Atm12] Atmel. Atmel ATSHA204 datasheet, March 2012. `http://www.atmel.com/Images/Atmel-8740-CryptoAuth-ATSHA204-Datasheet.pdf`. [Cited on page 61]

[BBGO08] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. Wireless Device Identification with Radiometric Signatures. In *Proc. MobiCom'08*, 2008. [Cited on page 23]

[BBT11] Kemal Bicakci, Ibrahim Eethem Bagci, and Bulent Tavli. Lifetime Bounds of Wireless Sensor Networks Preserving Perfect Sink Unobservability. *Communications Letters, IEEE*, 15(2):205–207, February 2011. [Cited on page 18]

[BGTB11] Kemal Bicakci, Hakan Gultekin, Bulent Tavli, and Ibrahim Ethem Bagci. Maximizing lifetime of event-unobservable wireless sensor networks. *Computer Standards & Interfaces*, 33(4):401–410, 2011. [Cited on page 18]

[BJT12] Raghav Bhaskar, Ragesh Jaiswal, and Sidharth Telang. Congestion Lower Bounds for Secure In-network Aggregation. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 197–204, New York, NY, USA, 2012. ACM. [Cited on page 18]

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In *Proceedings of the 17th international*

conference on The Theory and Application of Cryptology and Information Security, pages 344–371. Springer-Verlag, 2011. [Cited on pages 50 and 59]

[BKR15] Martin Bor, Alex King, and Utz Roedig. Lifetime bounds of Wi-Fi Enabled Sensor Nodes. In *Proc. IUPT'15*, 2015. [Cited on page 76]

[BM07] Neerja Bhatnagar and Ethan L. Miller. Designing a secure reliable file system for sensor networks. In *Proceedings of the 2007 ACM workshop on Storage security and survivability*, StorageSS '07, pages 19–24, New York, NY, USA, 2007. ACM. [Cited on pages 18, 25, 26, and 27]

[BRRV15] Ibrahim Ethem Bagci, Shahid Raza, Utz Roedig, and Thiemo Voigt. Fusion: coalesced confidential storage and communication framework for the iot. *Security and Communication Networks*, pages n/a–n/a, 2015. [Cited on page 117]

[CAE+07] Nathan Cooprider, Will Archer, Eric Eide, David Gay, and John Regehr. Efficient Memory Safety for TinyOS. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, pages 205–218, New York, NY, USA, 2007. ACM. [Cited on page 18]

[Cer] CertiVox. MIRACL - Multiprecision Integer and Rational Arithmetic C/C++ Library. [Cited on pages 35 and 49]

[CFPS09] Claude Castelluccia, Aurélien Francillon, Daniele Perito, and Claudio Soriente. On the Difficulty of Software-based Attestation of Embedded Devices. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 400–409, New York, NY, USA, 2009. ACM. [Cited on page 18]

[CMYP09a] Xiangqian Chen, K. Makki, Kang Yen, and N. Pissinou. Sensor network security: a survey. *Communications Surveys Tutorials, IEEE*, 11(2):52 –73, quarter 2009. [Cited on page 18]

[CMYP09b] Xiangqian Chen, Kia Makki, Kang Yen, and N. Pissinou. Sensor network security: a survey. *Communications Surveys Tutorials, IEEE*, 11(2):52–73, Second 2009. [Cited on pages 17 and 18]

[CT09] Lander Casado and Philippas Tsigas. ContikiSec: A Secure Network Layer for Wireless Sensor Networks Under the Contiki Operating System. In *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*, NordSec '09, pages 133–147, Berlin, Heidelberg, 2009. Springer-Verlag. [Cited on page 18]

[CY05] Seyit A Camtepe and Bülent Yener. Key distribution mechanisms for wireless sensor networks: a survey. *Rensselaer Polytechnic Institute, Troy, New York, Technical Report*, pages 05–07, 2005. [Cited on page 17]

[CYS+10] Bogdan Carbunar, Yang Yu, Weidong Shi, Michael Pearce, and Venu Vasudevan. Query Privacy in Wireless Sensor Networks. *ACM Trans. Sen. Netw.*, 6(2):14:1–14:34, March 2010. [Cited on page 18]

[DC09] Boris Danev and Srdjan Capkun. Transient-based Identification of Wireless Sensor Nodes. In *Proc. IPSN'09*, 2009. [Cited on page 23]

[DGV04] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462. IEEE Computer Society, 2004. [Cited on pages 25, 29, and 49]

[DHBC09] Boris Danev, Thomas S Heydt-Benjamin, and Srdjan Capkun. Physical-layer Identification of RFID Devices. In *Proc. USENIX'09*, 2009. [Cited on page 23]

[Dun11] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science, December 2011. [Cited on page 66]

[DZC12] Boris Danev, Davide Zanetti, and Srdjan Capkun. On Physical-layer Identification of Wireless Devices. *ACM Comput. Surv.*, 45(1):6, 2012. [Cited on page 23]

[FC06] Daniel B. Faria and David R. Cheriton. Detecting identity-based attacks in wireless networks using signalprints. In *Proc. WiSe'06*, 2006. [Cited on page 23]

[FC08] Aurélien Francillon and Claude Castelluccia. Code Injection Attacks on Harvard-architecture Devices. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 15–26, New York, NY, USA, 2008. ACM. [Cited on page 18]

[FGS09] Christopher Ferguson, Qijun Gu, and Hongchi Shi. Self-healing Control Flow Protection in Sensor Applications. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 213–224, New York, NY, USA, 2009. ACM. [Cited on page 18]

[FMMA06] Sepideh Fouladgar, Bastien Mainaud, Khaled Masmoudi, and Hossam Afifi. Tiny 3-TLS: a trust delegation protocol for wireless sensor networks. In *Proceedings of the Third European conference on Security and Privacy in Ad-Hoc and Sensor Networks*, pages 32–42. Springer-Verlag, 2006. [Cited on page 20]

[GFN11] Qijun Gu, Christopher Ferguson, and Rizwan Noorani. A study of self-propagating mal-packets in sensor networks: Attacks and defenses. *Computers & Security*, 30(1):13 – 27, 2011. [Cited on page 18]

[GN08] Qijun Gu and Rizwan Noorani. Towards Self-propagate Mal-packets in Sensor Networks. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 172–182, New York, NY, USA, 2008. ACM. [Cited on page 18]

[GSM09] Matthias Gauger, Olga Saukh, and Pedro José Marrón. Enlighten me! secure key assignment in wireless sensor networks. In *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*, pages 246–255. IEEE, 2009. [Cited on page 17]

[GWMA07] Joao Girao, Dirk Westhoff, Einar Mykletun, and Toshinori Araki. TinyPEDS: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Netw.*, 5:1073–1089, September 2007. [Cited on pages 19 and 27]

[Han09] Morten Tranberg Hansen. Asynchronous Group Key Distribution on Top of the Cc2420 Security Mechanisms for Sensor Networks. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 13–20, New York, NY, USA, 2009. ACM. [Cited on page 17]

[HCSO09] Wen Hu, Peter Corke, Wen Chan Shih, and Leslie Overs. secFleck: A Public Key Technology Platform for Wireless Sensor Networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, EWSN '09, pages 296–311, 2009. [Cited on pages 17 and 26]

[HHSW11] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Tool Release: Gathering 802.11n Traces with Channel State Information. *ACM SIGCOMM CCR*, 41(1):53, 2011. [Cited on pages 78, 86, and 95]

[HKH+10] Sungmin Hong, Daeyoung Kim, Minkeun Ha, Sungho Bae, Sang Jun Park, Wooyoung Jung, and Jae-Eon Kim. SNAIL: an IP-based wireless sensor network approach to the internet of things. *Wireless Communications, IEEE*, 17(6):34 –42, december 2010. [Cited on page 20]

[HTW+08] Ding-Jie Huang, Wei-Chung Teng, Chih-Yuan Wang, Hsuan-Yu Huang, and J.M. Hellerstein. Clock Skew Based Node Identification in Wireless Sensor Networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5, 2008. [Cited on pages 21 and 66]

[HUW11] Isabelle Hang, Markus Ullmann, and Christian Wieschebrink. Short Paper: A New Identity-based DH Key-agreement Protocol for Wireless Sensor Networks Based on the Arazi-Qi Scheme. In *Proceedings of the Fourth ACM Conference on Wireless Network Security*, WiSec '11, pages 139–144, New York, NY, USA, 2011. ACM. [Cited on page 17]

[IEE03] IEEE std. 802.15.4 - 2003. Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs). *IEEE*, 2003. [Cited on page 19]

[ILMé10] Marian Kamal Iskander, Adam J. Lee, and Daniel Moss é. Privacy and Robustness for Data Aggregation in Wireless Sensor Networks. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 699–701, New York, NY, USA, 2010. ACM. [Cited on page 18]

[Iro] Ironkey. http://www.ironkey.com/. [Cited on page 19]

[JBMC10] Marcos A. Simplício Jr., Paulo S.L.M. Barreto, Cintia B. Margi, and Tereza C.M.B. Carvalho. A survey on key management mechanisms for distributed Wireless Sensor Networks. *Computer Networks*, 54(15):2591 – 2612, 2010. [Cited on page 17]

[JK08] Suman Jana and Sneha Kumar Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, MobiCom '08, pages 104–115, New York, NY, USA, 2008. ACM. [Cited on page 20]

[JZL+13] Zhiping Jiang, Jizhong Zhao, Xiang-Yang Li, Jinsong Han, and Wei Xi. Rejecting the Attack: Source Authentication for Wi-Fi Management Frames using CSI Information. In *Proc. INFOCOM'13*, 2013. [Cited on pages 22 and 77]

[KBC05] Tadayoshi Kohno, Andre Broido, and K.C. Claffy. Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on*, 2(2):93–108, 2005. [Cited on pages 20 and 62]

[KBG+09] Ioannis Krontiris, Zinaida Benenson, Thanassis Giannetsos, Felix C. Freiling, and Tassos Dimitriou. Cooperative Intrusion Detection in Wireless Sensor Networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, EWSN '09, pages 263–278, Berlin, Heidelberg, 2009. Springer-Verlag. [Cited on page 18]

[Ken05] Stephen Kent. IP Encapsulating Security Payload (ESP). RFC 4303, 2005. [Cited on page 44]

[KFLFS11] Silvija Kokalj-Filipović, Fabrice Le Fessant, and Predrag Spasojević. Trade-offs of source location protection in globally attacked sensor networks: A case analysis. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on*, pages 323–331, June 2011. [Cited on page 18]

[KHNE10] Charlie Kaufman, Paul Hoffman, Yoav Nir, and Pasi Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996, 2010. [Cited on page 29]

[KHS+11] Thomas Kothmayr, Wen Hu, Corinna Schmitt, Michael Bruenig, and Georg Carle. Poster: Securing the Internet of Things with DTLS. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 345–346, New York, NY, USA, 2011. ACM. [Cited on page 17]

[KKS07] Ram Kumar, Eddie Kohler, and Mani Srivastava. Harbor: Software-based Memory Protection for Sensor Nodes. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, IPSN '07, pages 340–349, New York, NY, USA, 2007. ACM. [Cited on page 18]

[KSW04] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*,

SenSys '04, pages 162–175, New York, NY, USA, 2004. ACM. [Cited on page 17]

[LMPG07] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: A Secure Sensor Network Communication Architecture. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, IPSN '07, pages 479–488, New York, NY, USA, 2007. ACM. [Cited on page 17]

[LN08] An Liu and Peng Ning. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 245–256, April 2008. [Cited on page 17]

[LON08] An Liu, Young-Hyun Oh, and Peng Ning. Secure and DoS-Resistant Code Dissemination in Wireless Sensor Networks Using Seluge. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, IPSN '08, pages 561–562, Washington, DC, USA, 2008. IEEE Computer Society. [Cited on page 18]

[LXMT06] Zang Li, Wenyuan Xu, Rob Miller, and Wade Trappe. Securing wireless systems via lower layer enforcements. In *Proc. WiSe'06*, 2006. [Cited on page 22]

[Mah36] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936. [Cited on page 82]

[MKHC07] Gabriel Montenegro, Nandakishore Kushalnagar, Jonathan Hui, and David Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, 2007. [Cited on page 41]

[MKSL04] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 39–49, New York, NY, USA, 2004. ACM. [Cited on pages 21 and 64]

[MSS10] Qi Mi, John A. Stankovic, and Radu Stoleru. Secure Walking GPS: A Secure Localization and Key Distribution Scheme for Wireless Sensor Networks. In *Proceedings of the Third ACM Conference on Wireless Network Security*, WiSec '10, pages 163–168, New York, NY, USA, 2010. ACM. [Cited on page 18]

[MST99] Sue B. Moon, Paul Skelly, and Don Towsley. Estimation and removal of clock skew from network delay measurements. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 227–234 vol.1, 1999. [Cited on page 64]

[Mur06] Steven J. Murdoch. Hot or not: revealing hidden services by their clock skew. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 27–36, New York, NY, USA, 2006. ACM. [Cited on page 21]

[NRLV08] Dennis K. Nilsson, Tanya Roosta, Ulf Lindqvist, and Alfonso Valdes. Key Management and Secure Software Updates in Wireless Process Control Environments. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 100–108, New York, NY, USA, 2008. ACM. [Cited on page 17]

[OM07] Melek Önen and Refik Molva. Secure Data Aggregation with Multiple Encryption. In *Proceedings of the 4th European Conference on Wireless Sensor Networks*, EWSN'07, pages 117–132, Berlin, Heidelberg, 2007. Springer-Verlag. [Cited on page 18]

[PK07] Neal Patwari and Sneha Kumar Kasera. Robust location distinction using temporal link signatures. In *Proc. MobiCom'07*, 2007. [Cited on pages 22 and 80]

[PMST08] Roberto Di Pietro, Di Ma, Claudio Soriente, and Gene Tsudik. POSH: Proactive co-Operative Self-Healing in Unattended Wireless Sensor Networks. In *Reliable Distributed Systems, 2008. SRDS '08. IEEE Symposium on*, pages 185 –194, oct. 2008. [Cited on pages 19 and 27]

[PWEV12] Paolo Pettinato, Niklas Wirström, Joakim Eriksson, and Thiemo Voigt. Multi-channel two-way time of flight sensor network ranging. In *Proceedings of the 9th European Conference on Wireless Sensor Networks*, EWSN'12, pages 163–178, Berlin, Heidelberg, 2012. Springer-Verlag. [Cited on page 69]

[RBZ+15] J. Roberts, I. E. Bagci, M. A. M. Zawawi, J. Sexton, N. Hulbert, Y. J. Noori, M. P. Young, C. S. Woodhead, M. Missous, M. A. Migliorato, U. Roedig, and R. J. Young. Using Quantum Confinement to Uniquely Identify Devices. *Scientific reports*, 5, 2015. [Cited on page 118]

[RDH+12] Shahid Raza, Simon Duquennoy, Joel Höglund, Utz Roedig, and Thiemo Voigt. Secure communication for the Internet of Things - a comparison of link-layer security and IPsec for 6LoWPAN. *Security and Communication Networks*, 2012. [Cited on pages 12, 20, 42, 45, and 49]

[RRZ08] Wei Ren, Yi Ren, and Hui Zhang. HybridS: A Scheme for Secure Distributed Data Storage in WSNs. In *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, volume 2, pages 318 –323, dec. 2008. [Cited on pages 19, 25, 26, and 27]

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000. [Cited on page 82]

[RWV13] Shahid Raza, Linus Wallgren, and Thiemo Voigt. SVELTE: Real-time Intrusion Detection in the Internet of Things. *Ad Hoc Networks*, 11(8), nov 2013. [Cited on page 114]

[RYLZ09] Kui Ren, Shucheng Yu, Wenjing Lou, and Yanchao Zhang. Multi-User Broadcast Authentication in Wireless Sensor Networks. *Vehicular Technology, IEEE Transactions on*, 58(8):4554–4564, Oct 2009. [Cited on page 18]

[SHCO08] Wen Chan Shih, Wen Hu, Peter Corke, and Leslie Overs. A Public Key Technology Platform for Wireless Sensor Networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 447–448, New York, NY, USA, 2008. ACM. [Cited on page 17]

[SHZ+09] Min Shao, Wenhui Hu, Sencun Zhu, Guohong Cao, S. Krishnamurth, and T. La Porta. Cross-layer Enhanced Source Location Privacy in Sensor Networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–9, June 2009. [Cited on page 18]

[SK05] Karen Seo and Stephen Kent. Security Architecture for the Internet Protocol. RFC 4301, 2005. [Cited on pages 41 and 44]

[SKSC09] Piotr Szczechowiak, Anton Kargl, Michael Scott, and Martin Collier. On the Application of Pairing Based Cryptography to Wireless Sensor Networks. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 1–12, New York, NY, USA, 2009. ACM. [Cited on page 17]

[Sky06] Tmote Sky. Datasheet, 2006. `http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf`. [Cited on page 58]

[SOS+08] Piotr Szczechowiak, Leonardo B. Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. NanoECC: testing the limits of elliptic curve cryptography in sensor networks. In *Proceedings of the 5th European conference on Wireless sensor networks*, EWSN'08, pages 305–320, 2008. [Cited on page 26]

[SSW+09] Cormac Sreenan, Jorge Sa Silva, Lars Wolf, Ruben Eiras, Thiemo Voigt, Utz Roedig, Vasos Vassiliou, and Gregor Hackenbroich. Performance control in wireless sensor networks: the ginseng project - [Global communications news letter]. *Communications Magazine*, 47(8), August 2009. [Cited on page 25]

[TD11] Nicolas Tsiftes and Adam Dunkels. A database in every sensor. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 316–332. ACM, 2011. [Cited on page 42]

[TDZV09] Nicolas Tsiftes, Adam Dunkels, He Zhitao, and Thiemo Voigt. Enabling large-scale storage in sensor networks with the Coffee file system. In *Proceedings*

of the 2009 International Conference on Information Processing in Sensor Networks, IPSN '09, pages 349–360, Washington, DC, USA, 2009. IEEE Computer Society. [Cited on pages 26, 29, and 49]

[Tex13] Texas Instruments. 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. C), March 2013. `http://www.ti.com/lit/ds/symlink/cc2420.pdf`. [Cited on page 69]

[TOZJ09] Hailun Tan, Diethelm Ostry, John Zic, and Sanjay Jha. A Confidential and DoS-resistant Multi-hop Code Dissemination Protocol for Wireless Sensor Networks. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 245–252, New York, NY, USA, 2009. ACM. [Cited on page 18]

[UC10] Md. Borhan Uddin and Claude Castelluccia. Toward clock skew based wireless sensor node services. In *Wireless Internet Conference (WICON), 2010 The 5th Annual ICST*, pages 1–9, 2010. [Cited on pages 21, 62, and 66]

[US07] Oktay Ureten and Nur Serinken. Wireless security through RF fingerprinting. *Electrical and Computer Engineering, Canadian Journal of*, 32(1):27–33, 2007. [Cited on page 23]

[WFA09] Yun Wang, Weihuang Fu, and D.P. Agrawal. Intrusion detection in Gaussian distributed Wireless Sensor Networks. In *Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on*, pages 313–321, Oct 2009. [Cited on page 18]

[XJ13] Jie Xiong and Kyle Jamieson. SecureArray: Improving Wifi Security with Fine-grained Physical-layer Information. In *Proc. MobiCom'13*, 2013. [Cited on page 23]

[YSZ+08] Yi Yang, Min Shao, Sencun Zhu, Bhuvan Urgaonkar, and Guohong Cao. Towards Event Source Unobservability with Minimum Network Traffic in Sensor Networks. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 77–88, New York, NY, USA, 2008. ACM. [Cited on page 18]

[Yu09] Haifeng Yu. Secure and Highly-available Aggregation Queries in Large-scale Sensor Networks via Set Sampling. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 1–12, Washington, DC, USA, 2009. IEEE Computer Society. [Cited on page 18]

[ZB11] Shanshan Zheng and John S. Baras. Trust-assisted anomaly detection and localization in wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on*, pages 386–394, June 2011. [Cited on page 18]

[ZCHX09] Yingpei Zeng, Jiannong Cao, Jue Hong, and Li Xie. Secure localization and location verification inwireless sensor networks. In *Mobile Adhoc and*

Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on, pages 864–869, Oct 2009. [Cited on page 18]

[ZDC11] Fan Zhang, Reiner Dojen, and Tom Coffey. Comparative performance and energy consumption analysis of different AES implementations on a wireless sensor network node. *International Journal of Sensor Networks*, 10(4):192–201, 2011. [Cited on page 58]

[ZFPK08] Junxing Zhang, Mohammad H. Firooz, Neal Patwari, and Sneha K. Kasera. Advancing Wireless Link Signatures for Location Distinction. In *Proc. MobiCom'08*, 2008. [Cited on pages 22, 77, and 80]

[ZM08] Sebastian Zander and Steven J. Murdoch. An improved clock-skew measurement technique for revealing hidden services. In *Proceedings of the 17th conference on Security symposium*, SS'08, pages 211–225, Berkeley, CA, USA, 2008. USENIX Association. [Cited on page 20]

[Zol10] Zolertia. Zolertia Z1 datasheet, March 2010. `http://zolertia.com/sites/default/files/Zolertia-Z1-Datasheet.pdf`. [Cited on page 67]

[ZYN08] Qing Zhang, Ting Yu, and Peng Ning. A framework for identifying compromised nodes in wireless sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 11(3):12, 2008. [Cited on page 114]