NTNU

Norwegian University of Science and Technology

Security of Internet of Things Protocol Stacks

Øystein Løvdal Andersen

Master of Science in Communication Technology

Submission date:	June 2016
Supervisor:	Colin Alexander Boyd, ITEM

Norwegian University of Science and Technology
Department of Telematics

Title: Security of Internet of Things Protocol Stacks

Student: Øystein Løvdal Andersen

Problem Description:

Research and development in the Internet of Things (IoT) is progressing fast. Today it is widely believed that there are too many standards but it is not yet clear which ones will win out in the marketplace. Since security is one of the main IoT challenges, an important consideration is which protocol stack provides best security and privacy services. Security can be provided at different levels so it is not simple to decide the optimal choice.

The Internet of Things covers several different domains and technologies, introducing challenges regarding interoperability between different stacks, and implementation of standards on low powered and low energy devices. All of this combined creates new challenges in security and questions regarding how to ensure confidentiality, integrity and availability. Apple HomeKit, Samsung Smart, Thread, ZigBee and IETFs suggested protocol stack are just some of the proposed frameworks and protocol stacks today, with many more upcoming and challenging the market with their own solutions.

The Goal of this master thesis is:

- Identify security requirements introduced in IoT
- Compare and review established protocol stacks in IoT based on privacy, confidentiality, integrity and availability to determine advantages and disadvantages of different protocol stacks
- Suggest guidelines of which protocol stacks to use based on different security requirements, technologies and/or domains

Responsible Professor: Colin Alexander Boyd

Supervisor: Britta Hale

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) as the concluding part of my Master of Science in the Communication Technology program at the Department of Telematics (ITEM), and was carried out during the spring semester of 2016.

Trondheim, 12.06.2016

Øystein Løvdal Andersen

Acknowledgment

I would like to thank my responsible professor Colin Alexander Boyd and supervisor Britta Hale for their guidance and feedback during the course of this project.

I would also like to thank Thomas Ulleberg at Wireless Trondheim for introducing me to Internet of Things, and answering my questions throughout this period.

Abstract

Internet of Things has become one of the big buzzwords in the IT market in recent years, and it is predicted to continue its rapid growth in the coming years. In order to talk about the Internet of Things, this thesis presents an introduction to Internet of Things, what it is, how it surrounds us, and why it is so important to provide security to Internet of Things devices. A 4-layered protocol stack is proposed to work towards a common development framework for Internet of Things. Due to the limitations in power, bandwidth and processing power of devices, many of the established technologies and solutions we have today is simply not compatible with the requirements brought along by the Internet of Things. Wearables, smart homes and the Industrial Internet of Things are just some examples of what Internet of Things is being used for, and together with the use of previous research findings, it is shown how the different use-case areas bring different security requirements to developers.

Standards such as ZigBee, Thread, Z-Wave, Bluetooth Low Energy, and WirelessHART are some examples of established standards trying to win out in the marketplace. Often, these standards serve specific use-case areas, and thus, a new standard is proposed. IP-Smart is based on open and well-known protocols and is intended to cover several use-case areas. Comparison of the different standards shows that the application layer is sometimes left open for developers (Thread, BLE, IP-Smart) to carry out, how weaknesses is found in standards proposing their own cryptographic algorithms (ZigBee, Z-Wave, and WirelessHART), how Thread, IP-Smart and (if properly configured) BLE fulfills security off wearables, how standards require proper implementations to fulfill smart home requirements, and WirelessHART being the only standard which fulfills the additional performance requirements found in the Industrial Internet of Things. While many of the standards offer satisfactory security properties, the actual implementation is sometimes left to the developers to ensure secure products. An investigation into the two application layer protocols MQTT and CoAP indicates how CoAP with its use of DTLS provides a reasonable option to MQTT if extra reliability in lossy networks is of importance for the developers.

Sammendrag

Tingenes Internet har blitt en av de store buzzordene i IT markedet i de senere årene, og er forventet å fortsette sin voldsomme vekst i årene som kommer. For å kunne snakke om Tingenes Internett, vil denne avhandlingen presentere en introduksjon til Tingenes Internett, hva det er, hvordan det omringer oss, og hvorfor det er så viktig å sørge for sikkerhet i Tingenes Internet enheter. En 4-lags protokoll stakk er foreslått for å jobbe mot et felles utviklings rammeverk for Tingenes Internett. På grunn av begrensninger i strøm, båndbredde og prosessorkraft, er mange av de etablerte teknologier og løsninger vi har i dag simpelthen ikke kompatible med kravene som kommer av Tingenes Internett. Wearables, Smart Hjem og det Industrielle Tingenes Internett er kun noen eksempler på hva Tingenes Internett blir brukt til, og sammen med tidligere forskningsresultater, er det vist hvordan forskjellige bruksområder bringer forskjellige sikkerhetskrav til utviklerene.

Standarder som ZigBee, Thread, Z-Wave, Bluetooth Low Energy og WirelessHART er noen eksempler på etablerte standarder som prøver å vinne frem i markedet. Ofte vil disse standardene betjene spesifikke bruksområder, og derfor er en ny standard foreslått. IP-Smart er basert på åpne og velkjente protokoller og er tiltenkt å dekke flere bruksområder. Sammenligning av de forskjellige standardene viser at applikasjonslaget noen ganger er etterlatt åpent for utviklere (Thread, BLE, IP-Smart) til å innfri, hvordan sårbarheter er funnet i standarder som foreslår deres egne kryptografiske algoritmer (ZigBee, Z-Wave, og WirelessHART), hvordan Thread, IP-Smart, og (hvis korrekt implementert) BLE oppfyller sikkert for wearables, hvordan standarder krever korrekt implementasjon for å oppfylle Smart Hjem krav, og WirelessHART er den eneste standarden som oppfyller de ytterligere kvalitetskravene funnet i the Industrielle Tingenes Internett. Mens mange av standardene tilbyr tilfredstillende sikkerhetsegenskaper, er selve implementeringen noen ganger overlatt til utviklerene for å garantere sikre produkter. Gransking av de to applikasjonslager protokollene MQTT og CoAP indikerer hvordan CoAP med dens bruk av DTLS gir et fornuftig alternativ til MQTT hvis ekstra pålitelighet i tapsfulle nettverk er av viktighet for utviklerene.

Contents

List of Figures

List of Tables

1 | Introduction

In recent years, the Internet of Things has become one of the prominent buzzwords in the IT world. Development is progressing at a rapid pace, and there are several different standards trying to win out in the marketplace. Wearables, home appliances, building automation systems and much more are being connected to the Internet, and well-established standards such as ZigBee and Bluetooth are competing with newcomers such as Thread. Connecting devices comes with a promise of making our everyday life easier by monitoring and controlling our health, our homes, industrial systems, and so on. However, with an increasing responsibility transferred from the user to the devices, a new set of challenges emerges.

Security is one of the main challenges, but little is known about which standard offers the best security services. IoT offers a new set of security challenges, and optimal choice of standard might also depend on the area of use. Wearable devices are put in charge of handling highly personal sensitive data, while smart home devices can provide surveillance and access control systems to one's home. To get a better view of the world of the Internet of Things, this thesis will look to investigate some of the standards available for the IoT and compare them in terms of security services provided.

1.1 Objectives

In order to compare different IoT standards available, the following objectives were set for this thesis.

- Identify security requirements introduced in the Internet of Things.
- Compare and review established protocol stacks in IoT based on privacy, confidentiality,

integrity, and availability to determine advantages and disadvantages of different protocol stacks.

- Suggest guidelines about which protocol stacks to use based on different security requirements, technologies and/or domains.

1.2 Limitations

Availability of information about the different protocol stacks chosen in this thesis is varying. Some of the protocol stacks are proprietary solutions, offering no official specification manuals, while others are open with more information available. Comparisons of the different protocols stacks reflects this, and are done to the best of the author's abilities.

1.3 Approach

In the first part of this thesis, a literature study is done to present a definition of IoT, and specific requirements introduced in IoT. Different use-case areas of IoT devices are identified, and a selection of standards for this thesis is done.

The next part of this thesis provides a comparison and review of the different protocol stacks in terms of security and suitability with the different use-cases. Security requirements identified in the first part of the thesis are used as the baseline for the comparisons.

The Last part of the thesis is presenting practical work done to perform simulations of the two protocols CoAP and MQTT. Results from comparisons and simulations were then used to propose a guideline for best practice of the different standards in this thesis.

1.4 Structure of the Report

In this thesis there are 8 chapters and 2 appendices. A short description of the chapters follows.

Chapter 2 starts out with a presentation of the term Internet of Things, and a suggested definition of the term for this thesis. Following this is a proposed common protocol stack model for IoT, based on the OSI model. Specific IoT requirements as interoperability and scalability, low power/processing/bandwidth and security are then presented. Different use-case areas and their main security requirements are identified, to be used later in the thesis for comparison of the different protocol stacks.

Chapter 3 presents enabling technologies for IoT. Concepts as M2M and Wireless Sensor Networks starts off the chapter, followed by a short introduction of the radio technologies 802.15.4, Bluetooth and HART. Lastly, an introduction to important protocols such as IPv6, CoAP, MQTT, Elliptic Curve Diffie-Hellman, Elliptic Curce Juggling-Password Authenticated Key Exchange and TLS/DTLS is provided.

Chapter 4 introduces the 6 different standards ZigBee, Thread, Z-Wave, Bluetooth Low Energy, WirelessHART and the authors proposed IP-Smart with key technical details. Followed by a mapping of the standards own defined protocol stacks into the the authors proposed model.

Chapter 5 is the comparison and review chapter. Providing comparisons of the different protocol stacks based on the identified security requirements of IoT, as well as the use-cases presented previously.

Chapter 6 contains a introduction of Contiki OS and the Cooja simulator, and the results of the simulations of CoAP and MQTT.

Chapter 7 makes use of the results from the comparison and simulations to provide some guidelines for best practices for the different standards and further development of the IoT. It also contains a brief look ahead of what is to come in IoT, followed by the concluding remarks of this thesis.

In the appendices, a list of acronyms and additional information on the simulations performed is provided.

2 | IoT and Security Requirements

In this chapter, an introduction to the concept of the *Internet of Things* (IoT) is presented. IoT has several different interpretations depending on who you ask and therefore the writer has decided to put into own words a definition of IoT for this thesis and the questions raised towards it.

Section 2.2 presents the authors proposed model for an IoT protocol stack derived from a combination of the OSI model, and different specified protocol stacks already found in IoT. Following this, two sections focusing on IoT-specific requirements in terms of technologic constraints and specific security requirements are introduced. Lastly, a section introducing some possible use-case scenarios of IoT and highlighted important security requirements of each use-case are presented.

2.1 What is the Internet of Things?

One of the big buzz words in the IT market today is the ***Internet of Things*** or ***IoT*** for short. It has entered our homes, our cars, cities, industry, our bodies and much more. Every little device you can think of is getting Internet access and thus promoted as a “Smart Device”, capable of offering great new possibilities and making our lives less complicated. An exact definition of what IoT is, cannot be found, but many have tried to put into words what the IoT is. Internet of Things came to life as a term in 1999 in a presentation by Ashton [28], covering the idea of connecting RFID tags to the Internet. Since then, different parties has given their own definition of the term as shown by the quotes below.

We see the IoT as billions of smart, connected "things" (a sort of "universal global neural network" in the cloud) that will encompass every aspect of our lives, and its foundation is the intelligence that embedded processing provides.
Freescale & ARM*[4]*

The Internet of Things (IoT) refers to the use of intelligently connected devices and systems to leverage data gathered by embedded sensors and actuators in machines and other physical objects. IoT is expected to spread rapidly over the coming years and this convergence will unleash a new dimension of services that improve the quality of life of consumers and productivity of enterprises, unlocking an opportunity that the GSMA refers to as the 'Connected Life'.
GSMA *[17]*

ARM and GSMA's definition of IoT exemplifies just some of the various definitions of the IoT, illustrating just how easy it is to be confused about what exactly the IoT is. From the beginning with Ashton's use of the term in 1999, it has evolved along with the new technological innovations and is no longer a specific case of connecting RFID tags to the Internet. Instead, it includes this, as well as several other technologies which connect embedded devices to each other and the Internet. Showing an evolution where gathering and processing data is a task moved away from the users, and onto devices we surround us with instead.

2.2 The IoT Protocol Stack

Little work has been done in terms of working towards a standardized protocol stack in the IoT. Previous work [39, 1] has shown some efforts to propose a standardized protocol stack, but looking at the products available in the market today shows a wide array of different approaches (see Chapter 4). One possible reason for a lack of a standardized protocol stack could stem from the different technologies vendors build their products on. Wireless technologies as Bluetooth, RFID, ZigBee, and 802.15.4 represents just some of the available technologies, all with their own specifications on how to be configured and used. To be able to talk about different protocol stacks in this thesis, the author proposes a model for an IoT protocol stack model as seen in figure 2.1 to serve as a guideline in a similar fashion as the OSI model[1] is used for communication systems. The model will also be used in Chapter 4 to map the chosen protocol stacks into a single protocol stack model, to provide a better understanding how the protocol stacks compare to each other.

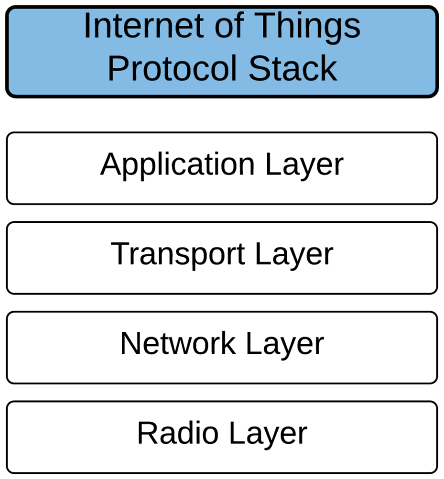


Figure 2.1: Proposed model for an IoT protocol stack

In this model, the *Radio Layer* is a unified layer of the Physical and MAC layers of the OSI model. Those two layers are closely connected in IoT, and are decided based on the radio technology in use (for instance 802.15.4 or Bluetooth). *Network* and *Transport* layers are kept the same as found in the OSI model, while the *Application Layer* serves as a unification of the Session, Presentation, and Application layers. As will be shown in Chapter 4, the Application Layer is sometimes kept open for OEMs to implement their own applications for the given product.

2.3 IoT Specific Requirements

With all the new devices being connected to the Internet with IoT, a set of new challenges and requirements arise. The new devices connected to the Internet is not necessarily equal in power and capacity as a PC, smartphone, tablet etc. Instead many of the new devices are limited in

[1]OSI model defined and explained: `https://support.microsoft.com/en-us/kb/103884`

their capabilities and created to serve a single purpose, a temperature sensor, a light bulb, pacemaker, car keys and the list goes on. Embedded devices are limited in what they can do by themselves, and with several different manufacturers and OEMs, the problem of how everything is supposed to connect together is also introduced. All showing that the IoT revolution is creating new challenges which needs to be addressed. The Internet Society, amongst many others as well[2] [2], acknowledges these challenges and the potential consequences of not addressing them properly [23].

2.3.1 Interoperability and Scalability

Interoperability and scalability will serve as two of the ground pillars if we are to exploit the full potential residing in the IoT. Expected growth projections of IoT devices in the coming years extend anything we have ever seen before[3][4], and serves as proof of the importance of solving interoperability and scalability issues in IoT. These challenges are not just of technical nature but also economic, QoS, user friendliness and much more [23], and solving these challenges could potentially generate great financial growth[5] in the future.

2.3.2 Low Power/Low Processing/Bandwidth

When we envision IoT devices, we often talk about small embedded devices created to serve a single purpose. Such as sensor devices (temperature, light, industrial, etc), wearables (fitness trackers, watches, health monitoring equipment, etc.), and other applicable devices. A common factor between these devices are size, and limitations in the battery, processing power, bandwidth capabilities, and the cost to realize the device. Thus, development and innovation for IoT must account for the constrained devices.

[2] Cisco: http://blogs.cisco.com/digital/the-internet-of-things-what-does-it-take-to-make-the-internet-of-everything-real

[3] Business Insider IoT projections: http://www.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10?r=US&IR=T&IR=T

[4] Gartner IoT projections: http://www.gartner.com/newsroom/id/3165317

[5] McKinnsey artcile: http://www.mckinsey.com/business-functions/business-technology/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world

2.4 Security

With the growth of new connected devices now and in the future, a whole new set of attack vectors for malicious tampering opens up. Attacks against smart meters/grids, unauthorized access of medical devices as pacemakers, physical tampering of robotic/industrial systems, is just some of the possible scenarios [30], if security is not properly implemented in the IoT. These attacks may ultimately have lethal consequences and proof of attacks have already emerged[6][7] shedding light on how, for instance, pacemakers potentially can be hacked and abused. Another problem encountered with the implementation of security in IoT is how established protocols has been created with more powerful devices in mind. Leading to the challenge of established protocol being incompatible with IoT devices. Thus, to offer the same QoS and security as provided by well-established protocols, new protocols has been and must be developed further to comply with the constrained IoT devices.

In order to investigate and discuss security of IoT protocol stacks in this thesis, a list of IoT specific security requirements, presented below, has been derived by previous work[30, 47, 16, 11] and the authors opinion of important security challenges in the IoT.

- **Access Control:** several devices is often part of a network in the IoT, and thus it is important to maintain some sort of access control method to provide a set of differentiating which devices have access to the network and their rights within the network.

- **Authentication:** with the set of new devices introduced in IoT, malicious parties might masquerade as authorized devices and alter data, providing false data. Depending on the data, this could prove highly critical, and the need for proper authentication of devices and data to neglect this problem is needed.

- **Availability:** IoT devices will not maintain a persistent connection to the network, to preserve energy. Devices entering sleep mode could prove a challenge when updating security parameters in the network and to alert nodes if a breach has been detected.

[6]Article from Wired: https://www.wired.com/2016/03/go-ahead-hackers-break-heart/
[7]Article from Forbes: http://www.forbes.com/sites/aarontilley/2015/03/06/nest-thermostat-hack-home-network/#f4302e95cb07

- **Confidentiality and Privacy:** Many areas of IoT will handle sensitive information and personal information. Information will be sent from wearables to users smartphones, from pasients to doctors, and so on. Strong cryptography will be an essential security property of IoT to ensure the protection of the vast amount of data being sent everywhere in the world of IoT.

- **Device Management:** in case of IoT devices being compromised, the network must be able to revoke certificates/authentication of a device to shut it out of the network. The network must ensure monitoring and discovery of compromised devices with minimal delay, to prevent further breach of the network and system.

- **Integrity:** message integrity is necessary to prevent tampering of messages. An attacker tampering with messages of medical devices, door locks, industrial machines etc. could lead to critical consequences

- **Multi Layer Security:** securing all the layers of the protocol stack will be important in the IoT because of new attack vectors. Physical tampering, sniffing attacks and malicious code attacks are examples of some of the different ways of attacking a device. The use of different protocols could also lead to issues with data fragmentation across the layers, showing some of the issues at hand if security is not made to work across all layers of the protocol stack.

2.5 IoT Use-Case Scenarios

In this section an introduction to 3 different use-case scenarios in the IoT is presented. Each use-case will be introduced with a closer look at possible benefits IoT provides, and the potential security issues found in the use-case. From this, a list of main security requirements of the use-case are presented, intended to serve as a guideline in Chapter 5 when comparing the different protocol stacks in this thesis. Following use-cases has been chosen for this thesis: wearables, home automation and the *Industrial Internet of Things* (IIoT).

2.5.1 Wearables

In recent years, the marked for wearables has grown rapidly[8]. Smartphones have brought endless possibilities and paved the way for activity trackers, smartwatches, medical devices, and so on. With the introduction of Internet connectivity to these devices, handling medical information of persons, several privacy and security concerns arise. Modification of for instance pacemakers through the Internet or other forms of wireless protocols could potentially open the device up to malicious use if not secured properly. Halperin et. al [18] highlighted this problem already back in 2008, and news articles from earlier this year[9] show us that this is still a highly relevant problem. All these devices collecting personal data about our health and well-being, and in some cases keeping people alive, will in all likelihood continue to be a central part of people's everyday life. Therefore, the use-case of wearable technologies, and in particular with a special focus on medical devices, is identified as one of the important challenges of the IoT to solve in terms of privacy and security.

Figure 2.2: Illustration of wearables[10]

[8]Article from Business Insider: http://www.businessinsider.com/the-wearable-computing-market-report-2014-10?r=US&IR=T&IR=T

[9]Wired: *Go ahead, hackers. Break my heart*, by Marie Moe. URL: https://www.wired.com/2016/03/go-ahead-hackers-break-heart/

[10]Source: https://tctechcrunch2011.files.wordpress.com/2015/06/wearables-e1455299947895.jpg?w=738

Use-Case Specific Security Requirements

NIST has released draft SP 1800-1[37], as an effort to highlight how to solve the stated problem of securing medical records on mobile devices. They highlight a set of security characteristics especially important for medical devices to ensure no loss of personal sensitive data and selective restriction of access to a device. Based on this, and previous work on the subject [3], the most important security characteristics of wearables/medical devices are listed as below:

- Access Control
- Device Integrity
- Person/Entity Authorization
- Transmission Security

2.5.2 Home automation

One of the more prominent scenarios for IoT has been to create the *Smart Home*. Solutions to simplify, and streamline different parts of the home such as HVAC (heating, ventilation, air conditioning), lighting, audio-visual, security systems (video surveillance, alarm systems, etc.) and so on is becoming more and more common among households today. The benefits of the smart home are not hard to see, with cost-saving and simplicity for the homeowner as the central selling points. However, with all the new possible solutions presented in the smart home, security issues arises with the different technological products.

Use-Case Specific Security Requirements

Jose et al. discuss this issue [25] looking at security challenges from different points of view. A smart home becomes an attractive target for an attacker with personal information and audio/video of a home environment transferred through the network. Other factors as different manufacturers of devices and possible lack of updates/patches could present different vulnerabilities to exploit. From the home user point of view, it must be assumed that not everyone will

[11]Source: http://searchsaltlake.com/wp-content/uploads/2016/01/Smart-Home-graphic.jpg

Figure 2.3: Illustration of the smart home[11]

be a tech savvy user and security may not be the main consideration when adding new devices the smart home. For the security engineers, this creates a difficult scenario of how to create a secure and easy to use product which maintains interconnectivity with the rest of the smart home.

Previous work on the topic by Jacobsson et al. [24] and Denning et al. [9] are just some examples of work done to identify main risks introduced in the smart home. The main security focuses identified by earlier work on the topic and in terms of this use case scenario are presented as listed below:

- Data Privacy
- Data Authenticity
- Device/User Authentication

2.5.3 Industrial Internet of Things - Smart Energy

The next milestone of the industrial revolution is dubbed as Industry 4.0. It represents the change towards the *Smart Factories* and the introduction of IoT into industrial control systems (ICS), creating the IIoT. Introduction of low-cost sensors, embedded devices etc. into manufacturing systems enables vendors to make more advanced systems and collect more data to streamline and improve the efficiency, and thus reduce cost. What differs IIoT from the other use-cases is an added focus on performance of the systems. ICS are dependent on real-time information sharing, availability and flexibility amongst others, in order to detect faults and errors in the system. A standard which does not fulfil the performance requirements could lead to economical consequences (manufacturing line stops producing without notice or data leakage of business sensitive data) or in worst case personal injury (for instance oil and gas systems not reporting critical information back in real-time could lead to critical accidents).

Use-Case Specific Security Requirements

To ensure the security of the IIoT, the research community has started to turn its attention towards ICS. Traditionally, such systems have been proprietary and had little focus on designing systems protected against dedicated attacks [26]. With the shift of focus, research has been mapping security challenges of the IIoT [54, 44] to identify important security requirements. Listed below are the requirements highlighted by previous research, which will be used later in this report for this use-case scenario:

- Access Control
- Availability
- Real-time Information Sharing providing Confidentiality and integrity
- Device Management

[12]Source: http://moneytechsearch.com/wp-content/uploads/2015/05/Internet-of-things-lg.jpg

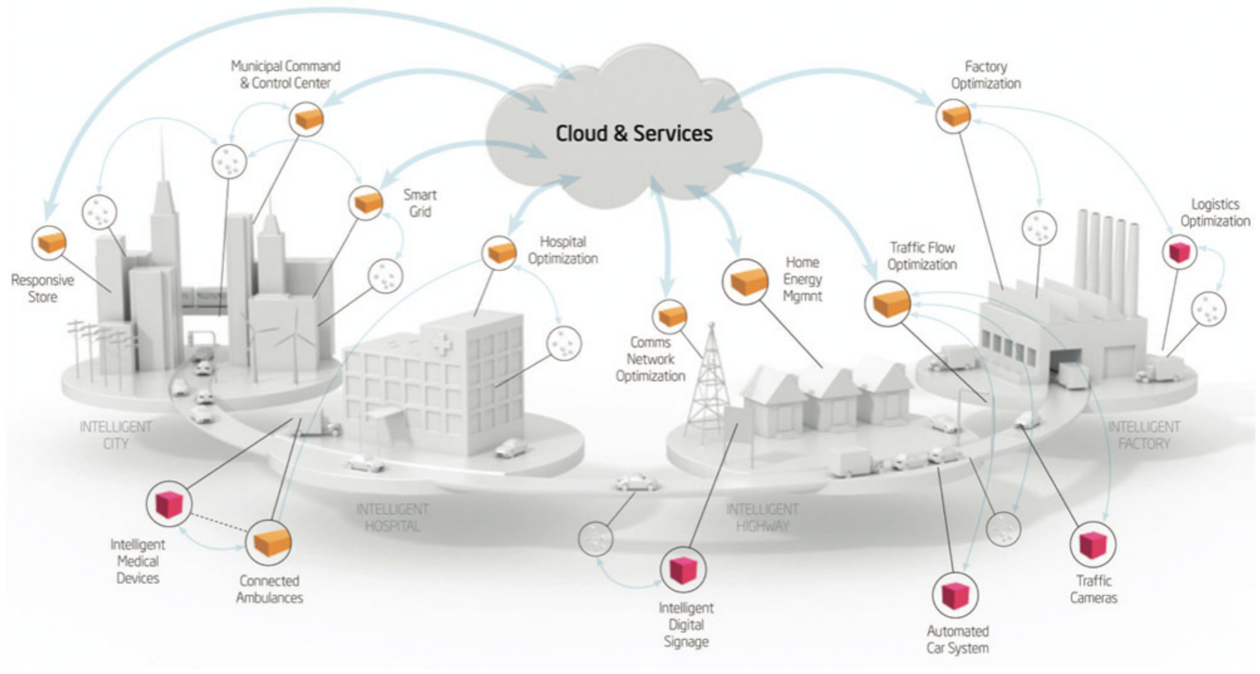


Figure 2.4: Illustration of the Industrial Internet of Things[12]

2.6 Summary

There are several different definitions of what the *Internet of Things* is, as shown with the quotes from ARM and GSMA. In common is the idea of connecting devices we surround us with, to exploit the possibilities it brings along. In order to work towards a common understanding of IoT, a 4-layered protocol stack model was proposed to be used in a similar way as the OSI model has been used for communication systems. IoT requires special requirements in order to be realized, as the constrained devices in use is incompatible with many of today's established standards. Different use-case areas require different focus in terms of security as well, and previous research and news have put light on the challenges and possible consequences by not solving these issues. New technologies and protocols has been developed in order to meet the challenges brought along by IoT.

3 | Enabling Technologies

In this chapter, introductions to enabling technologies of IoT are presented. In order to achieve a better understanding of how the IoT is realized, several technologies and protocols are presented in short to provide the need-to-know. Section 3.1 and 3.2 introduces the term M2M and the concept of Wireless Sensor Networks, which both are important concepts of enabling the IoT.

Section 3.3, 3.4 and 3.5 presents some of the underlying radio technologies central to the IoT. IEEE 802.15.4 is a wireless technology developed to offer a low-cost communication network for constrained devices. Bluetooth and HART are well-established communication technologies which are the backbone of Bluetooth Low Energy and WirelessHART, which will be presented in Chapter 4.

Section 3.6 - 3.11 introduces the protocols: IPv6, CoAP, MQTT, Elliptic Curve Diffie-Hellman, Elliptic Curve Juggling-Password Authenticated Key Exchange, and TLS/DTLS. These protocols serve as important protocols to enable and secure IoT devices.

3.1 M2M

M2M is a term often referred to machine-to-machine communication, but it could also refer to mobile-to-machine, machine-to-mobile, man-to-machine etc. [10]. It is used as a general term when talking about two machines communicating with each other, mostly without human interaction, through a wired or wireless communication channel. M2M as a term has evolved from the early times of telephony systems with services as caller ID and towards the rapidly growing device market and its services and applications. In more modern times the term is more closely connected to the Internet of Things and is widely used in industrial applications, sensor networks, wearables etc. Whereas the PC market focused on making people more productive and offering powerful high-end computing and communication platforms, the M2M market will be focused on services and applications devices can offer with minimal human intervention [13]. As IoT continues to develop and grow, so does its dependence on M2M and the possibilities it brings to the end users.

3.2 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) is a central backbone structure of the IoT. It is the network structure of many tiny devices capable of sensing, computing and communicating data. Sensors can monitor environmental or physical conditions in several different domains (air, water, soil, wind, structural data, industrial data, etc.) and communicate its data back to main nodes for further applications. A WSN is often organized in one of three possible network topologies: star, cluster tree, or mesh [33]. In all three topologies, a gateway node is the central point which connects the sensors to the rest of the network infrastructure. The chosen standard for the WSN (802.15.4, 802.11 or proprietary radio) is determined based on the use-case of the network.

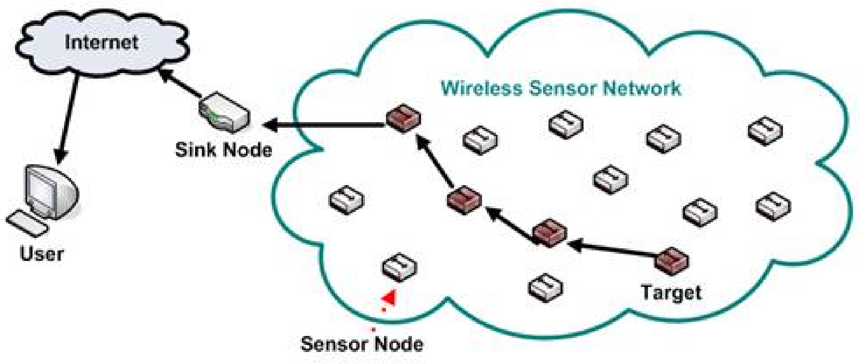


Figure 3.1: Illustration of a Wireless Sensor Network[1]

3.3 802.15.4

IEEE has created 802.15.4 as a standard which specifies the Physical and MAC layer for a Low Rate Wireless Personal Area Network (LR-WPAN). 802.15.4 offers a low-cost communication net-

[1]Source: http://monet.postech.ac.kr/images/introduction/image007.jpg

work for low powered/low speed devices, and ensures simple and easy installation, reliable data transfer, low-cost and reasonable battery life. Some of the important specifications of 802.15.4 can be seen in Table 3.1

	Frequency	**Channels**	**Data Rate**	**Channel Access Method**
802.15.4	2.4GHz	16	250 kbits/s	CSMA/CA

Table 3.1: Overview 802.15.4 specifications

3.3.1 Components

There are two different device types which can participate in an 802.15.4 network, *Full-function Device* (FFD) and *Reduced-function Device (RFD)*. A FFD has the capabilities to serve as a PAN coordinator or coordinator in the network. A PAN coordinator is the main coordinator which is in charge of the whole network while regular coordinators implement communication with devices and relay messages through the network. RFDs, on the other hand, are simple devices which have no need to send large amounts of data and can thereby be implemented using minimal resources and memory, an example of such a device can be a light switch or simple sensors.

3.3.2 Topologies

802.15.4 has two different network topologies defined: star networks or peer-to-peer networks. As illustrated in Figure 3.2, the Star topology has a single PAN coordinator, handling the communication between all the devices on the network. A P2P topology allows for more complex network structures/formations, and every device does not have to be connected to the PAN coordinator. In such case, regular coordinators (FFDs) can serve as masters for RFDs and relay messages to the PAN coordinator.

3.3.3 Functional Overview

In this subsection, a short description of different core components of 802.15.4 follows, as specified in the 802.15.4 standard [20].

[2]Source: http://zeitgeistlab.ca/doc/doc_images/pans.jpg

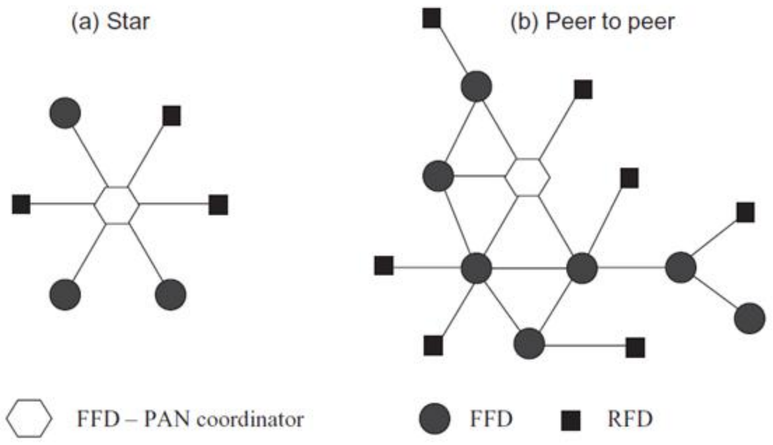


Figure 3.2: Illustration of 802.15.4 network topology[2]

Frame Structure

A MAC frame in the MAC layer of 802.15.4 is divided into 4 frames. A beacon frame used by the coordinator to transmit beacons, a data frame for transfer of data, an acknowledgement frame used for confirming successful reception of a frame and a MAC command frame used for handling all MAC peer entity control transfers.

Data Transfer

There are 3 different data transfer transactions defined in 802.15.4. There is data transfer to a coordinator where the device is transmitting data, data transfer from a coordinator where the device is receiving data and data transfer between two peer devices. The last one is only used in a P2P topology as data transfer in a Star topology must include a coordinator. All 3 data transfer methods are dependent on a beacon for network discovery. Data transfer can be done without the beacon enabled as well, if there are no dependencies on synchronization or low latency devices.

Improving probability of successful data delivery

802.15.4 employs several different mechanisms to improve the probability of successful data delivery between devices. Some of the important ones are listed below.

- Slotted CSMA-CA in beacon-enabled networks, using aligned backoff periods on all devices according to the PAN coordinator to check if a channel is idle or busy
- Unslotted CSMA-CA in non-beacon-enabled networks, a device wanting to transmit data waits a random period before it checks the channel
- ALOHA protocol in light loaded network. A device transmits data without checking the channels status
- Frame acknowledgement. If no acknowledgement is received the transmission of the frame is retried
- Cyclic Redundancy Check to detect errors
- Power consumption considerations through duty cycling, i.e devices shifting between sleeping and awake state, periodically listening to the RF channel to determine if a message is pending

Security

Because of the low-cost and low powered devices that is usually found in a LR-WPAN there are limits on the security overhead and are therefore dependent on higher layer implementation of several security architectural elements. The cryptographic mechanism in 802.15.4 is based on symmetric-key cryptography where higher layer processes provides the keys and insurance of a secure implementation of cryptographic operations and secure and authentic storage of keys. With correct cryptographic mechanism from higher layers, 802.15.4 provides data confidentiality, data authenticity and replay protection.

3.4 Bluetooth

Bluetooth is a short-range wireless technology operating in the 2.4GHz radio band. Originally, Bluetooth was developed to replace cabled technology as a means of transferring data. It has since progressed to become a leading short-range wireless technology in the consumer market for transferring of audio and data between devices. Bluetooth enabled devices interacts in either a point-to-point scheme or in a piconet. A point-to-point scheme defines a *master* and a *slave* device. For instance, a smartphone (master) and a fitness tracker (slave), allowing the smartphone to send commands to the fitness tracker and receive useful data back. It is possible to connect more slaves to form a piconet. Such a setup uses one master device and maximum 7 slave devices. To implement even more devices, an ad-hoc network can be set up with up to 10 piconets connecting to each other. Communication between the piconets is handled by the master devices, to form a peer-to-peer network. Recent specifications of Bluetooth (4.0 and newer) [6] introduced Bluetooth Low Energy to offer Bluetooth as a communication protocol to the constrained devices in the IoT. BLE and its protocol stack are further introduced in Chapter 4.

3.5 HART

The HART (Highway Addressable Remote Transducer) protocol is the leading standard of protocols for communication in industrial wired systems between smart devices and control/monitoring systems[3]. HART emerged in the 1980s and are built of the Bell 202 standard. It uses a master/slave model for communication, with two-way field communication between the master and slave. This provides two communication channels, the 4-20mA analog signal and a digital signal on top, with data rates of 1200 bps without interruption between the two. The analog signal is used to communicate the measured value from the device, while the digital signal is used to communicate additional information as device status, diagnostics, additional calculations, etc. The protocol can be setup in the different modes point-to-point (as illustrated in figure 3.3 or multidrop. Both setups allowing two masters (primary and secondary) in the communication setup.

[3]HART Overview: `http://en.hartcomm.org/hcp/tech/aboutprotocol/aboutprotocol_what.html`
[4]Source: `http://en.hartcomm.org/hcf/developer/images/developer_mktpos_clip_image006.jpg`

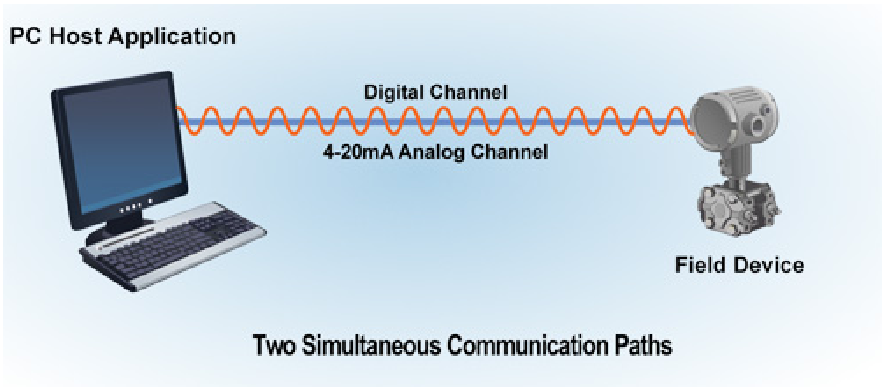


Figure 3.3: Illustration of HART protocol in point-to-point setup[4]

3.6 IPv6

It is widely believed that the number of devices connected to the Internet is growing rapidly and entering new domains and areas [2], and the projection for the future shows no signs of slowing down, rather instead growing past 20 billion connected devices by 2020 [56]. With IPv4 only supporting an address range of 2^{32} (4.2 billion) addresses, the industry identified the addressing issues with IPv4 a long time ago. This fact also shows how IPv4 is unable to work the vision for IoT with billions of connected devices. To be able to enable this vision, IoT has to make use of IPv4s successor IPv6. In the development of IPv6 the address range was increased from IPv4s 2^{32} addresses to an address range of 2^{128} (more than 340 trillion) unique addresses. Thus eliminating the problem of sufficient unique addresses for every devices in the IoT. In developing IPv6 several other features was added, which was not enabled in IPv4, such as [8]:

- **Header Format Simplification:** dropping or making optional some IPv4 headers
- **Improved Support for Extensions and Options:** change encoding, less stringent limits on length of options and greater flexibility in introducing new options
- **Flow Labeling Capability:** labeling of packets belonging to particular traffic flows
- **Authentication and Privacy Capabilities:** specified extensions to support authentication, data integrity and data confidentiality

6LoWPAN

In order to enable IPv6 on top of low power networks, an adaption layer, *IPv6 over Low-Power Wireless Personal Area Networks* (6LoWPAN), is defined [32]. 6LoWPAN offers header compression, fragmentation and reassembly of frames and stateless auto configuration to ensure full IPv6 compatibility in low power networks as 802.15.4, Bluetooth Low Energy and Sub-1GHz low power Radio Frequencies (RF).

3.7 CoAP

Constrained Application Protocol (CoAP) is a web transfer protocol for use with resource-constrained devices and networks. It was designed to simply translate to HTTP, but deliever services as low overhead, multicast support and simplicity for the constrained environments found in IoT. CoAP interacts in a similar client/server model as HTTP, but often operates in M2M settings resulting in an implementation with devices acting both as client and server. As with HTTP, CoAP uses a request/response, but deals with these interchanges asynchronously over UDP. To provide reliability, CoAP uses Confirmable or Non-Confirmable messages in a request. With a Confirmable message, an Acknowledgement with the same message ID is required to ensure a message has been delievered, if the client does not receive said acknowledgement it will retransmit its initial request to the server. A Non-Confirmable will not require an acknowledgement for the requests and responses. To ensure security CoAP implements DTLS with 4 different modes of security: NoSec, PreSharedKey, RawPublicKey and Certificate. NoSec disables DTLS and offer no security. Presharedkey offers a list of pre-shared keys and a list for eack key of which devices it can communicate with through DTLS. Rawpublickey offers each device with an asymmetric key pair validated with an out-of-band mechanism[5], the device has an identity calculated from the public key and a list of devices it can communicate with. With a Certificate, the device has an asymmetric key pair with a x.509 certificate signed by a common trust root [46].

[5]Out-of-Band uses an independent data stream to transfer the data

3.8 MQTT

MQTT is a client server publish/subscribe messaging transport protocol [36]. It is designed as a light weight, open protocol for use in constrained environments as the IoT. MQTT is built to run over TCP to provide one-to-many message distribution, by having messages being published to an address known as a topic. Clients subscribes to topics to listen for messages, and the server makes sure that messages sent to the topic is forwarded to the clients. Three quality of services are defined for message delivery: at most once (message loss may occur), at least once (messages are assured to arrive, but duplicates can occur) and exactly once (messages are assured to arrive exactly once). To subscribe to topics, clients may have to provide username and password to the server, and to ensure privacy TLS is implemented.

3.9 Elliptic Curve Diffie-Hellman Key Exchange (ECDH)

Elliptic Curve Diffie-Hellman Key Exchange is the elliptic curve variant of the well known Diffie-Hellman Key Exchange. Such an approach requires smaller key size than Public-Key Cryptography, thus reducing storage and transmission requirements between the two parties of the key exchange. For the IoT, this is beneficial because of the constrained resources offered by IoT devices. ECDH is done as follows [53]:

1. Alice and Bob agree on a basepoint G on the elliptic curve $E: y^2 \equiv x^3 + bx + c(modp)$
2. Alice and Bob creates their private keys: d_A and d_B, and public keys: $H_A = d_A G$ and $H_B = d_B G$. Where d_A and d_B are randomly chosen.
3. Alice and Bob transfers their public keys to each other over an insecure channel
4. Alice calculates $S = d_A H_B$, and Bob calculates $S = d_B H_A$
5. Alice and Bob ends up with the same shared secret: $S = d_A H_B = d_A(d_B G) = d_B(d_A G) = d_B H_A$

An attacker trying to perform a man-in-the-middle attack would only end up with H_A and H_B, but would not be able to derive d_A and d_B, unless he solved the discrete logarithm problem.

3.10 Elliptic Curve Juggling-Password Authenticated Key Exchange (EC-JPAKE)

Password Authenticated Key Exchange is a technique used to establish a secure channel of communication between two parties based on a shared password[12]. EC-JPAKE is a balanced PAKE algorithm where both parties of the key exchange uses the password to negotiate and authenticate a shared key to be used between them. It uses elliptic curves to compute the session key and provides off-line dictionary attack resistance, forward secrecy, known-key security and on-line dictionary attack resistance. By using a low entropy password as the shared secret between the two parties, and elliptic curves for negotiation, it is suitable for use in IoT, where a device is often connected to a smartphone when joining a network, and the two shares a password.

3.11 TLS and DTLS

Transport Layer Security (TLS) is the successor of Secure Sockets Layer (SSL) and is a cryptographic protocol aimed at making use of TCP to provide a reliable end-to-end secure service. TLS consists of 4 protocols used to obtain end-to-end security: *Record Protocol*, *Handshake Protocol*, *Cipher Spec Protocol* and *Alert Protocol* [48]. The Record Protocol is used to take an application message and fragment data into blocks, compress data (optional), apply a message authentication code (MAC), encrypt block, add a header and transmit the resulting block in a TCP segment. As described, it provides basic security services to higher layer protocols, specifically *Confidentiality* by defining a shared secret key used for encryption and *Message Integrity* by defining a shared secret key used to form (MAC). A Handshake Protocol is used to establish a session between the server and a client, it provides mutual authentication, negotiation to decide encryption and MAC algorithms, and cryptographic keys. The Change Cipher Spec Protocol is used to update the cipher suite in use on the connection. Lastly, the Alert Protocol is used to alert messages to the peer entity. Alert messages can be either a *warning* or a *fatal* message, if it is a fatal message the connection is immediately terminated, and contains information on what the alert is (for instance: unexpected_message, handshake_failure, certificate_revoked, etc.)

DTLS

Datagram TLS (DTLS) is a continuation of TLS, to provide TLS over UDP. The protocol is similar to TLS and adds only some functionality to handle the problem of datagrams being lost, duplicated or received in wrong order. To handle this DTLS adds the following [42]:

- DTLS Record mapped to a datagram
- An explicit sequence number is added in the Record Protocol to correctly verify the TLS MAC
- DTLS can handle lost, duplicated, reordered or modified datagrams
- Stateless encryption, meaning RC4 cannot be used with DTLS
- Fragmentation of handshake messages over a number of records
- Retransmission of datagrams if timeout
- Protection against Denial of Service/Spoofing attacks

3.12 Summary

In order to meet the requirements and realize the vision of IoT, several technologies and protocols are essential. M2M and Wireless Sensor Networks is the pillars of IoT, and 802.15.4 is one of the technologies that brings wireless communication to constrained devices. Bluetooth and HART are well-established technologies which have been further developed to provide Bluetooth Low Energy and WirelessHART to IoT. Protocols as MQTT, CoAP, ECDH, EC-JPAKE, and TLS/DTLS have been used to realize some of the standards currently in the market.

4 | Introduction of IoT Protocol Stacks

In this chapter, an introduction to the IoT protocol stacks: ZigBee, Thread, Z-Wave, Bluetooth Low Energy, WirelessHART and the author's proposed IP-Smart is presented. Those protocol stacks were chosen on the basis of investigating stacks primarily developed to the different use-case presented in Chapter 2, and because they are believed to be protocol stacks which have established themselves in the marketplace. IP-Smart is a proposed protocol stack based on IEEE and IETF standardized protocols, which potentially could compete in all use-cases and not restrict device interoperability by using proprietary protocols or technologies. Each section of this chapter will introduce one of the chosen protocols, with a short introduction of background and history, some key technical details, and a mapping of the protocol stacks into the proposed IoT model of Chapter 2. And finally, a section concluding what has been presented in this chapter.

4.1 ZigBee

Established in 2002[1], the ZigBee alliance is formed by 450 members comprised of tech companies, universities, and government agencies to create and develop IoT standards. The goal is to provide reliable and easy-to-use standards for use in consumer, commercial, and industrial areas as smart homes, healthcare, smart energy and more. ZigBee's protocol stack is built on the well-known wireless standard 802.15.4 and implements its own network layer and an open application layer for specific ZigBee applications or OEM applications. In order to comply with the IoT model proposed previously in this thesis, the ZigBee protocol stack is defined as in figure 4.1, where the new ZigBee 3.0 standard[2] has been chosen at the network layer. ZigBee 3.0 is the

[1]ZigBee Alliance: `http://www.zigbee.org/zigbeealliance/`
[2]ZigBee 3.0: `http://www.zigbee.org/zigbee-for-developers/zigbee3-0/`

new standard unifying the previous network layer standards: ZigBee PRO, ZigBee RF4CE, and ZigBee IP/920IP.

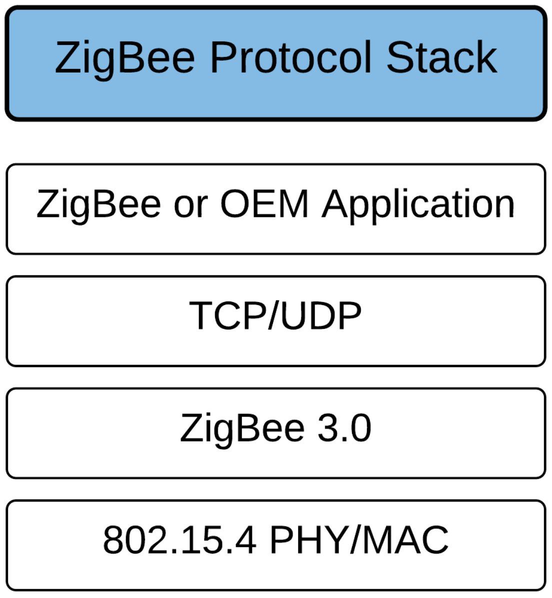


Figure 4.1: Illustration of ZigBee's protocol stack

Technical Details

A ZigBee network uses a mesh topology[3] to allow self-healing between communication devices, thus preventing a single point of failure in the network [35]. ZigBee networks are scalable to cope with local networks of greater than 250 nodes, and the network consist of 3 different node types: coordinator, router, and end device. A **Coordinator** is the main node of the network, it is the node which establishes the network and stores information of the network (including security keys). A **Router** is intermediate nodes which relays and routes packages in the network. The **End-devices** is the low powered devices in the network which can only send their data to a router, to be further processed in the network. With the introduction of ZigBee 3.0, the restraints of devices not being able to talk to each others is removed, thus unifying the different application profiles (smart home, health monitors, etc.) supported in ZigBee. This is achieved by giving devices in a ZigBee network a shared "base device" software implementation which

[3] A *mesh network* is a network topology where every device is interconnected to each other

incorporates a set of common commissioning methods to enable the unification of all device types.

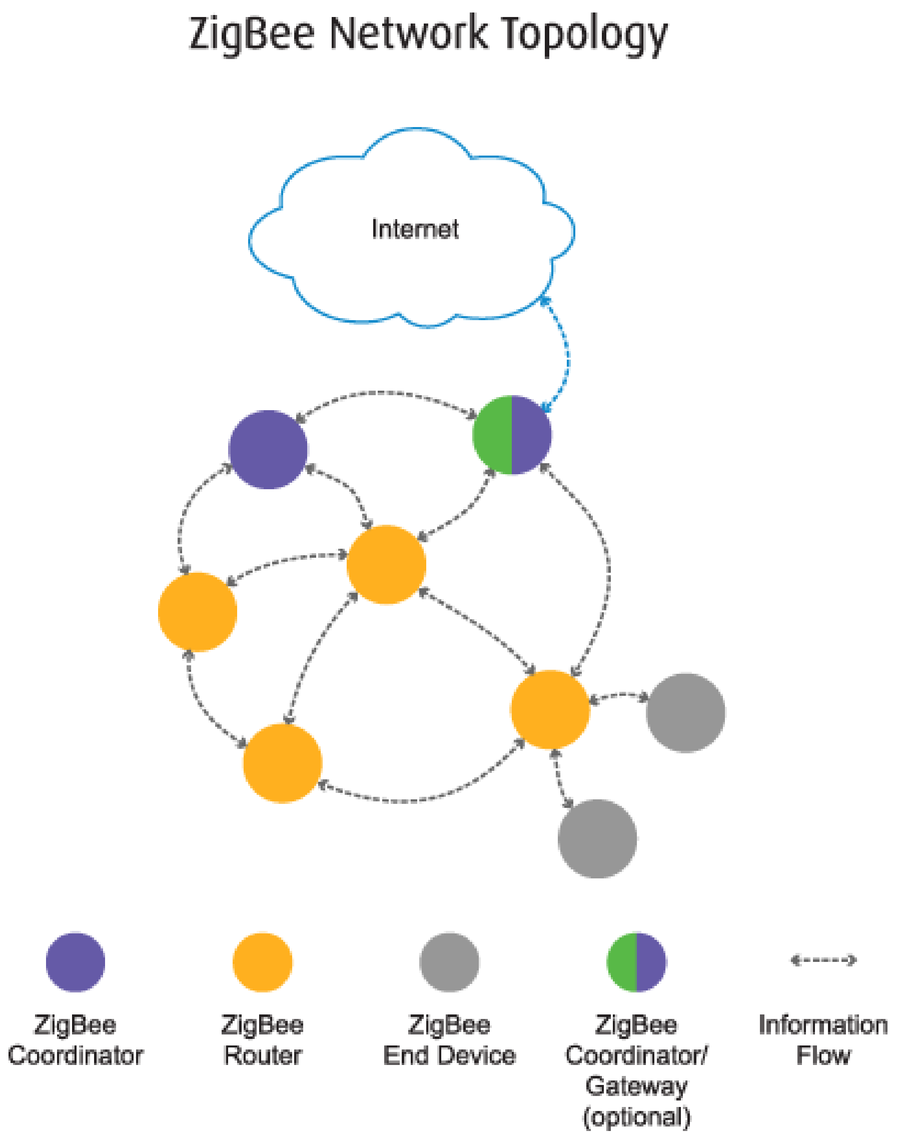


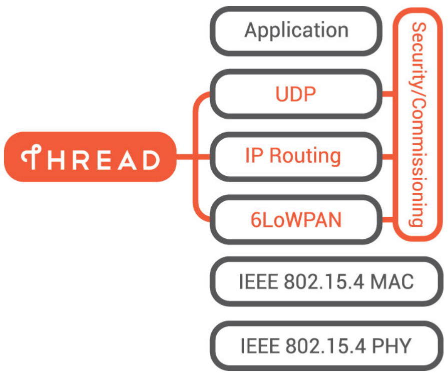
Figure 4.2: Illustration of ZigBee network topology[4]

4.2 Thread

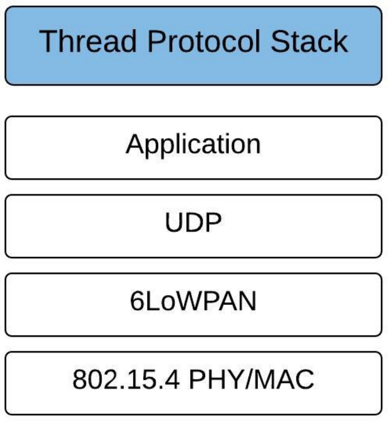
Thread Group was unveiled in July 2014 by Google Inc's Nest Labs [5] as an industry group consisting of several of the leading technology companies, to create and cooperate on a new standard

[4]Source: http://zigbee.org/wp-content/uploads/2014/11/network_topology_ZigBee-PRO.gif
[5]http://www.reuters.com/article/us-google-nest-idUSKBN0FK0JX20140715

to provide a user-friendly, secure, scalable and battery friendly protocol stack for every device of the home. The protocol stack is built on well established and known technologies as 802.15.4, 6LoWPAN and UDP, and has defined its protocol stack as in figure 4.3a. To comply with the definition of IoT and the IoT protocol stack in this report, the protocol stack of Thread has been modified and presented as in figure 4.3b.



(a) Thread protocol stack



(b) Modified Thread protocol stack

Figure 4.3: Illustration of Thread specified and authors modified protocol stack

Technical Details

A Thread network consists of several different device types. A **Border Router** is a specific router acting as a gateway from the 802.15.4 network to adjacent networks using other physical layers as for instance 802.11. Setting up a Thread network requires at least 1 border router, but more can be implemented if necessary. Apart from the gateway functionality of the border router, it also offers routing services for off-network operations[49]. **Routers** provides routing services, joining operations and security services to network devices. They are designed to always stay on, but could be downgraded to a **Router-eligible End Devices** (REED). REED acts as backup devices in the network and are only activated as routers when necessary, and activation of REED are done without user interaction. Lastly, the **Sleepy End Devices** are the host devices of the Thread network and communicates only with their parent router[51].

Thread networks will automatically initialize as a mesh network (as illustrated in Figure 4.4 if there is more than one router (with a set limit of max 32 routers) in the network. Every router keeps a record of all routes in the network such that the network is always up-to-date and connected by single hop Mesh Link Establishment (MLE) messages[27], which establishes and configures secure radio links, detects neighbor devices, and maintains routing costs between devices.

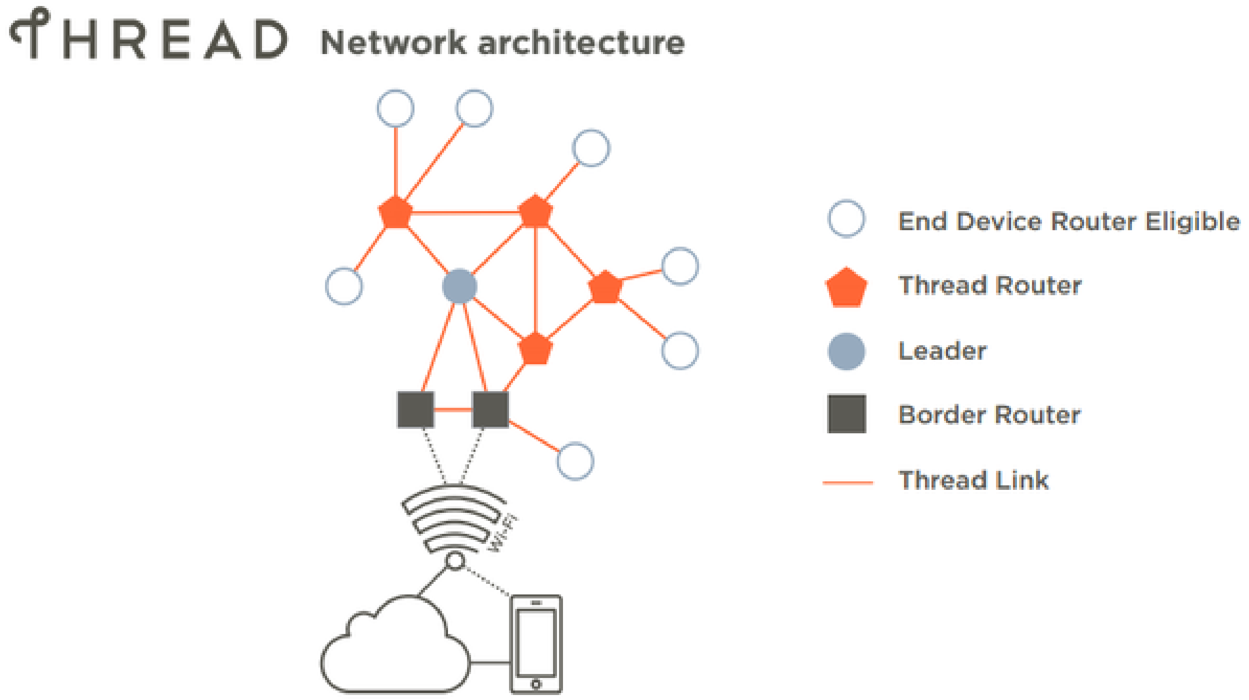


Figure 4.4: Illustration of Thread network topology[6]

In terms of security in a Thread network a set of roles must be established in the network. These are as follows: the border router, a joiner router and a commissioner. There are different scenarios how these roles are defined:

- border router is joiner router
- border router is not joiner router
- joiner router is commissioner
- joiner router is not commissioner

A commissioner will have connectivity in the network either directly through the Thread network or externally through a WLAN[50].

[6]Source: http://media.bestofmicro.com/J/U/511338/gallery/thread-architecture_w_600.png

4.3 Z-Wave

The Z-Wave alliance was established in 2005 by manufacturers of smart home products. Development of the Z-Wave protocol stack was done to overcome the obstacles of incompatible proprietary technologies, non-converging standardization attempts and undue cost for the whole market chain[7]. As Z-Wave is a proprietary solution, information on the protocol stack was scarce. A possible illustration of Z-Wave's defined protocol stack is shown in figure 4.5. In order to map the protocol stack to the author's suggested model. the network and transport layer have been defined as Z-Wave Proprietary, as shown in figure 4.6.

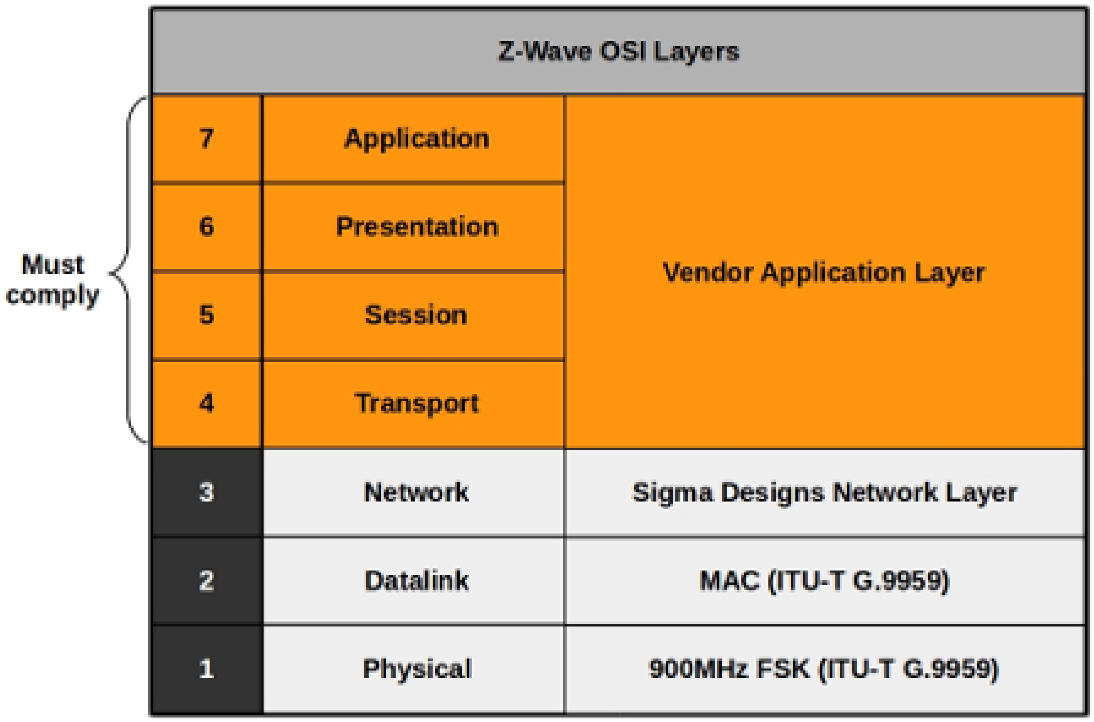


Figure 4.5: Illustration of Z-Wave protocol stack[8]

[7]Z-Wave Alliance History: http://z-wavealliance.org/z-wave_alliance_history/
[8]Source: http://www.allbits.eu/wp-content/uploads/2015/02/ZWaveOSILayers.jpg

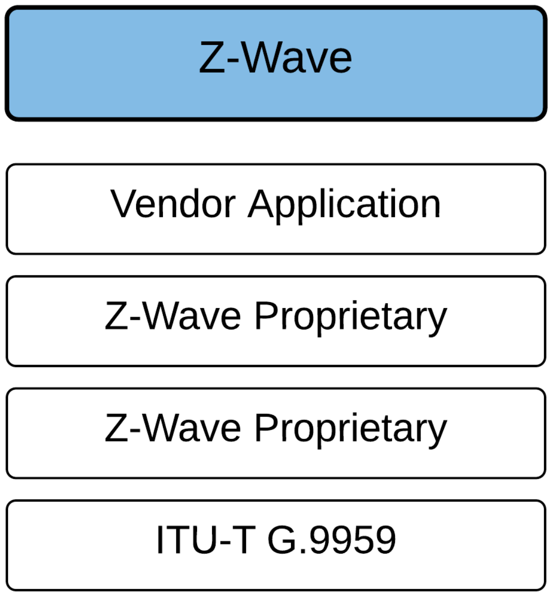


Figure 4.6: Illustration of modified Z-Wave protocol stack

Technical Details

Z-Wave uses the ITU G.9959 radio technology at the radio layer. It operates in the sub-GHz spectrum at 868.42 MHz in Europe and 908.42 MHz in the US and Canada as the main frequencies. Thus offering data rates of 40-100 kbit/s in a wireless mesh network topology (as illustrated in figure 4.7). A Z-Wave network consists of two different devices, **the gateway**, and **end-devices**. The gateway is the leader of the network and handles the network management, security management and connecting the Z-Wave network to the Internet.

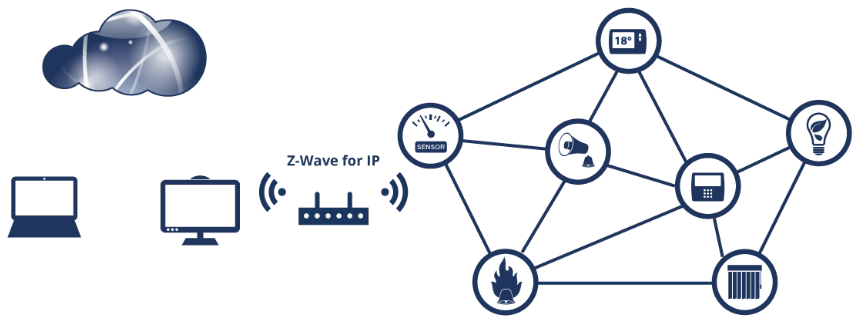
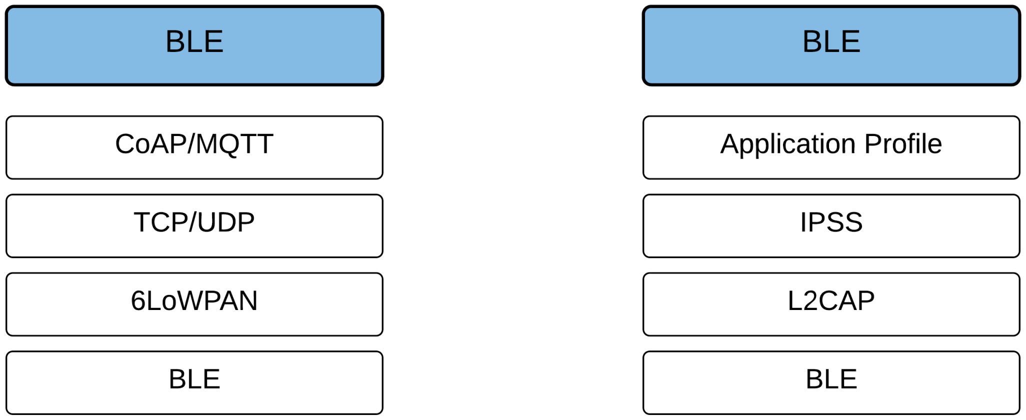


Figure 4.7: Illustration of Z-wave network topology[9]

[9]Source: http://z-wave.sigmadesigns.com/img/z-wave_gateway_controller_for_ip.png

4.4 Bluetooth Low Energy

Bluetooth Low Energy (BLE) was introduced in the Bluetooth 4.0 specification[6]. It was designed to enable Bluetooth technology to small battery powered devices and sensors and offer easy integration with handheld devices such as smartphones and tablets. Such an approach differs from other IoT protocol stacks in terms of out-of-the-box support for communication with handheld devices, neglecting the need for a border router as for instance Thread and Zigbee has. BLE's protocol stack is defined in figure 4.8, where 4.8a shows BLE with 6LoWPAN[34] and 4.8b shows BLE with Bluetooth specific protocols.



(a) Modified BLE protocol stack with IPv6 support (b) Modified BLE protocol stack with Bluetooth standards

Figure 4.8: Illustration of mapped BLE protocol stacks

Technical Details

At the technical level, BLE offers an over-the-air data rate of 1 Mbit/s, with application throughput of 0.27 Mbit/s in one of 40 channels operating in the 2.4GHz band[43]. Offering a possible range of up to 50m between devices, in a master-slave topology, with each master-slave connection at different physical channels[38] as illustrated in figure 4.9.

[10] http://www.summitdata.com/blog/wp-content/uploads/2013/02/BLE_topology1.jpg

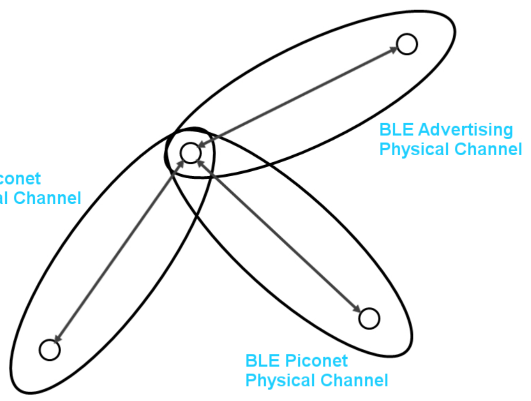


Figure 4.9: Illustration of Bluetooth Low Energy network topology[10]

4.5 WirelessHART

WirelessHART is an industrial wireless standard for industrial process automation. Built on the HART (Highway Addressable Remote Transducer) protocol, released in 2007[5], it became the first approved industrial open wireless standard (IEC62591, EN62591) in 2010, and has grown to become one of the important standards in IIoT. WirelessHART's protocol stack has been mapped to the proposed model as seen in figure 4.10.

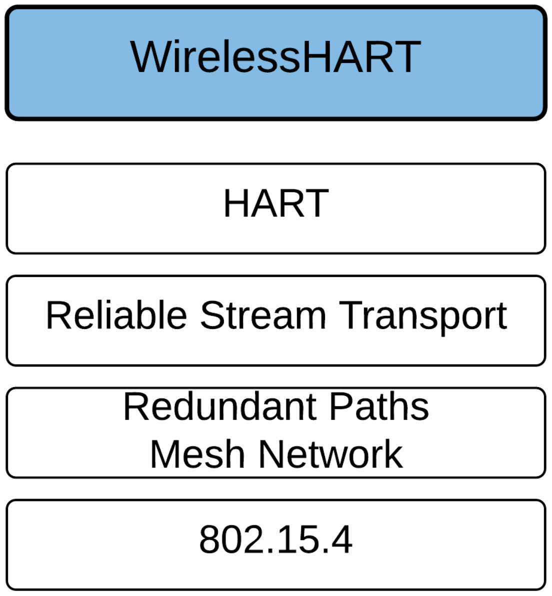


Figure 4.10: Illustration of WirelessHART protocol stack

Technical Details

In a WirelessHART network, the **gateway** serves as the access point, network manager, security manager, gateway, and HOST interface. A gateway is the responsible node in the network to manage the network, commission new devices and communicating with the wired HART protocol[31]. Apart from the gateway, a WirelessHART network consists of **access points/router devices**, and **field devices**. With the implementation of several access points in the network, it can be scaled to several thousands of devices if needed. WirelessHART networks are built on 802.15.4 in the 2.4GHz band, providing a mesh network topology with data rates of 250kb/s as shown in figure 4.11. WirelessHART does not use standard IP networking, and the rest of the protocol stack is built on HART specific protocols.

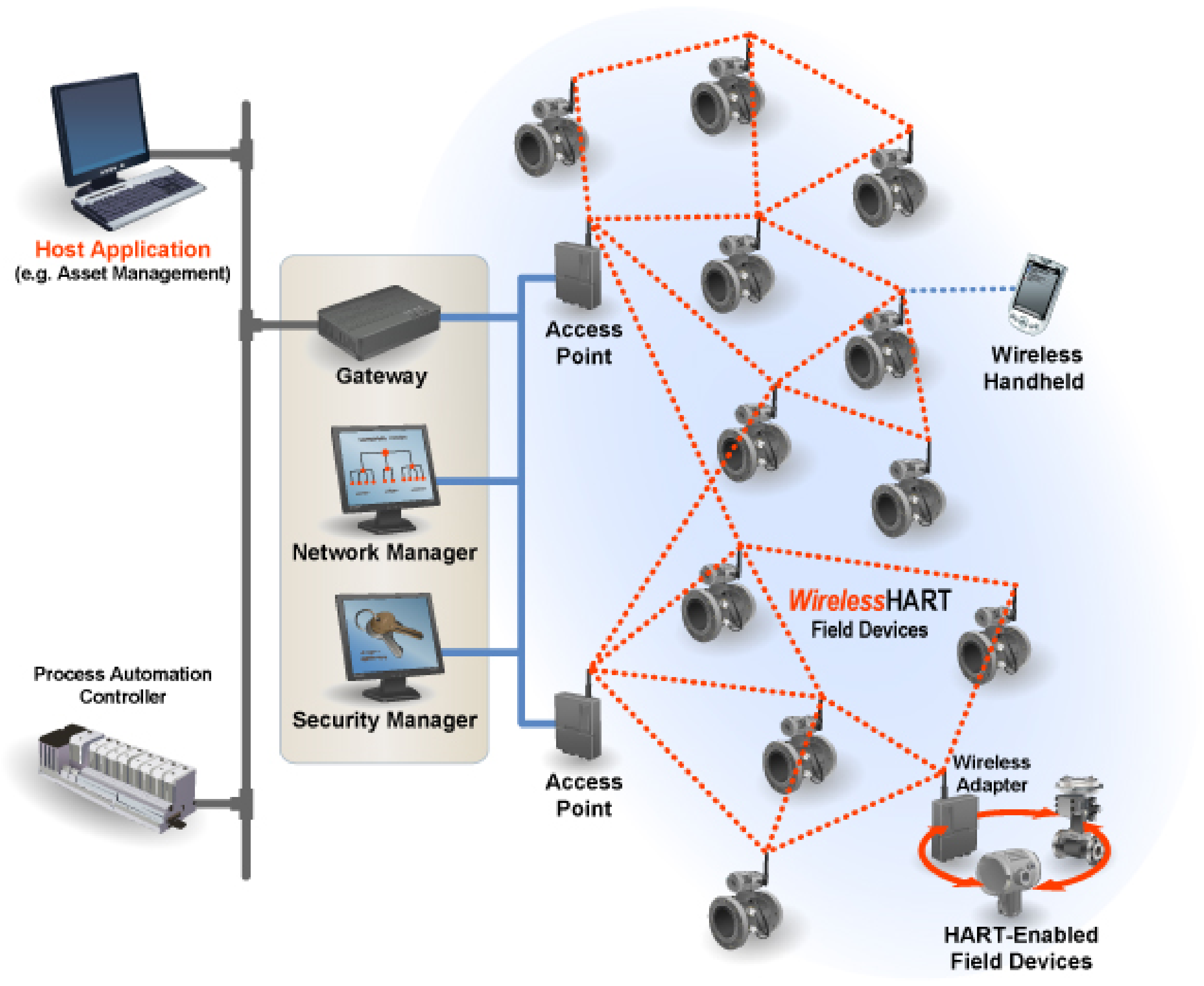


Figure 4.11: Illustration of WirelessHART network topology[11]

4.6 IP-Smart

IP-Smart is the author's proposal of a protocol stack built from IEEE and IETF standardized protocols, and previous work on a standardized protocol stack for the IoT[39]. IEEE and IETF has proposed several standardized protocols as 802.15.4 [20], 6LoWPAN [32] and CoAP [46] to connect the constrained devices of IoT. Such a protocol stack would offer IP-technology to constrained devices and be built on open, well-known, standards. IP-smart is thus believed to be able to compete with the formerly presented protocol stacks and not be restricted by proprietary protocols and technologies, in the work towards interoperable IoT devices.

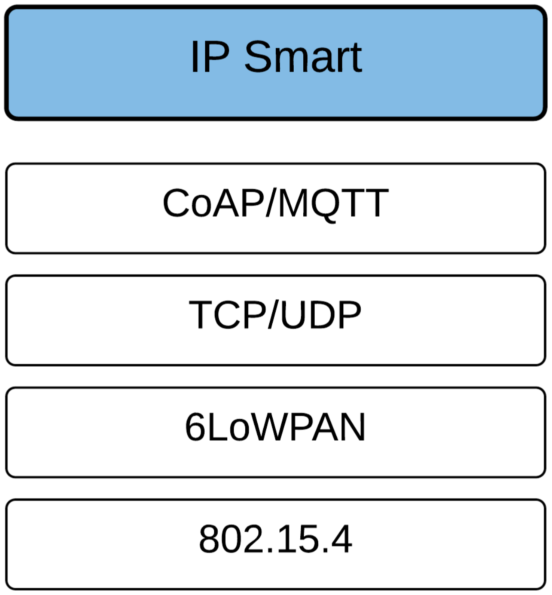


Figure 4.12: Illustration of proposed IP-Smart protocol stack

Technical Details

With the use of 802.15.4, the IP-Smart protocol stack would offer a star or mesh topology for devices. Working in the 2.4GHz band offers 16 channels and data rates of up to 250kbits/s, with a range of 10-30m. A device can either be a **Full-function Device** (FFD) or a **Reduced-function Device** (RFD). If a device is a Full-function Device, it can serve as the PAN coordinator

[11]Source: http://en.hartcomm.org/hcp/tech/wihart/images/Expanded_WirelessHART_Mesh_Network.jpg

of the network. It will be responsible for maintaining control of the network and relaying routing information etc. RFDs are not able to take up the role of coordinator in the network and only offers simple communications with network coordinator.

4.7 Summary

There are a lot of different standards trying to solve the challenges brought by IoT. ZigBee, Thread, and WirelessHART represents both established standards and more recently released standards build on top of 802.15.4. Bluetooth Low Energy is a continuation of the well-established Bluetooth technology, Z-Wave is a proprietary solution created for the smart home, and IP-Smart serves as a proposal to create a protocol stack suited for different use-case areas. They all offer different approaches to try to win out in the marketplace.

5 | Review of IoT Protocol Stacks

This chapter will present a more in-depth comparison of the security features provided by the protocol stacks presented in chapter 4. The protocol stacks have been created with different areas of use in mind, and the comparison will make use of the presented use-case scenarios of chapter 2.5. Such an investigation is done to determine how the protocol stacks solve the identified security requirements of each use-case scenario. Investigation into whether a protocol stack is applicable beyond its intended area of use is also done. Table 5.1 and 5.2 provides a simple overview of the provided security features offered by the presented protocol stacks of Chapter 4.

Section 5.1 will present the commissioning procedure of the different protocol stacks. The commissioning procedure consist of a device joining a new network, said device being authenticated and key/cipher agreements between device and network.

In Section 5.2, an investigation into how different protocol stacks manages a network in terms of access control, network protection, application payload protection, and device management is presented.

Section 5.3 presents the findings and maps them to the different use-case scenarios and the specific security requirements of each scenario. A short discussion of how different protocols stacks suits the use-cases is also presented.

5.1 Commissioning

In the device lifecycle, the initial setup is the commissioning procedure. Commissioning is the procedure of a device requesting to join a network, the device being authenticated to the network, the key exchange between the joining device and the authenticator device, and lastly the

	Commissioning	**Authentication**	**Key Exchange**	**Confidentiality**
ZigBee	MAC Association	Mutual Symmetric-Key Entity Authentication	Symmetric-Key Key Establishment	AES128
Thread	Passcode/QR Code	DTLS Handshake	EC-JPAKE	AES128
Z-Wave	PIN/QR-code	TLS Handshake	ECDH	AES128
Bluetooth Low Energy	Passkey	LE Legacy Pairing	ECDH	AES128-CCM
WirelessHART	Pre-configured Join Key	Join Key Confirmation	Unknown	AES128
IP-Smart	Vendor Specifc Implementation	DTLS Handshake	ECDH/APKES	AES128

Table 5.1: Overview of security features provided in commissioning by the protocol stacks

confidential transfer of network/security parameters to the joining device. Table 5.1 gives an overview of the different technical solutions implemented by the different protocol stacks to solve the security requirements of the commissioning procedure. In the commissioning column of Table 5.1, passcode, and passkey is mentioned as possible commissioning services. These are from the information gathered believed to be low entropy password used as a pre-shared secret between a smart device and joining device and are provided by user input in a smart device application. In general, the commission phase of the commissioning serves to communicate a pre-shared secret between authenticating device and joining devices. This pre-shared secret is then further used in key exchanges to derive session keys, long-term keys or other specified keys in the protocol stack.

5.1.1 Commission

ZigBee

To start the commissioning procedure, a joining device must start a network discovery, which enables it to look for networks in its proximity, and identify a coordinator. When the device has identified the network it wishes to join, a MAC layer association is initialized. A joining node must send a join request to the coordinator, in which the coordinator starts a search in the network for the device's 64-bit IEEE unique identifier[21]. If the identifier is not found in the network, the coordinator adds it to its neighbor table and confirms the joining request.

Thread

To initiate the commissioning process in Thread, a smart device (commissioner) is used to connect with the joining device. A passcode can be input by a user to a smart device application, or a QR-code on the device can be scanned by the smart device application to perform the commissioning.

Z-Wave

To initiate commissioning, a user must enter a pin code or read a QR code with a smart device, from the joining device. The gateway of the network is put in *inclusion mode* which enables it to actively listen to requests from new devices, and accept it into the network.

Bluetooth Low Energy

BLE provides 3 different modes of commissioning: Just Works, Passkey, and Out-of-Band (OoB) [6]. In terms of security, Just Works is the weakest of the methods, as it allows every device to pair. Passkey entry is dependent on a smart device supporting keyboard input, such that a 6-digit passkey can be used. If the devices support some OoB technology, as NFC for instance, the OoB method can be used to pair the devices.

WirelessHART

Field devices will need a Join Key to be able to identify and connect to a network. This join key can either be common for all field devices or unique for each device. The unique Join Key is recommended and stated to provide stronger security. The system guide [19] states that the Join Key must be impossible to read both physically and digitally. Physical tampering resistance and proper encryption of channels used to send Join key must be provided. There is, however, no information on how to achieve this, leaving the correct implementation to the manufacturers of the devices.

IP-Smart

A commissioning procedure would have to be implemented by manufacturers, as the proposed IP-Smart protocol has no specified function of initializing a commissioning.

5.1.2 Authentication

ZigBee

Authentication is provided by a *Mutual Symmetric-key Entity Authentication* protocol. A 16-octet random challenge[57] is created by both the joining device and the Trust Center and then sent to each other to calculate a response in order to achieve mutual authentication.

ZigBee IP[1], offers ZigBee with full IP mesh networking and implements TLS 1.2, to use TLS Handshake for authentication. However, ZigBee 3.0 is stated[2] to be built on ZigBee PRO which continues the use of the ZigBee specific protocols.

Thread

When a device wishes to join the network it instigates a DTLS handshake with the commissioner to authenticate itself [50]. The commissioner is a smart device, for instance, a smartphone, and connects directly with the joining device through the border router. The DTLS handshake ensures authentication of the device and the secure transfer of network and security parameters as illustrated in figure 5.1.

Z-Wave

Z-Wave has support for TLS 1.1, and could use TLS handshake for authentication, however, a recent announcement[3] highlights an *Authenticated Deployment* as one of the new security features. Information on what the authenticated deployment is or how this is achieved could not be found, and thus it is not possible to say anything about if this is some specific device authentication from Z-Wave or the strength of it.

[1] ZigBee IP: http://www.zigbee.org/zigbee-for-developers/network-specifications/zigbeeip/
[2] ZigBee 3.0: http://www.zigbee.org/zigbee-for-developers/zigbee3-0/
[3] Z-Wave S2 announcement: http://www.sigmadesigns.com/news/sigma-designs-announces-advanced-iot-security-measures-for-the-smart-home/

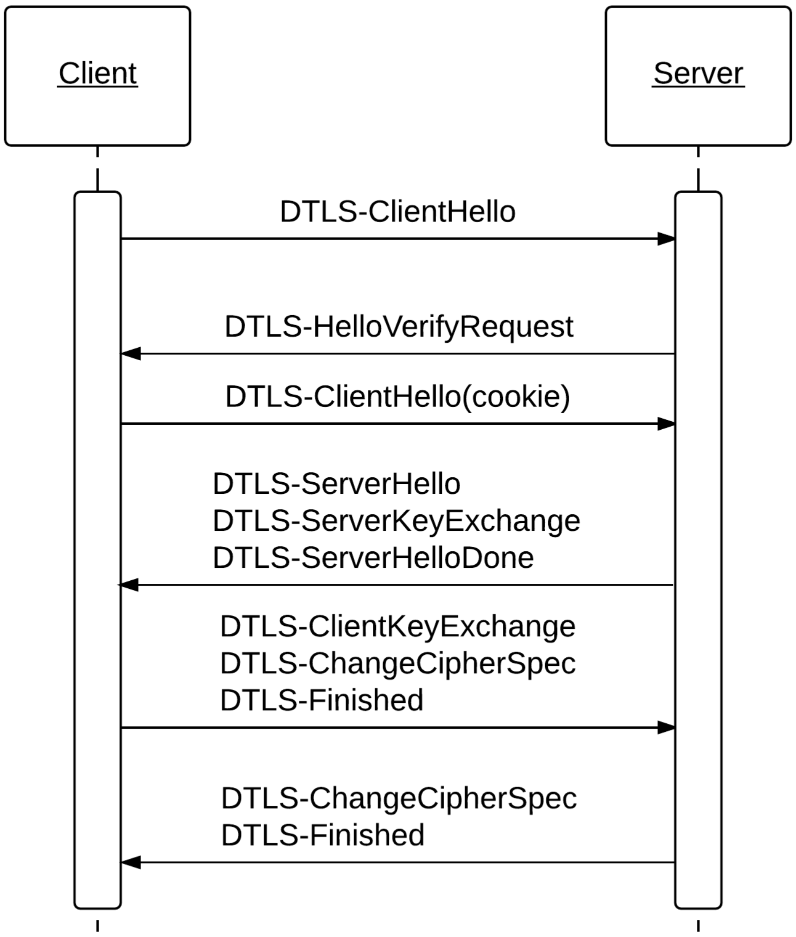


Figure 5.1: Illustration of DTLS Handshake

Bluetooth Low Energy

BLE authentication is derived from the commissioning methods. Just Works will authenticate every device sending a pairing request, while Passkey or OoB is dependent on user input/interaction to authenticate a device.

WirelessHART

There is little information to be found on how WirelessHART achieves device authentication. All information found suggests that the Join Key serves as the authentication of a device to the network[4] (with the Join Key possibly being universal among devices), and that it is used to au-

[4]Source: http://en.hartcomm.org/hcp/tech/wihart/wihart_security.html

thenticate the NPDU (Network Protocol Data Unit) payload and verify a Message Integrity Code (MIC) of the joining request[40].

IP-Smart

With the use of CoAP or MQTT, the IP-Smart protocol stack can offer TLS or DTLS handshakes to ensure authentication of devices.

5.1.3 Key Exchange

ZigBee

ZigBee uses a *Symmetric-key Key Establishment* (SKKE) protocol for key exchange. It establishes a *Link Key* between the joining device and the Trust Center. A master key is required by the joining device to start the SKKE, this can either be: pre-installed on the device, installed by the Trust Center or be some user-entered data (PIN code, password or key)[57].

Previous work [58, 41, 55] shows that the key exchange protocol of ZigBee is vulnerable to sniffer attacks when ZigBee is used in *Standard Security Level.* The Trust Center will send the network key unencrypted over-the-air to devices when devices want to join the network, and a capture of this key could give the attacker the possibility to capture data in the network and perform replay attacks. Setting ZigBee in *High Security Level* would remove the sending of the network key over-the-air unencrypted, and mitigate the vulnerability.

ZigBee IP offers X.509 for certificate and key exchange, but as mentioned with ZigBee authentication, ZigBee 3.0 continues the use of ZigBee specific protocols as SKKE.

Thread

Key Exchange in Thread is done through EC-JPAKE [12]. It uses the NIST-256 elliptic curve and the shared secret (passcode/QR-code) of the commissioner and the device to provide mutual authentication and establish a session key between them. Further details on how key exchange is done with EC-JPAKE is found in Section 3.10.

Z-Wave

The S2 framework announcement for Z-Wave states that ECDH has been implemented as key exchange protocol for Z-Wave (see Section 3.9 for information on ECDH).

Bluetooth Low Energy

With the implementation of LE Secure Connections in the Bluetooth 4.2 specification[6], BLE can make use of ECDH for key exchange. BLE devices on previous Bluetooth standards (4.1 and former) implements its own *Key Transport Protocol* for key exchange. Dependent on the commissioning method, the protocol agrees on a TK (Temporary Key) and derives the Short Term Key (STK), which again is used to derive the Long Term Key (LTK). If the BLE key exchange is not based on ECDH cryptography, it has a weakness against eavesdropping [38, 43]. Therefore, an attacker might be able to capture the pairing frames and determine the LTK as shown by Ryan [43], and carry out a MitM attack.

WirelessHART

Information on key exchange could not be found. Previous work [40] has also identified the lack of specification of key management.

IP-Smart

Key Exchange proves to be one of the most challenging aspects of the IP-Smart protocol stack. Early work on 802.15.4 [45] shows problems with no support for group keying, network shared key vulnerability in replay protection and pairwise keying inadequately supported. More recent work shows progress in this problem and ECDH has been implemented [22]. Other work [29] suggests the use of the Adaptable Pairwise Key Establishment Scheme (APKES) as an approach to solving the issue of Key Exchange.

	Application Payload Protection	**Network Protection**	**Access Control**	**Device Management**
ZigBee	ZigBee Specific	Network Key	Permission Configuration Table	Trust Center
Thread	DTLS	Network Key (HMAC)	Access Control List	Leader node
Z-Wave	TLS 1.1	Network Key	Unknown	Gateway Controller
Bluetooth Low Energy	TLS/DTLS	IRK and CSRK	Vendor Specific Implementation	Handled by Master device
WirelessHART	Unknown	Unique Keys for Broadcast or Point-to-Point	Access Control List	Gateway, Network Manager, Security Manager
IP-Smart	TLS/DTLS	Network Key	Access Control List	PAN Coordinator

Table 5.2: Overview of security features for managing the network

5.2 Managing the network

After devices have been commissioned onto the network, the protocol stacks must ensure management of the network. Table 5.2 highlights the different security features implemented to ensure that a network is managed and protected.

5.2.1 Access Control

ZigBee

ZigBee devices define a *Trust Center* in the network responsible for functions as key distribution, end-to-end application configuration management, removing devices from the network, updating device list, and the maintenance of the permission configuration table (an access control list, determining authorization levels of devices) [57]. Which device that takes up the role of Trust Center depends on whether a device is pre-loaded as a Trust Center. In this case, every device joining the network must have the Trust Center address and initial master key pre-loaded onto the device. If not pre-loaded, the Trust Center defaults either to the ZigBee Coordinator or a device chosen by the ZigBee Coordinator.

Thread

802.15.4 offers an Access Control List to Thread, with information about trusted neighbors. Apart from this, Thread operates as a self-managing network, with every device sharing the same information, and if specified as REED, can become a router, and commissioner in the network if needed.

Z-Wave

There has not been found any information regarding whether Z-Wave implements some sort of Access Control List.

Bluetooth Low Energy

No ACL is defined by BLE, the master device is in control if its slave devices.

WirelessHART

With the use of 802.15.4, WirelessHART can implement an Access Control List. Technical notes and system guides of WirelessHART [31, 7, 19] states that the gateway should have a security policy defining different user accounts with differing access to critical security and configuration parameters. Implementation of this security policy seems to be left to the manufacturers, implying that manufacturers would need to provide sufficient security of the user accounts to prevent unauthorized access to the network.

IP-Smart

In the proposed IP-Smart, 802.15.4 would implement an Access Control List (ACL) [45] with information on address, security suite, key, last IV (initialization vector) and replay counter. This ACL would be used by devices to ensure communication only with other trusted devices.

5.2.2 Network Protection

ZigBee

ZigBee uses a *Network Key* to encrypt network frames with AES128-CCM to ensure network protection[35, 57]. It is a common key shared among all devices in the network, and an alternate network key is generated at different intervals to replace old network key and provide key rotation. Previous work [58] however suggests that an automatic key rotation could not be identified in an eleven-month time frame, revealing a severe flaw in the implementation of the automatic key rotation.

Thread

The Thread network uses a *Network-wide Key* to protect 802.15.4 MAC (Media Access Control) data frames from eavesdropping or targeted disruption. The Network-wide Key is reported [52] to be an HMAC hash of a 32-bit key identifier using a master key, with no further information how the master key is derived.

Z-Wave

There is information of a *Network Key* [15, 14] being used between controller and devices. From this information, the Network Key is believed to be the same key as the temporary key set in the device's firmware. Thus deducing that the network key is a 16-byte key, which possibly could be only zeros as shown by Fouladi and Ghanoun[14].

Bluetooth Low Energy

BLE implements an Identity Resolving Key (IRK) to resolve private to public device address mapping, by doing this, devices can mitigate the risk of being tracked by its static public address. A Connection Signature Resolving Key (CSRK) is used to enable data signing to protect a connection between two devices.

WirelessHART

WirelessHART has implemented a *Network Key* (known by all devices) and a *Session Key* (known only by the two communicating devices) to provide network security. A Network key is used to encrypt and protect data from attackers outside the network, while a session key is used to protect the network path between source and destination [5].

Bayou et al [5] showed that there is a vulnerability in the Disconnect DLPDU feature, and the network key. Those two weaknesses together can enable an attacker to disturb the routing protocol, isolate nodes and harm the network behavior.

IP-Smart

A Network Key model could be used to provide network protection in the IP-Smart protocol stack. There is, however, evidence [45, 29] which suggests that such an approach would not provide replay protection with the standard ACL implementation using replay counters, and also make devices susceptible to be compromised by physical tampering.

5.2.3 Device Management

ZigBee

Device management is handled by the *Controller* and/or *Trust Center* and provides functions as updates of device lists, and revoking devices from the network if a device does not comply with the set security parameters.

Thread

A *Leader* node is responsible for making decisions within the network. It can promote *Router Eligible* devices to *Router* to improve connectivity of the network. All routers of the network send periodic MLE messages to update routing information and other parameters on devices to maintain connectivity of network. If a leader node fails, the network automatically promotes a router to become a leader. No information on how to revoke a device from the network has been found.

Z-Wave

A *Controller* device is set to manage all the other devices in a Z-Wave network. From the information available the *Controller* device and the *Gateway* is possibly the same device. This device would manage the communication between the application on the smart device, and the devices on the Z-Wave network. Management of devices thus seems to be handled by the smart device application. Since the Z-Wave network operates as a mesh network, all devices are capable of sending updates of routing information to other devices.

Fuller and Ramsey [15] reports exploits of the gateway and the possibility to inject rogue controllers in the Z-Wave network. The gateway uses HTTP POST and HTTP GET requests to send commands to their server, which then relays information to the network. By using *Burp Proxy*[5] Fuller and Ramsey was able to modify request made from the smart device application and send it to devices. They show that even devices as door locks which use encryption on data packets will accept the modified requests. They also demonstrate the possibility of injecting a rogue controller into the network, gaining full control of all the devices in the network. These vulnerabilities seem to be vendor specific, suggesting that vendors will have to ensure security on their products. Recently a new framework for Z-Wave, the S2, has been announced[6] claiming to remove the vulnerability found by Fuller and Ramsey.

Bluetooth Low Energy

BLE works in a device-to-device manner, leaving the master node in charge of handling the connection between the two devices.

WirelessHART

As presented in Section 4.5 and Figure 4.11, the *Gateway* of a WirelessHART network serves as the roles of the gateway, network manager, and security manager. These roles can either be integrated into one enclosed device or distributed across different devices in the network, with the integrated solution seen as the preferred option [31]. It is responsible for the generation

[5] Burp Proxy: `https://portswigger.net/burp/proxy.html`

[6] Z-Wave S2 announcement: `http://www.sigmadesigns.com/news/sigma-designs-announces-advanced-iot-security-measures-for-the-smart-home/`

and maintaining of routing information, allocating communication resources[5] and can inflict a quarantined or suspended state on a device to limit the actions of the device. The quarantined state still leaves a security clearance on the device to talk to the network manager, but can no longer perform data acquisition or control functions and is not able to communicate with the gateway in any other way. The suspended state is not clearly explained, but is assumed to be the state after the Disconnect DLPDU is sent [5], which disconnects a device from the network.

IP-Smart

A PAN coordinator has to be established for an IP-Smart network, which would be in charge of device management.

5.3 Use-Cases

In this section the findings of Section 5.1 and 5.2 is mapped to the security requirements of the different use-case scenarios presented in Chapter 2.

5.3.1 Wearables

Wearables were identified to require Access Control (including Person/Entity Authorization), Device Integrity and Transmission Security. These requirements were deemed important to ensure that attackers would not be able to get authorization to modify devices, camouflage as trusted devices and modify/capture transmission of data. Table 5.3 shows how the different protocol stacks are secured or vulnerable with the use of wearable devices.

From Table 5.3, we can see that Thread and IP-Smart are the only two protocol stacks which seem to fulfill the security requirements of the wearables use-case scenario. BLE can provide satisfactory device integrity if the appropriate commissioning method is used. A passkey or OoB method would require user input/interaction to pair two devices, ensuring that the devices communicating are known and trusted. There are no Access Control List specified for BLE, but with devices operating in a master-slave setting and on its own physical radio channel, the master device is the only device authorized to perform operations. By securing the master device from unauthorized access (for instance a smartphone acting as master, and ensuring that owner

	Access Control & Person/Entity Authorization	**Device Integrity**	**Transmission Security**
ZigBee	Ensured by an Access Control List and Mutual Symmetric-key Entity Authentication	Attacker can take over devices in network, compromising device integrity	Secured by a ZigBee specific protocol
Thread	Ensured by an Access Control List and DTLS Handshake for authentication	Ensured by DTLS Handshake	Secured by DTLS
Z-Wave	TLS Handshake for Authentication, but no known Access Control List implementation	TLS Handshake provides, but unknown how Authenticated Deployment works or if it provides integrity	Secured by TLS
BLE	No Access Control, LE Legacy Pairing for Authentication	Depending on commissioning method, could be exploited if Just Key is used	Secured by TLS/ DTLS if BLE over IPv6
WirelessHART	Ensured by an Access Control List and Join Key Confirmation for Authentication	Unknown whether the Join Key Confirmation is providing device integrity. Universal Join Key, would give every device same Join Key	Unknown how Transmission Security is achieved
IP-Smart	Ensured by an Access Control List and TLS/DTLS Handshake for Authentication	Ensured by DTLS Handshake	Secured by TLS/ DTLS

Table 5.3: Overview of Wearables security requirements

	Data Privacy	Data Authenticity	Device Authentication
ZigBee	AES128-CCM	TLS 1.2 provides data integrity	Mutual Symmetric-key Entity Authentication provides mutual authentication between devices
Thread	AES128	DTLS provides data integrity	DTLS Handshake ensures device authentication
Z-Wave	AES128	TLS 1.1 provides data integrity	Unknown how Authenticated Deployment ensures authentication of device
Bluetooth Low Energy	AES128-CCM	If BLE is enabled with IPv6, TLS or DTLS can be used to ensure data integrity	LE Legacy Pairing provides authentication if passkey or OoB are used to commission device
WirelessHART	AES128	WirelessHART uses MICs to ensure data integrity	Devices are authenticated based on a Join Key. A unique Join Key would provide authentication.
IP-Smart	AES128	TLS or DTLS provides data integrity	DTLS Handshake provides authentication

Table 5.4: Overview of Smart Home security requirements

is only one capable of accessing smartphone), BLE could prove to fulfill security requirements for wearables. A look at wearables found in the consumer market today also indicates that BLE is, as of now, one of the preferred solutions to enable wearable technology.

5.3.2 Smart Home

Smart Homes could potentially transfer personal data through the network. Requirements to ensure that data transferred through the network remains private was identified as important, and also that data transferred kept its integrity (Data Authenticity). Table 5.4 shows the mapping of how the protocol stacks fulfill the smart home security requirements.

As can be seen in Table 5.4, all of the protocol stacks provides relatively well-known solutions to fulfill the security requirements of the smart home. However, as was seen in Section 5.1 and

5.2, ZigBee has known vulnerabilities in its key exchange protocol which could compromise the network key used by AES. Resulting in a compromise of the data privacy of ZigBee. Z-Wave was also shown to be vulnerable to injection of rogue controllers in the network, enabling attackers to unlock door locks (from some specific vendors) even though data privacy and integrity is provided. With Thread and IP-Smart leaving the application layer of the protocol stack open for vendor-specific solutions, there is proof found that in order to fulfill the smart home requirements, the correct implementation of security features by vendors is required.

5.3.3 Industrial Internet of Things

In the industrial Internet of Things, the performance of the protocol stacks become more of an importance, than seen in the other use-cases. Table 5.5 shows how the different protocol stacks satisfy the requirements of IIoT. An interesting remark is WirelessHARTs implementation of 3 different roles for network and device management. This approach differs from the other protocol stacks, which puts the same roles into one device or 1 gateway and 1 coordinator/trust center.

Ultveit-Moe et al. [54] has pointed out how established IP protocols as TLS/SSL could provide problems in terms of performance for real-time information sharing. WirelessHART is also the only protocol stack which enables Time Division Multiple Access (TDMA) and time synchronization between devices, as seen in Table 5.6. By doing this, WirelessHART can offer better performance compared to the other protocol stacks offering CSMA techniques, by ensuring every device getting an allocated time slot for data transfer without collisions. From the information gathered ZigBee, Thread, and IP-Smart implements security measures which coincide with the requirements of IIoT. However, with the indications of TLS not fulfilling performance requirements and WirelessHART as the only protocol stack using TDMA, WirelessHART seems to be the only protocol stack of the ones compared which also fulfills performance requirements. More research into the protocol stacks would be required to determine further if some other than WirelessHART could be applicable for IIoT.

	Access Control	**Availability**	**Real-time Information Sharing with Confidentiality and Integrity**	**Device Managent**
ZigBee	Ensured by the use of an Access Control List	Remote connection to network through gateway	Implementation of ZigBee specific protocol	Handled by a Trust Center in the network
Thread	Ensured by the use of an Access Control List	Remote connection to network through gateway	Provided by DTLS	Handled by a Coordinator in the network
Z-Wave	Unknown if any Access Control is implemented	Remote connection to network through gateway	Provided by TLS 1.1	Handled by a Gateway Controller
BLE	No Access Control implemented	Master device must be in distance to slave device	Provided by TLS or DTLS with BLE over IPv6	Handled by the master device
WirelessHART	Ensured by the use of an Access Control List	Remote connection to network through gateway	Some WirelessHART specific solution, unknown how it is achieved	Handled by Gateway, Security Manager and Network Manager
IP-Smart	Ensured by the use of an Access Control List	Remote connection to network through gateway	Provided by TLS or DTLS	Handled by a PAN Coordinator

Table 5.5: Overview of IIoT security requirements

	Frequency	**Channels**	**Data Rates**	**Channel Access Method**
ZigBee	2.4GHz	16	250kbit/s	CSMA/CA
Thread	2.4GHz	16	250kbit/s	CSMA/CA
Z-Wave	Sub-1GHz	-	100kbit/s	CSMA/CA
BLE	2.4GHz	40	1Mbit/s	Frequency Hopping
WirelessHART	2.4GHz	16	250kbit/s	TDMA
IP-Smart	2.4GHz	16	250kbit/s	CSMA/CA

Table 5.6: Illustration of protocol stack specifications

5.4 Summary

Comparing the different protocol stacks show different approaches providing the different security services required by IoT. Some stacks rely heavily on established protocols to offer security services, while other stacks as ZigBee, has knowingly chosen to go ahead with their own specific protocols. Evidence from the research community shows the dangers of choosing own implementations of security services. ZigBee, Z-Wave, BLE, and WirelessHART has known weaknesses which possibly could lead to critical consequences if exploited. A common factor between all the stacks is that more research and investigation into the stacks are required to further identify vulnerabilities and to help the continuation of work to ensure security in IOT.

6 | Introduction to Contiki OS & IoT Simulation Test Cases

In this chapter, investigation into the protocols CoAP and MQTT is presented. As Thread, Z-Wave, BLE, and IP-smart all have an open application layer with support for implementation of the two protocols at hand. It was deemed an interesting case to compare the two protocols in terms of cost in power consumption vs the offered reliability of packet delivery between them. CoAP implements DTLS, while MQTT implements TLS, but both are subject to possible packet loss in transmission. CoAP could also provide communication with HTTP by use of a proxy, and thus it was interesting investigate if CoAP is a reasonable alternative to MQTT in IoT networks.

In order to test this, Contiki OS and its built in WSN simulator Cooja was used to perform real-life simulations of the two protocols. Section 6.1 introduces Contiki OS and the Cooja Simulator, as well as providing a simple tutorial in Cooja to show of it works, while Section 6.2 presents the actual simulations with setup and results.

6.1 Contiki OS

Contiki OS[1] is an open source operating system for the Internet of Things. It has an active community with contributors from known companies as Atmel, Cisco and Thingsquare, with many more working to continue the development of the OS. The OS is currently at version 3.0 and has support for several different low powered devices[2], in order to enable IoT functionality by offering IPv4, IPv6, 6LoWPAN, CoAP, etc. Contiki is today used in both commercial and non-

[1]Contiki OS Home Page: `http://www.contiki-os.org/`
[2]Hardware support list: `http://www.contiki-os.org/hardware.html`

commercial applications to offer services as industrial monitoring, smart city devices, and much more.

6.1.1 Cooja Simulator

Cooja simulator is a network simulator bundled with Contiki OS. It can be used to simulate small and/or large networks of Contiki nodes before compiling applications onto the hardware. It supports hardware level simulation, thus giving realistic simulation for the given application scenario. This gives the developer precise results for inspection, in order to optimize the application before compiled to hardware and put into real life use. To give the reader a greater understanding of how a Cooja simulation works, a Cooja "Hello World" tutorial is provided in the next subsection.

6.1.2 Hello World Tutorial for Cooja

This tutorial shows a "Hello World" example in Instant Contiki 3.0[3] run as virtual machine in VmWare Player 12[4]. The first step after launching the virtual machine is to open a terminal and run the following commands:

```
$ cd contiki/tools/cooja
$ ant run
```

This will start the Cooja simulator and open a window similar to what can be seen in Figure 6.1. When the Cooja simulator is up and running the following listed step-by-step procedure will create a simple Hello World scenario where the nodes will send a "Hello World" message to each other.

1. Create a new simulation by clicking *File -> New Simulation*
2. In the pop up window, give the simulation a name. For this example "Hello World" is entered as name and then click *Create*

[3]Instant Contiki is a virtual machine, providing a complete Contiki development environment. Download: https://sourceforge.net/projects/contiki/files/Instant%20Contiki/

[4]VmWare Player 12 can be downloaded and tested for free from: https://www.vmware.com/products/player

3. Add mote(s) to the simulation by clicking *Motes* -> *Add Motes* -> *Create new mote type*, this will give a list of different mote types. For this example choose *Sky mote*

4. If desired, a description for the mote can be added, for this example no explicit description is added. Then click *Browse* and navigate to */home/user/contiki/examples/hello-world/* and choose *hello-world.c* and click *Compile* and *Create* when compilation has finished

5. A new pop-up window will appear, with options of how many motes to add, and their positions. For this example 5 motes are added and random positioning is used.

6. In the *Simulation Control* panel, click *Start* to start the simulation.

In the *Network* panel, the user can click the *View* tab to get a list of different viewing options for the simulation. The *Output mote* panel will show the different motes being assigned its Rime address[5], MAC address, IPv6 address and will show how the motes start to broadcast a “Hello World” message.

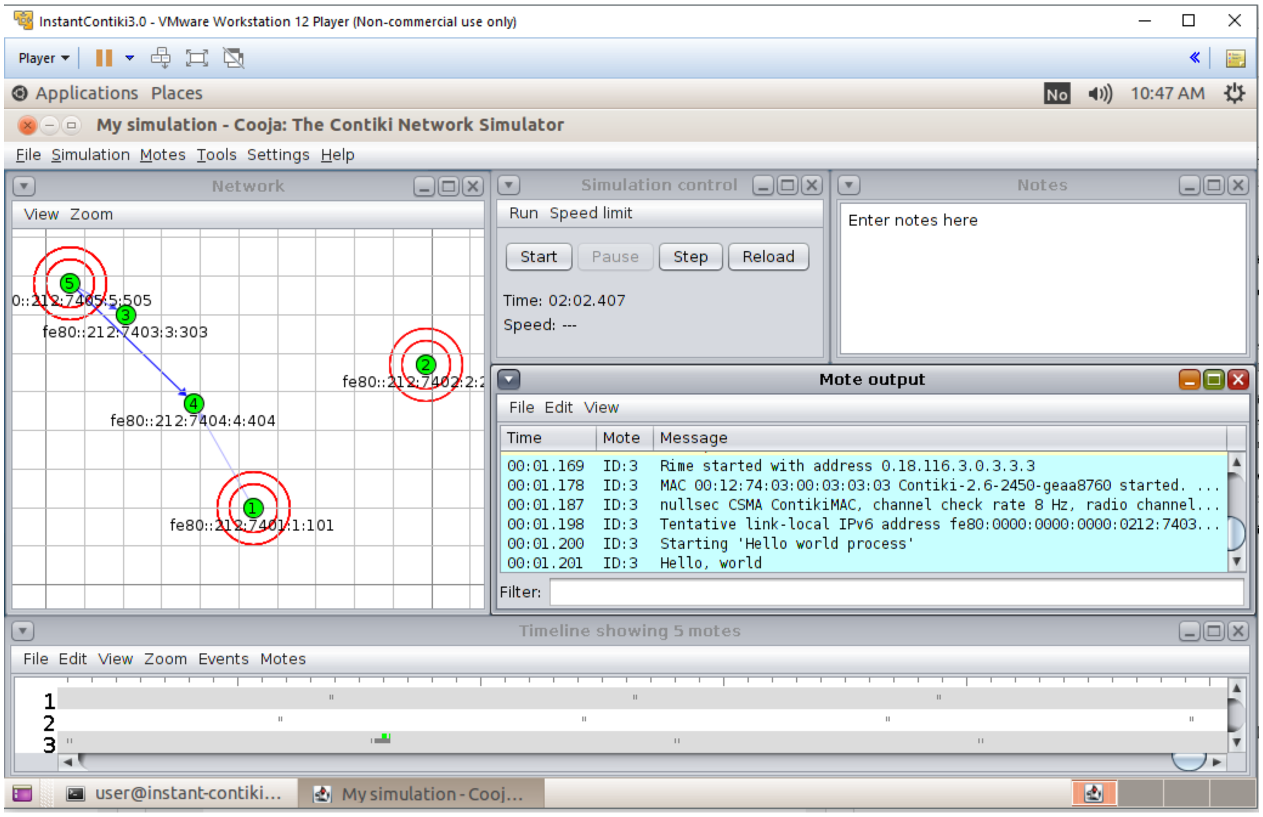
6.2 Simulation CoAP vs MQTT

An interesting feature to investigate further was identified as differences between the CoAP protocol and the MQTT protocol. CoAP offers DTLS, while MQTT uses TLS. Meaning a difference in terms of reliability of packet deliveries between devices. A look at differences in power consumption was deemed interesting to get some idea of the cost of adding extra reliability with DTLS in CoAP. This investigation would also be relevant for the authors proposed IP-Smart protocol stack, in order to check how the proposed protocols of IETF and IEEE match up with an already established protocol as MQTT.

6.2.1 Simulation Setup

Performing the simulations, the mentioned Contiki OS and Cooja Simulator was used. Instant Contiki 3.0, the virtual machine development environment of Contiki, was used together with VMWare Player 12. Benchmarking of the simulations was done on an ASUS U31SD laptop with

[5]Rime is a lightweight communication stack provided by Contiki OS



```
Time        Mote   Message
00:01.169   ID:3   Rime started with address 0.18.116.3.0.3.3.3
00:01.178   ID:3   MAC 00:12:74:03:00:03:03:03 Contiki-2.6-2450-geaa8760 started. ...
00:01.187   ID:3   nullsec CSMA ContikiMAC, channel check rate 8 Hz, radio channel...
00:01.198   ID:3   Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7403...
00:01.200   ID:3   Starting 'Hello world process'
00:01.201   ID:3   Hello, world
```

Figure 6.1: Illustration of the “Hello World” example in Cooja

the hardware specifications as found in table 6.1. In the simulations the Zolertia Z1[6] was used as the device, to ensure support for both MQTT and CoAP.

CPU:	Intel(R) Core(TM) I3-2310M 2.10 GHZ
RAM:	4.00 GB
HDD:	Samsung EVO840 SSD 256 GB
OS:	Windows 10 Pro x64

Table 6.1: HW configuration of Asus U31SD laptop used for testing

To investigate the differences in power consumption between CoAP and MQTT, and adding an extra layer of reliability with CSMA, the following test setups were benchmarked:

1. CoAP without CSMA
2. CoAP with CSMA
3. MQTT without CSMA

[6]Zolertia Z1: http://zolertia.io/z1

4. MQTT with CSMA

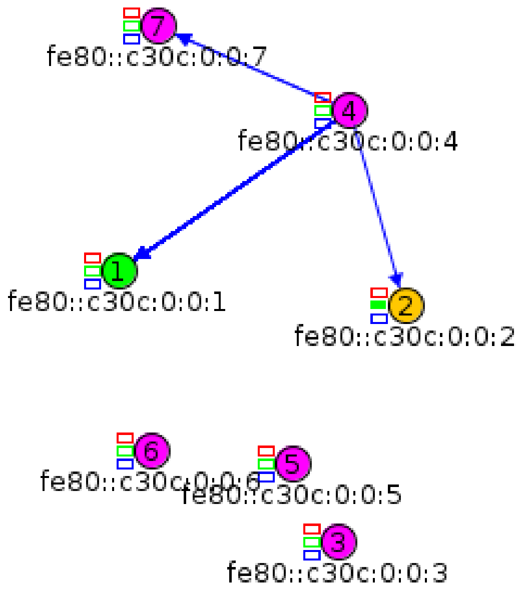


Figure 6.2: Illustration of physical setup of devices in CoAP simulation. Node 1: Border Router, Node 2: CoAP Server and Node 3-7: CoAP clients

In simulation #1 and #2, an 802.15.4 network was set up with a border router, a CoAP server and 5 CoAP Clients as shown in Figure 6.2. To interact with the CoAP devices to retrieve data and perform simple actions as turning LEDs on and off, the Copper(CU)[7] add-on for Firefox[8] was used, to implement a CoAP user-agent. This setup made use of already embedded libraries in Contiki (see Appendix B) to implement border router and CoAP server/client functionality. To enable/disable CSMA, configurations was made in the *project-conf.h* file, as seen in Appendix B.

[7]Source: https://addons.mozilla.org/en-US/firefox/addon/copper-270430/
[8]Source: https://www.mozilla.org/en-US/firefox/new/?utm_source=firefox-com&utm_medium=referral

Simulation #3 and #4 set up an 802.15.4 network with a border router and 5 MQTT clients, as seen in Figure 6.3. Also, the Mosquitto[9] MQTT broker was used operating as the user-agent. As with the CoAP simulations, embedded libraries of Contiki was used to implement the border router and MQTT clients, and configurations were done in the *project-conf.h* file to enable/disable CSMA (See Appendix B).

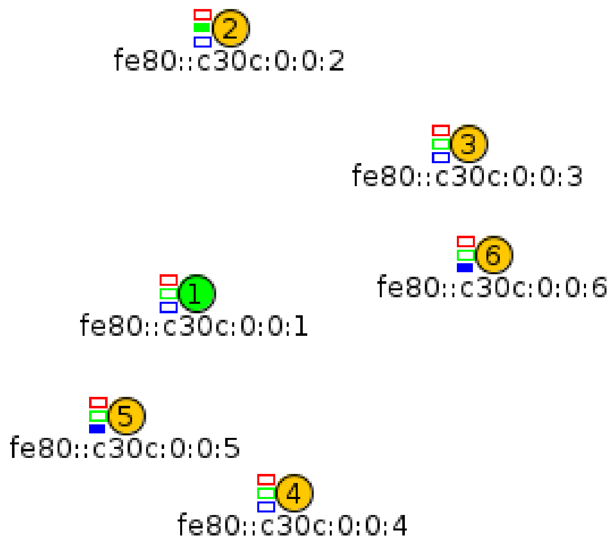


Figure 6.3: Illustration of the physical setup of devices in MQTT simulation. Node 1: Border Router, Node 2-6: MQTT clients

6.2.2 Simulation Results

Results of the simulations are presented in Table 6.2, 6.3, 6.4 and 6.5. A Contiki OS tool named PowerTracker was used to measure the Radio Duty Cycle[10] of each device during the simulations. Radio On is total time device is active, Radio TX is the time of radio transferring data and Radio RX the is time of radio receiving data.

By looking at the results, evidence of CoAP requiring a device to be in an active state for

[9]Source: http://mosquitto.org/

[10]A Radio Duty Cycle is the percentage of time in which the system is active

longer than with MQTT can be seen. Comparing *CoAP Client #1* of Table 6.2 and *MQTT Client #1* of Table 6.4, shows an increase of Radio On by 0.57% when using CoAP. A device which is in an active state for longer equals drawing more power. However, looking at the numbers overall should suggest no critical increase of RDC by using CoAP.

Enabling CSMA shows no significant difference of RDC in the CoAP simulations, while in MQTT a slight increase of RDC can be found, comparing *MQTT Client #3* in Table 6.4 and 6.5. It must be stated, that these simulations are done in Contiki OS, using the OS specific MAC layer implementations. This means that the implementations of how a device goes from sleeping to active, and vice-versa, could give other results with other IoT-specific operating systems and different implementations. Overall, the results are advised to be interpreted as an indication of CoAP being a reasonable alternative to MQTT, which would increase the reliability of packet delivery in lossy networks with the implementation of DTLS.

Mote	**Radio On(%)**	**Radio TX(%)**	**Radio RX(%)**
Border Router	99.86	0.24	2.77
CoAP Server	0.75	0.17	0.07
CoAP Client #1	1.50	0.81	0.03
CoAP Client #2	1.44	0.79	0.02
CoAP Client #3	1.42	0.73	0.05
CoAP Client #4	1.44	0.75	0.04
CoAP Client #5	1.04	0.48	0.03

Table 6.2: PowerTracker results CoAP without CSMA

Mote	**Radio On(%)**	**Radio TX(%)**	**Radio RX(%)**
Border Router	99.86	0.29	3.15
CoAP Server	0.76	0.17	0.07
CoAP Client #1	1.47	0.79	0.04
CoAP Client #2	1.45	0.79	0.03
CoAP Client #3	1.50	0.78	0.05
CoAP Client #4	1.50	0.80	0.04
CoAP Client #5	1.46	0.80	0.03

Table 6.3: PowerTracker results CoAP with CSMA

Mote	**Radio On(%)**	**Radio TX(%)**	**Radio RX(%)**
Border Router	97.91	2.88	0.37
MQTT Client #1	0.93	0.12	0.17
MQTT Client #2	0.76	0.12	0.12
MQTT Client #3	0.73	0.09	0.12
MQTT Client #4	0.83	0.17	0.13
MQTT Client #5	0.74	0.15	0.08

Table 6.4: PowerTracker results MQTT without CSMA

Mote	**Radio On(%)**	**Radio TX(%)**	**Radio RX(%)**
Border Router	98.58	4.58	0.69
MQTT Client #1	0.99	0.21	0.17
MQTT Client #2	1.01	0.16	0.21
MQTT Client #3	1.10	0.25	0.21
MQTT Client #4	0.97	0.14	0.22
MQTT Client #5	0.87	0.19	0.11

Table 6.5: PowerTracker results MQTT with CSMA

7 | Conclusion

7.1 Recommendations

7.1.1 Implementations at the Application Layer

As seen in Chapter 4, Thread, Z-Wave, BLE, and IP-Smart have left the application layer open for vendor-specific solutions. Protocols as CoAP and MQTT are supported by Thread, BLE, and IP-Smart which are well-known protocols, using existing secure solutions as TLS and DTLS. Chapter 6 shows that CoAP is a reasonable alternative to MQTT, therefore it is recommended to ***use CoAP to add reliability over lossy networks***. Many IoT products on the market today as Philips Hue[1], NEST Thermostat[2], and Apple Watch[3] are just some examples of devices connecting to a smartphone app. Using a smartphone app to communicate with the devices could potentially open up new attack surfaces for devices. As this has not been the main focus of this thesis, the author still believes it to be important to point out the importance of following security guidelines as OWASP Mobile Security Project[4] or similar, when developing smartphones app to use with IoT devices, to ensure no unwanted tampering with devices caused by vulnerable apps.

7.1.2 Implementations for Secure Commissioning

In the commissioning process of a new device, several vulnerabilities was pointed out in Chapter 5. To ensure secure and trusted commissioning of devices ***using unique Join Keys in WirelessHART***, ***using passkey or some OoB method for authentication in BLE*** and ***only using high***

[1]Philips Hue: `http://www2.meethue.com`
[2]NEST Thermostat: `https://nest.com/`
[3]Apple Watch: `http://www.apple.com/watch/`
[4]OWASP Mobile Security Project: `https://www.owasp.org/index.php/OWASP_Mobile_Security_Project`

security mode of ZigBee are recommended to mitigate the risk of exploitation of the documented vulnerabilities. Other findings suggests that ***if deemed necessary: devices must be protected from physical tampering and implement secure transmission of joining data*** such as pre-configured joining keys. If the proposed IP-Smart protocol stack is to be used, an implementation of a secure commissioning process is left to the vendors. Deciding what would be an appropriate solution would be entirely use-case dependent, and the only guideline this thesis can give is to implement at least some sort of OoB method or passkey/QR-code solution.

7.2 Future of IoT

With the estimated growth of smart devices exceeding at least 20 billion devices by 2020 [5] [6] [7] there is no doubt that the *Internet of Things* will affect our way of living in the years to come. This thesis has presented some of the enabling technologies and protocol stacks of the IoT, but it only represents a small amount of all the different solutions manufactured to enable the IoT in different use-case scenarios. In order to fully realize the potential of the IoT, interoperability between products becomes the main focus and leading initiatives as ZigBee and Thread have announced their cooperation on making each other's solutions interoperable[8]. The WiFi Alliance has also announced its Wi-Fi HaLow [9] to enable the commonly used 802.11 technology for low powered devices in the IoT. This will further the work to enable interoperability with already existing technologies and solutions in a hope of working towards a common standard for the IoT. However, with a magnitude of different stakeholders trying to claim their share of the potential in the IoT market, the future is hard to predict.

[5]Business Insider: http://uk.businessinsider.com/top-internet-of-things-trends-2016-1?r=US&IR=T

[6]McKinnsey: http://www.mckinsey.com/industries/high-tech/our-insights/the-internet-of-things-sizing-up-the-opportunity

[7]Cisco: https://newsroom.cisco.com/press-release-content?articleId=1621819

[8]ZigBee & Thread Cooperation: http://www.zigbee.org/zigbee-alliance-creating-end-to-end-iot-product-development-solution-that-brings/

[9]WiFi Alliance: http://www.wi-fi.org/discover-wi-fi/wi-fi-halow

7.3 Final Remarks

In this thesis Internet of Things has been presented with some of the use-case areas and benefits it provides, but also the requirements that come with it in terms of constrained devices and security. A proposed model, inspired by the OSI model, was presented to suggest a way of standardizing future solutions in IoT. There are many different standards trying to establish themselves in the IoT market, each with their own proposed solution to solve the requirements of IoT. ZigBee, Thread, Z-Wave, BLE, WirelessHART, and IP-Smart has been presented to investigate the different solutions and how they measure up in terms of security and for use in different use-case areas. Standards providing their own set of protocols to offer different security services has been shown to contain weaknesses versus using more established and recognized services. There is, however, a lack of work done in the research community on different standards in IoT, and with the predicted rapid growth of IoT devices, a need for more focused research into IoT is off utmost need. The suggested IP-Smart protocol stack has been shown to be a promising approach for IoT, but there is still uncertainties in terms of best approach for key exchange, and implementation on the application layer is left open for OEMs. CoAP vs MQTT showed indications of CoAP being a reasonable option to the well-established MQTT protocol, in order to use DTLS and UDP setting up an IoT network. IoT is still in the beginning of its lifecycle and will only continue to grow and be a more important part of several aspects of our lives. There is great potential in IoT to enrich our lives and create economic growth, but as has been shown there are still pitfalls to watch out for, in order to realize the vision that is the Internet of Things.

A | Acronyms

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks

ACL Access Control List

AES Advanced Encryption Standard

APKES Adaptable Pairwise Key Establishment Scheme

CoAP Constrained Application Protocol

CSMA Carrier Sense Multiple Access

CSMA-CA Carrier Sense Multiple Access - Collision Avoidance

DLPDU Data Link Protocol Data Unit

DTLS Datagram Transport Layer Security

ECDH Elliptic Curve Diffie-Hellman

EC-JPAKE Elliptic Curve Juggling-Password Authenticated Key Exchange

FFD Full Function Device

GSM GSM Association

HART Highway Addressable Remote Transducer

HMAC Hash-based Message Authentication Code

HTTP Hypertext Transfer Protocol

ICS Industrial Control Systems

IoT Internet of Things

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

LE Low Energy

LR-WPAN Low Rate Wireless Personal Area Network

LTK Long Term Key

M2M Machine-to-Machine

MitM Man-in-the-Middle

MAC Media Access Control

MLE Mesh Link Establishment

NFC Near Field Communication

NIZK Non-Interactive Zero Knowledge

OEM Original Equipment Manufacturer

OoB Out-of-Band

PAN Personal Area Network

P2P Peer-to-Peer

QoS Quality of Service

RDC Radio Duty Cycle

REED Router-eligible End Device

RF Radio Frequency

RFD	Reduced Function Device
STK	Short Term Key
SKKE	Symmetric-key Key Establishment
TDMA	Time Division Multiple Access
TK	Temporary Key
TLS	Transport Layer Security
WSN	Wireless Sensor Networks

B | Additional Information

B.1 Libraries

List of libraries from Contiki OS, used to create CoAP and MQTT simulations:

- MQTT-demo
- ER-Rest Client
- ER-Rest Server
- RPL Border Router

B.2 Enable/disable CSMA

project-conf.h configurations to enable/disable CSMA in ER-Rest Client/Server and MQTT-demo:

```
//Enable CSMA
#indef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC csma_driver

//Disable CSMA
#indef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC nullmac_driver
```

B.3 Z1 Platform Configuration

Configuration of Z1 device (file: contiki/platform/z1/contiki-conf.h) for CoAP simulation), preventing a buffer overflow.

```
//Changed from 140 to 240
#UIP_CONF_BUFFER_SIZE 240
```

B.4 MQTT Simulation Configuration

Modification of MQTT-demo library (contiki/examples/cc2538dk/mqtt-demo/mqtt-demo.c, to use it with Z1.

```
#include "dev/sht25.h" /*Replacing: "dev/cc2538-sensors.h"*/

#define APP_BUFFER_SIZE 256/*512*/

static void
publish(void)
{
  .
  int16_t value; /*added for MQTT simulation*/
  .
  .
  /* Reconfigure sensor readings to ensure Z1 sensors are used correctly*/
  value = sht25.value(SHT25_VAL_TEMP);
  len = snprintf(buf_ptr, remaining, ",\"SHT25 Temp (mC)\":%d", value);
  /*Replaced: len = snprintf(buf_ptr, remaining, ",\"On-Chip Temp (mC)\":%d",
                cc2538_temp_sensor.value(CC2538_SENSORS_VALUE_TYPE_CONVERTED));*/
  .
  .
  .
  value = sht25.value(SHT25_VAL_HUM);
  len = snprintf(buf_ptr, remaining, ",\"Humidity (RH)\":%d", value);
  /*Replaced: len = snprintf(buf_ptr, remaining, ",\"VDD3 (mV)\":%d",
               vdd3_sensor.value(CC2538_SENSORS_VALUE_TYPE_CONVERTED));*/
  .
  .
}
```

Bibliography

[1] Aijaz, A. and Aghvami, A. H. (2015). Cognitive Machine-to-Machine Communications for Internet-of-Things: A Protocol Stack Perspective. *IEEE Internet of Things Journal*, 2(2):103–112.

[2] Al-Fuqaha, A. I., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347–2376.

[3] Arias, O., Wurm, J., Hoang, K., and Jin, Y. (2015). Privacy and Security in Internet of Things and Wearable Devices. *IEEE Trans. Multi-Scale Computing Systems*, 1(2):99–109.

[4] ARM (2014). What the Internet of Things (IoT) Needs to Become a Reality. White paper, http://www.nxp.com/files/32bit/doc/white_paper/INTOTHNGSWP.pdf. Accessed: 07.03.16.

[5] Bayou, L., Espes, D., Cuppens-Boulahia, N., and Cuppens, F. (2015). Security Issue of WirelessHART Based SCADA Systems. In *Risks and Security of Internet and Systems - 10th International Conference, CRiSIS 2015, Mytilene, Lesbos Island, Greece, July 20-22, 2015, Revised Selected Papers*, pages 225–241.

[6] Bluetooth Special Internet Group (2014). Specification of the Bluetooth System. Core Specification 4.2, Bluetooth SIG.

[7] Cobb, J., Rotvold, E., and Potter, J. (2010). WirelessHART Security Overview. http://www.hcf-files.com/webasyst/published/DD/2.0/file_link.php?sl=8fc5017b478acccd8f2806675669d68c&DB_KEY=V0VCRklMRVM%3D. Accessed: 24.05.16.

[8] Deering, S. and Hinden, R. (1998). Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, RFC Editor.

[9] Denning, T., Kohno, T., and Levy, H. M. (2013). Computer Security and the Modern Home. *Commun. ACM*, 56(1):94–103.

[10] DU Jiang and CHAO Shi Wei (2010). A Study of Information Security for M2M of IoT. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5579563. Accessed: 07.03.16.

[11] Elkhodr, M., Shahrestani, S. A., and Cheung, H. (2016). The Internet of Things: New Interoperability, Management and Security Challenges. *CoRR*, abs/1604.04824.

[12] F. Hao, E. (2016). J-PAKE: Password Authenticated Key Exchange by Juggling. http://tools.ietf.org/html/draft-hao-jpake-03. Accessed: 12.06.16.

[13] FocalPoint Group (2013). M2M White Paper: The Growth of Device Connectivity. White paper, https://www.qualcomm.com/documents/m2m-white-paper-growth-device-connectivity. Accessed: 07.03.16.

[14] Fouladi, B. and Ghanoun, S. (2013). Security Evaluation of the Z-Wave Wireless Protocol. https://www.sensepost.com/cms/resources/conferences/2013/bh_zwave/Security%20Evaluation%20of%20Z-Wave_WP.pdf. Accessed: 05.06.16.

[15] Fuller, J. D. and Ramsey, B. W. P. (2015). Rogue Z-Wave Controllers: A Persistent Attack Channel. In *40th IEEE Local Computer Networks Conference Workshops, LCN Workshops 2015, Clearwater Beach, FL, USA, October 26-29, 2015*, pages 734–741.

[16] Garcia-Morchon, O., Kumar, S., Hummen, R., and Struik, R. (2013). Security Considerations in the IP-based Internet of Things. https://tools.ietf.org/pdf/draft-garcia-core-security-06.pdf. Accessed 12.06.2016.

[17] GSMA (2014). Understanding the Internet of Things (IoT). White paper, http://www.gsma.com/connectedliving/wp-content/uploads/2014/08/cl_iot_wp_07_14.pdf. Accessed: 07.03.16.

[18] Halperin, D., Heydt-Benjamin, T. S., Ransford, B., Clark, S. S., Defend, B., Morgan, W., Fu, K., Kohno, T., and Maisel, W. H. (2008). Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 129–142.

[19] HART Communication Foundation (2013). IEC 62591 WirelessHART System Engineering Guide. http://www.hcf-files.com/webasyst/published/DD/2.0/file_link.php?sl=a7cef97803a279607b7da3fae7648230&DB_KEY=V0VCRklMRVM%3D. Accessed: 24.05.16.

[20] IEEE (2011). IEEE 802.15.4-LOCAL AND METROPOLITAN AREA NETWORK STANDARDS. Standards publication, https://standards.ieee.org/getieee802/download/802.15.4-2011.pdf. Accessed: 08.03.16.

[21] IEEE Standards Association. Guidelines for 64-bit Global Identifier (EUI-64). https://standards.ieee.org/develop/regauth/tut/eui64.pdf. Accessed: 10.06.16.

[22] Ilia, P., Oikonomou, G. C., and Tryfonas, T. (2013). Cryptographic Key Exchange in IPv6-Based Low Power, Lossy Networks. In *Information Security Theory and Practice. Security of Mobile and Cyber-Physical Systems, 7th IFIP WG 11.2 International Workshop, WISTP 2013, Heraklion, Greece, May 28-30, 2013. Proceedings*, pages 34–49.

[23] Internet Society (2015). The Internet of Things: An Overview. White paper, http://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151221-en.pdf. Accessed: 08.03.16.

[24] Jacobsson, A., Boldt, M., and Carlsson, B. (2016). A Risk Analysis of a Smart Home Automation System. *Future Generation Comp. Syst.*, 56:719–733.

[25] Jose, A. C. and Malekian, R. (2015). Smart Home Automation Security: A Literature Review. *Smart CR*, 5(3):269–285.

[26] Kargl, F., van der Heijden, R. W., König, H., Valdes, A., and Dacier, M. (2014). Insights on the Security and Dependability of Industrial Control Systems. *IEEE Security & Privacy*, 12(6):75–78.

[27] Kelsey, R. (2015). Mesh Link Establishment. Internet-Draft draft-kelsey-intarea-mesh-link-establishment-06, Internet Engineering Task Force. Accessed: 12.06.16.

[28] Kevin Ashton (2009). That 'Internet of Things' Thing. RFID Journal, http://www.rfidjournal.com/articles/view?4986. Accessed: 07.03.16.

[29] Krentz, K., Rafiee, H., and Meinel, C. (2013). 6lowpan security: adding compromise resilience to the 802.15.4 security sublayer. In *Proceedings of the International Workshop on Adaptive Security, ASPI@UbiComp 2013, Zurich, Switzerland, September 8, 2013*, pages 1:1–1:10.

[30] Kumar, S. A., Vealey, T., and Srivastava, H. (2016). Security in Internet of Things: Challenges, Solutions and Future Directions. In *49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016*, pages 5772–5781.

[31] Lohmann, G. (2010). WirelessHART Device Types - Gateways. http://www.hcf-files.com/webasyst/published/DD/html/scripts/getfolderfile_zoho.php?DL_ID=MTA4NQ%3D%3D&ID=80b897344a6663d70b5a00c37408ea60&DB_KEY=V0VCRklMRVM%3D. Accessed: 11.05.16.

[32] Montenegro, G., Kushalnagar, N., Hui, J., Culler, D., Microsoft, Intel, and Arch Rock Corp (2007). Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, RFC Editor.

[33] National Instruments (2012). What Is a Wireless Sensor Network? http://www.ni.com/white-paper/7142/en/. Accessed: 09.03.16.

[34] Nieminen, J., Savolainen, T., Isomaki, M., Nokia, Patil, B., AT&T, Shelby, Z., ARM, and Gomez, C. (2015). IPv6 over BLUETOOTH(R) Low Energy. Internet-Draft RFC7668, IETF.

[35] NXP Semiconductors (2015). ZigBee 3.0 - Facilitating the 'Internet of Things'. http://cache.nxp.com/documents/other/75017677.pdf. Accessed: 03.05.16.

[36] Oasis (2014). MQTT Version 3.1.1. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf. Accessed: 05.06.16.

[37] O'Brien, G., Pleasant, B., Bowers, C., Wang, S., Zheng, K., Kamke, K., Lesser, N., and Kauffman, L. (2015). SECURING ELECTRONIC HEALTH RECORDS ON MOBILE DEVICES - Approach, Architecture, and Security Characteristics. https://nccoe.nist.gov/sites/default/files/nccoe/NIST_SP1800-1b_Draft_HIT_Mobile_Approach-Arch-Security.pdf. Accessed: 16.05.16.

[38] Padgette, J., Scarfone, K., and Chen, L. (2012). Guide to Bluetooth Security. http://csrc.nist.gov/publications/nistpubs/800-121-rev1/sp800-121_rev1.pdf. Accessed: 10.05.16.

[39] Palattella, M. R., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L. A., Boggia, G., and Dohler, M. (2013). Standardized Protocol Stack for the Internet of (Important) Things. *IEEE Communications Surveys and Tutorials*, 15(3):1389–1406.

[40] Raza, S., Slabbert, A., Voigt, T., and Landernäs, K. (2009). Security considerations for the wirelesshart protocol. In *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2009, September 22-25, 2008, Palma de Mallorca, Spain*, pages 1–8.

[41] Razouk, W., Crosby, G. V., and Sekkaki, A. (2014). New security approach for zigbee weaknesses. In *The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2014)/ The 4th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2014)/ Affiliated Workshops, September 22-25, 2014, Halifax, Nova Scotia, Canada*, pages 376–381.

[42] Rescorla, E., Modadugu, N., RTFM, Inc., and Stanford University (2006). Datagram Transport Layer Security. RFC 4347, RFC Editor.

[43] Ryan, M. (2013). Bluetooth: With Low Energy Comes Low Security. In *7th USENIX Workshop on Offensive Technologies, WOOT '13, Washington, D.C., USA, August 13, 2013*.

[44] Sadeghi, A., Wachsmann, C., and Waidner, M. (2015). Security and Privacy Challenges in Industrial Internet of Things. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pages 54:1–54:6.

[45] Sastry, N. and Wagner, D. (2004). Security considerations for IEEE 802.15.4 networks. In *Proceedings of the 2004 ACM Workshop on Wireless Security, Philadelphia, PA, USA, October 1, 2004*, pages 32–42.

[46] Shelby, Z., ARM, Hartke, K., and Bormann, C. (2014). The Constrained Application Protocol (CoAP). RFC 7252, RFC Editor.

[47] Sicari, S., Rizzardi, A., Grieco, L. A., and Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146–164.

[48] Stallings, W. (2014). *Cryptography and Network Security - Principles and Practice*, chapter 17, pages 544–562. Pearson Education, 6. edition.

[49] Thread Group (2015a). Thread Border Routers. White Paper 2.0, Thread Group.

[50] Thread Group (2015b). Thread Commissioning. White Paper 2.0, Thread Group.

[51] Thread Group (2015c). Thread Stack Fundamentals. White Paper 2.0, Thread Group.

[52] Thread Group (2015d). Thread Technical Overview.

[53] Trappe, W. and Washington, L. C. (2006). *Introduction to Cryptography with Coding Theory*, chapter 16.5, pages 363–366. Pearson Education, 2. edition.

[54] Ulltveit-Moe, N., Nergaard, H., Erdödi, L., Gjøsæter, T., Kolstad, E., and Berg, P. (2016). Secure Information Sharing in an Industrial Internet of Things. *CoRR*, abs/1601.04301.

[55] Vidgren, N., Haataja, K., Patino-Andres, J. L., Ramirez-Sanchis, J. J., and Toivanen, P. (2013). Security threats in zigbee-enabled systems: Vulnerability evaluation, practical experiments, countermeasures, and lessons learned. In *46th Hawaii International Conference on System Sciences, HICSS 2013, Wailea, HI, USA, January 7-10, 2013*, pages 5132–5138.

[56] Want, R., Schilit, B. N., and Jenson, S. (2015). Enabling the Internet of Things. *IEEE Computer*, 48(1):28–35.

[57] ZigBee Alliance (2012). ZigBee Specification. http://www.zigbee.org/download/standards-zigbee-specification/. Accessed: 03.05.2016.

[58] Zillner, T. (2015). ZigBee Exploited - The good, the bad and the ugly. https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly-wp.pdf. Accessed: 11.05.16.