José Miguel Camacho Tavares

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Internet of Things: Security and Organization

Dissertação apresentada para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores, pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.

Orientador: Luís Manuel Camarinha de Matos, Professor Doutor, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

Júri:

Presidente:	Prof. Doutor José Manuel Matos Ribeiro da Fonseca, FCT/UNL
Arguente:	Prof. Doutora Patrícia Alexandra Pires Macedo, EST/IPS
Vogal:	Prof. Doutor Luís Manuel Camarinha de Matos, FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2015

Internet of Things: Security and Organization

To my parents

Acknowledgements

To my mother, father and sister a big thank you for the endless support throughout my years at the university, regardless of the distance.

I would like to thank my supervisor, Professor Luis M. Camarinha-Matos, for the sympathy, guidance, and help during the making of this thesis.

To Evan Parks, a big thank you for the help in finding and correcting grammatical errors in this thesis.

Finally a special thanks to my girlfriend, Dorota Abel, for the company, support, patience, and motivation throughout this thesis and last years at the university.

Resumo

A Internet de hoje é principalmente focada a humanos e os dispositivos conectados correspondem a technologia bem conhecida (e.g. computadores, telemóveis). A Internet das Coisas altera esta situação ao introduzir uma vasta variedade de novos objectos conectáveis à Internet, por forma a melhorar o dia a dia de pessoas e empresas com a criação de novos serviços e induzindo inovações nestes ambientes. Contudo as soluções de segurança aplicadas aos dispositivos previamente conectados não são suficientes para estes novos objectos. Isto deve-se à sua miniaturização que leva a alguns constrangimentos em termos de poder computacional, memória, armazenamento e bateria, o que introduz novos desafios em termos de implementação de soluções de segurança. Para além do estabelecimento de comunicações seguras, os dados recolhidos destes objectos também levantam algumas ameaças à privacidade. Como esses dados são processados, armazenados e ou compartilhados ainda não é claro. Com o número crescente de objectos a tornarem-se parte da Internet o risco também aumenta se medidas apropriadas de segurança e privacidade não forem implementadas desde início na Internet das Coisas. Este trabalho foi desenvolvido por forma a identificar vulnerabilidades de segurança e privacidade associadas à implementação da Internet das Coisas. Esta dissertação começa por descrever o conceito da Internet das Coisas, seguido de uma descrição do actual estado de segurança dos produtos para a Internet das Coisas. As tecnologias que permitirão o crescimento da Internet das Coisas e os desafios de segurança, requisitos, ameaças e contramedidas são então sumarizados nesta dissertação. No final uma avaliação experimental da segurança implementada num dispositivo presente no mercado foi feita, e uma proposta mais robusta para a segurança e privacidade do dispositivo foi desenvolvida. Este trabalho foi seguido da adaptação de um software por forma a organizar e gerir os dispositivos heterogéneos da Internet das Coisas que farão parte da casa inteligente. Finalmente algumas conclusões e direcções para trabalho futuro são apresentadas.

Palavras Chave: Internet das Coisas, Segurança.

Abstract

The Internet today is mainly human-centric and many of the connected devices are well known technology (e.g. desktops, laptops, mobile phones). The Internet of Things (IoT) changes this situation by introducing a wide variety of new objects to the traditional Internet, so as to improve the daily lives of people and business, by enhancing new services and triggering innovations into these environments. However, the security solutions built around previous connected technologies is not completely sufficient for these new connected objects. This is due to their miniaturization they have many constraints in terms of computational power, memory, storage, and battery lifetime, which introduces new challenges in terms of implementing security solutions. Apart from the establishment of secure communications, the data collected from these devices also raises some privacy threats; how these data are processed, stored and shared, is still not clear. As more objects become Internet-enabled the threat will only increase if no measures are put in place during the early stages of the Internet of Things. This work was developed to identify potential security and privacy vulnerabilities associated with the current and future Internet of Things. This thesis starts by describing the concept of the Internet of Things followed by a description of the current state of security approaches for the IoT products. The technologies that will enable the IoT growth and the security challenges, requirements, threats, and countermeasures are thus summarized in this thesis. At the end an experimental assessment of the security implementation of a device present in the current market was performed and a more robust proposal for the security and privacy of the device was developed. The work was then followed by the implementation of an environment that will help organize and manage the heterogeneous IoT devices that will be part of a smart home. Finally some conclusions and directions for future work are presented.

Keywords: Internet of Things, Security.

Acronyms

ACK *Acknowledgement*

AMQP *Advanced Message Queuing Protocol*

CoAP *Constrained Application Protocol*

DDS *Data Distribution Service*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hypertext Transfer Protocol Secure*

IP *Internet Protocol*

IoT *Internet of Things*

MAC *Media Access Control*

MQTT *Message Queuing Telemetry Transport*

M2M *Machine to Machine*

QoS *Quality of Service*

TLS *Transport Layer Security*

TCP *Transmission Control Protocol*

UDP *User Datagram Protocol*

XMPP *Extensible Messaging and Presence Protocol*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

With the paradigm of the Internet of Things rapidly entering our lives it also raises new challenges in security, privacy, and in technical terms. A world of pervasive computing, populated by Internet-enabled objects that can share sensed data and/or physically act on their environment is vulnerable to the current and novel security and privacy threats that the cyber world currently faces.

"A world full of smart objects holds enormous promise for improving business processes and people's lives, but it also comes with serious threats and technical challenges that must be overcome"(Whitmore et al., 2014).

According to a recent HP security report (HP, 2014b), it was found that 70% of the IoT products did not encrypt communications to the Internet and local network. Therefore, there are growing concerns about the security and privacy measures adopted by the IoT manufacturers.In this context, this dissertation focuses on identifying those threats, challenges, and possible solutions.

1.2 Approach

This thesis focuses on providing the reader with an understanding of the security issues that the IoT raises and proposing some solutions to those issues. It thus introduces the IoT concept and discusses the security vulnerabilities a current IoT product might have. The technologies that will help the IoT grow are described and the tools with which the IoT devices can be organized and secured are presented. The security threats as well as possible solutions are first described theoretically and then put into practice through the analysis of the security measures introduced in a IoT product and implementation of a tool that might help integrating and securing the different devices in one single framework. To prove the capabilities of this approach to integrate different devices, a prototype was implemented with different technologies and integrated with this tool. This solution is mainly focused on a smart home scenario.

1.3 Structure

In Chapter 2, there is an attempt to guide the reader to understand the potential of the IoT as well the new vulnerabilities the concept raises.

Chapter 3 identifies the technologies being developed to enable a cooperative, intelligent, and secure Internet of Things. In Chapter 4 an analysis of the security threats and solutions is done.

In Chapter 5 describes an experiment performed to show the vulnerabilities that the IoT products may currently carry and propose a solution to secure and organize the IoT devices in the context of the Smart Home is proposed.

Chapter 6 presents the final conclusions and directions for future work.

Chapter 2

State of the Art

"At the present time, IoT is an extensively used term. News about products being manufactured, researched or announced is becoming increasingly common. Meanwhile, connecting one or a few embedded systems or smartphones to the Internet is not sufficient to achieve a true Internet of Things. Before tackling any other issues it is necessary, first of all, to define what exactly is this Internet of Things" (Mendes, 2013).

2.1 What is the Internet of Things?

The term 'Internet of Things' was firstly introduced in 1999 by Kevin Ashton (Ashton, 2009) in a presentation made at Procter & Gamble (P& G), where he was discussing the new idea of linking RFID technology in the company's supply chain to the Internet.

"We need to empower computers with their own means of gathering information, so they can see, hear and smell the world for themselves, in all its random glory. RFID and sensor technology enable computers to observe, identify and understand the world - without the limitations of human-entered data. [...] The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so"(Ashton, 2009).

The Internet of Things since then has gained a lot of attention from different areas from the academia to the industry. The main idea behind the Internet of Things is to connect the everyday-objects or things (e.g. cars, tables, doors, lights, etc) to the Internet. Making

a useful use of the characteristics of these objects (i.e. sensing, actuating, traceability, interoperability, etc) so as to improve everyday-work of people and business.

Acoordingly to (Perera et al., 2014), the Internet of Things can be seen as a *"world where all the objects (...) around us are connected to the Internet and communicate with each other with minimum human intervention. The ultimate goal is to create 'a better world for human beings', where objects around us know what we like, what we want, and what we need and act accordingly without explicit instructions"*.

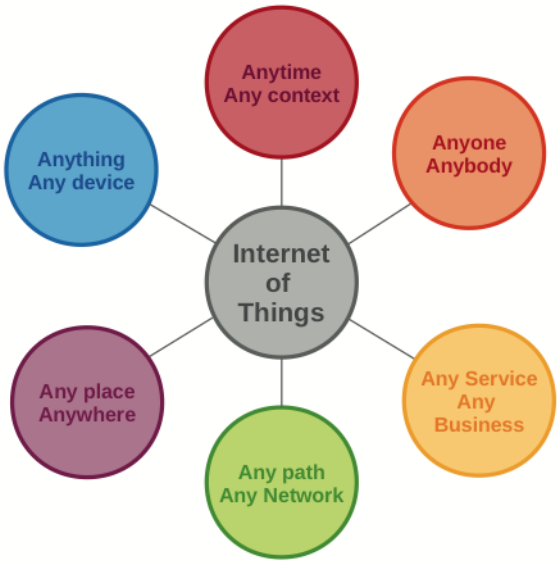


Figure 2.1: Definition of the Internet of Things (Perera et al., 2014)

Another definition by (Vermesan et al., 2011), states that *"The Internet of Things could allow people and things to be connected Any-time, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service"*.

The Internet of Things is a vision which is under development where everybody is trying to interpret IoT in respect to their needs (Singh et al., 2014). The present vision involves sensor base data collection, data management, data mining, remote and autonomous actuators, the World Wide Web, and the hardware that makes this possible.

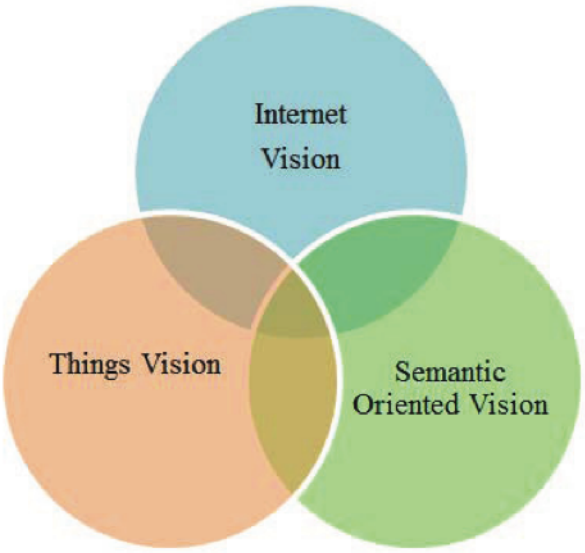


Figure 2.2: Three main visions of IoT (Singh et al., 2014)

The authors in (Atzori et al., 2010), proposed three particular visions (represented in Figure 2.2) that all together contribute to the IoT paradigm:

- **Things Oriented Vision:** is the most predominant vision today (Aggarwal et al., 2013). It focuses on linking the real world with the digital world (a kind of cyber-physical system), through traceable and uniquely identifiable objects. Leveraging the objects or 'things' capability to sense, actuate, and share information.
- **Internet Oriented Vision:** focus on the development of the communication protocols that will enable smart objects to be Internet-connected. Since the IP protocol is the main protocol used in the Internet, each device will require its own IP-address. This vision focuses also on developing the Internet infrastructure to accommodate the growing number of connected 'things'.
- **Semantic Oriented Vision:** is focused on the data management that will arise from the information generated by an ever-expanding number of 'things'. For this reason, machine processable semantics are critical for the mechanization of search in the IoT context. Thus providing processable semantics to the IoT objects will enable intelligent search without human intervention, allowing a better machine-to-machine communication and the proposed infrastructure to scale nicely (Toma et al., 2009).

2.2 IoT Architecture

Being a new area, the IoT still does not have a clearly defined and commonly accepted architecture. Nevertheless, several proposals can be found in the literature. For example in (Bandyopadhyay and Sen, 2011), an IoT architecture is represented as a composition of five layers. From a data acquisition layer to an application layer at the top. Where in (Zhao and Ge, 2013; Jing et al., 2014; Yang et al., 2011; Ye et al., 2014), the proposed architecture is divided in three layers: perception layer, network layer, and application layer. Depending on the author, the layer's name might vary, but the meaning remains basically the same. For the purpose of this thesis a three layered architecture will be adopted and the perception layer will be referenced as device layer.

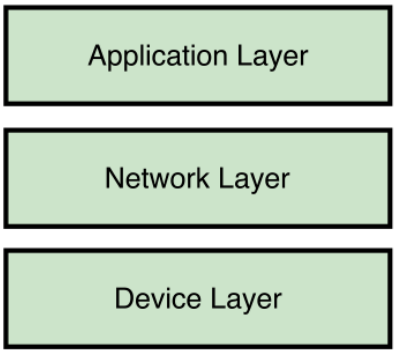


Figure 2.3: IoT layered architecture (Wu et al., 2010)

1. **Device layer:** is where the connected objects are, as the name suggests. All sensing, actuation, and data collection are done at this layer. This layer can be seen as part of the *things-oriented vision*.
2. **Network layer:** receives all data collected from the Device layer and sends them to the application layer. Furthermore, it also supports transmission of commands to the devices, provides basic networking support, and data transfer over wireless or wired network. This layer connects the *smart objects* with the Internet, thus this layer represents the *Internet-oriented vision*. This layer might involve QoS management and control according to the requirements of users/applications.
3. **Application Layer:** is where all the object's data are collected, analysed, and processed so as to support new services and applications. This layer provides the means for a *semantic-oriented vision*.

2.3 What is a Thing?

In IoT the terms 'things', 'objects', 'smart objects', 'devices', and 'nodes' are used interchangeably to give the same meaning as they are frequently used in IoT related documents (Singh et al., 2014). In a recommendation by (ITU-T, 2012), the standardization branch of International Communication Union (ITU), the following definitions are provided:

- **Device:** "With regard to the Internet of Things, this is a piece of equipment with the mandatory capabilities of communication and the optional capabilities of sensing, actuation, data capture, data storage, and data processing".
- **Thing:** "With regard to the Internet of Things, this is an object of the physical world (physical thing) or the information world (virtual thing), which is capable of being identified and integrated into communication networks".

The use of small embedded devices lowers production costs and leverages the wide deployment of these devices in highly competitive markets across different areas (Sehgal et al., 2012). The embedded devices used in IoT are expected to be in the vast majority resource constrained by for example low computational power, memory, storage, radio signal and battery dependency. This raises new challenges to the IoT concept as security measures, operating systems, communication protocols, and network access technologies need to be lightweight in order to be able to work within such constraints.

2.4 Application Examples

The IoT vision is expected to affect different areas of the society. The use of intelligent, small, embedded, connected devices, with sensory and actuation capabilities, will thrive a variety of services and applications that will affect each area of our society. IoT is impacting sectors from Buildings, Energy, Transportation, Environment, to Health etc. The number of possible services that can be derived from the integration of the IoT concept is rather vast. Figure 2.4 illustrates IoT as a tree where at the roots the development in device

technology and its connections will influence the growth of many applications.

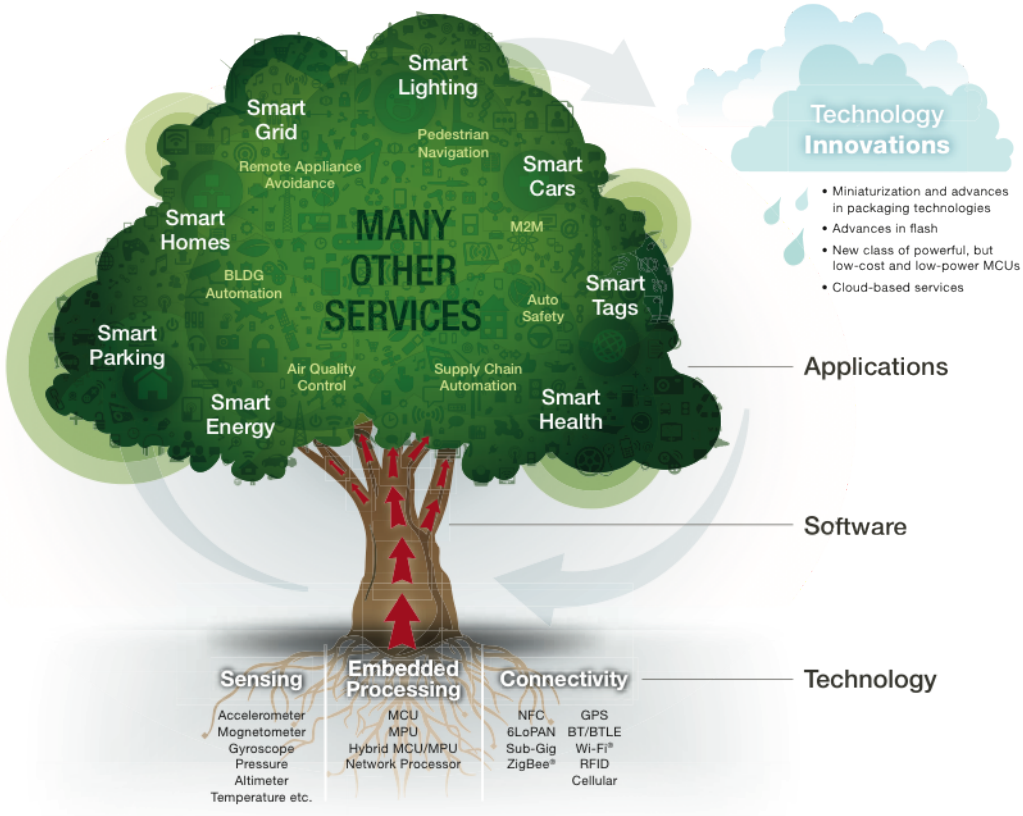


Figure 2.4: The IoT: Different Services, Technologies, Meanings for Everyone (Karimi and Atkinson, 2013)

A brief list of applications and services that can exist within the IoT is described in the following subsections based on (Vermesan and Friess, 2014). The application areas were grouped as follows: Smart Cities, Smart Environment, Security and Emergencies, Logistics and Retail, Smart Industry, Smart Agriculture, Smart Home and Building, and Smart Health. This list is not exhaustive but it gives a reasonable idea of the impact of IoT.

2.4.1 Smart Cities

The cities provide the infrastructure where we travel, work and live. Transportation, energy efficiency and infrastructural health can be improved with the adoption of the Internet of Things. The following bullets identify some possible application scenarios in the city context.

- **Smart Parking:** Real-time monitoring of parking spaces availability in the city

making residents able to identify and reserve the closest available spaces.

- **Structural Health:** Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.
- **Noise Urban Maps:** Sound monitoring in bar areas and centric zones in real time.
- **Traffic Congestion:** Monitoring of vehicles and pedestrian levels to optimize driving and walking routes.
- **Smart Lighting:** Intelligent and weather adaptive lighting in street lights.
- **Waste Management:** Detection of rubbish levels in containers to optimize the trash collection routes.
- **Smart Roads:** Intelligent Highways with warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.
- **Smart Energy Grid**: Real-time energy consumption monitoring and management.

2.4.2 Smart Environment

The Environment can gain a lot from adopting the IoT. As our environment is essential for the well being of life in our planet it could, for instance, benefit from real time monitoring of human caused pollution. Natural disasters such as avalanches, earthquakes, tsunamis and storms could eventually provide more data in real-time that could be used to better prevent and warn future events. Some examples of applications in the environment are as follow:

- **Forest Fire Detection:** Monitoring of combustion gases and preemptive fire conditions to define alert zones.
- **Air Pollution:** Control of CO2 emissions of factories, pollution emitted by cars and toxic gases generated in farms.
- **Landslide and Avalanche Prevention:** Monitoring of soil moisture, vibrations and earth density to detect dangerous patterns in land conditions.
- **Water Pollution Monitoring:** Real-time control of rivers and oceans water for waste and chemical leakages.
- **Wild Life Protection:** Detection and warning of near by sea animals to ships and boats so as to avoid collisions.

2.4.3 Security and Emergencies

Security is important as a preemptive measure to avoid damage, harm and causalities. As recently seen in Tianjin, China an explosion of containers with hazardous-chemicals occurred (Bloomberg, 2015). In an IoT context, the real time monitoring of the state, composition and condition of the content stored in the containers could avoid the damage it caused. Another case was the explosion in a nuclear power plant in Fukushima in 2011, Japan that led to the evacuation of a large populated areas due to radiation threats (McCurry, 2011). Since then, efforts to build networks of mobile sensors that monitor and map the levels of radiation have been pursued (George, 2013), namely with an approach in which the citizens help measuring the radiation levels and share them in a common Internet platform. The following bullets describe possible IoT applications in this context.

- **Perimeter Access Control:** Access control to restricted areas and detection of intruders in non-authorized areas.
- **Liquid Presence:** Liquid detection in data centers, warehouses and sensitive building grounds to prevent breakdown and corrosion.
- **Radiation Levels:** Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.
- **Explosive and Hazardous Gases:** Detection of gas levels and leakages in industrial environments, surroundings of chemical factories and inside mines.

2.4.4 Logistics and Retail

The retail sector could benefit from the consented collection of data about their consumer habits levering a better marketing of their products. While logistics could benefit from a real-time monitoring and management of their assets. The following bullets express the IoT possible contribution to these areas.

- **Supply Chain Control:** Monitoring of storage conditions along the supply chain and product tracking for traceability purposes.
- **Intelligent Shopping Applications:** Getting advices in the point of sale according to customer habits, preferences, presence of allergenic components for them or expiring dates.

- **Smart Product Management:** Control of rotation of products in shelves and warehouses to automate restocking processes.
- **Quality of Shipment Conditions:** Monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes.
- **Item Location:** Search of individual items in big surfaces like warehouses or harbours.
- **Storage Incompatibility Detection:** Warnings on containers storing inflammable goods close to others containing explosive material.
- **Fleet Tracking:** Control of routes followed for delicate goods like medical drugs, jewels or dangerous merchandises.

2.4.5 Smart Industry

The IoT presence in the Industry could improve the operational efficiency and productivity gains. The IoT could also help Industry connect with other services such as smart grid, logistics, maintenance or sharing the production facility as a service enabling a smarter and more efficient production line (Vermesan and Friess, 2014). The following text gives some examples of applications in the Industry context.

- **M2M Applications:** Machine auto-diagnosis and assets control.
- **Indoor Air Quality:** Monitoring of toxic gas and oxygen levels inside chemical plants to ensure workers and goods safety.
- **Temperature Monitoring:** Control of temperature inside industrial and medical fridges with sensitive merchandise.
- **Ozone Presence:** Monitoring of ozone levels during the drying meat process in food factories.
- **Indoor Location:** Asset indoor location by using active (ZigBee) and passive tags (RFID/NFC).
- **Maintenance and Repair:** Early predictions on equipment malfunctions where service maintenance can be automatically scheduled ahead of an actual part failure by installing sensors inside equipment to monitor and send reports.

2.4.6 Smart Agriculture

The agriculture sector faces great challenges to feed a growing population. The smart agriculture could improve the production and quality of our food supply. Contributing for intelligent fields that allow a precise deployment of pesticides or watering only when needed, and a better monitoring of the animals health are some examples. The following bullets give an example of possible applications in the agriculture context.

- **Green Houses:** Control micro-climate conditions to maximize the production of fruits and vegetables and its quality.
- **Intelligent Irrigation:** Selective irrigation in dry zones to reduce the required water resources.
- **Smart Pest Control:** Early detection, through sound and movement sensors, of harmful organisms and selective deployment of pesticides or insecticides reducing waste of the latter and pollution of the environment.
- **Wine Quality Enhancing:** Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.
- **Animal Tracking:** Location and identification of animals grazing in open pastures or location in big stables.
- **Monitoring Gas Levels:** Study of ventilation and air quality in farms and detection of harmful gases from excrements.

2.4.7 Smart Home and Building

Smart Home and Building is another sector where IoT is expected to have a larger impact in helping improve our daily lives and change how we view our homes and buildings. Rather than a passive home, the IoT could leverage a home that provides services such as informing you what is missing in your refrigerator. The following bullets express possible applications in a smart home or intelligent building context.

- **Energy and Water Use:** Energy and water supply consumption monitoring to obtain advice on how to save cost and resources.
- **Remote Control Appliances:** Monitoring and switching on and off remotely-controlled appliances to avoid accidents and save energy.

- **Intrusion Detection Systems:** Detection of windows and doors openings and violations to prevent intruders.
- **Intelligent Thermostat:** Thermostat that learns the user's programming schedule after a few days, and from that programs itself.
- **Intelligent Fire Alarm:** System with sensors measuring smoke and carbon monoxide, notifying the user where the threat is coming from and if needed the fire-fighters.

Regarding this sector, the IoT is already making its presence felt on the consumer market with numerous Internet enabled sensors and appliances.

2.4.8 Smart Health

The IoT can provide the means to help connect patients with their doctors, leveraging the use of devices that can remotely monitor and report about patients' conditions. Possible application examples are described as follows.

- **Fall Detection:** Assistance for elderly or disabled people living independent.
- **Medical Refrigeration:** Control of conditions inside freezers storing vaccines, medicines and organic elements.
- **Athlete Care:** Monitoring of vital signs in high performance centers and fields.
- **Patient Surveillance:** Monitoring the conditions of patients inside hospitals and in nursing homes.
- **Ultraviolet Radiation:** Measurement of UV sun rays to warn people not to be exposed in certain hours.
- **Remote Diagnosis:** Ingestible Small devices could provide accurate diagnostics about a sick individual without leaving home.
- **Dental Health:** smart toothbrush analyses brushing, monitors possible dental disease and shares collected information with the user's smartphone and/or dentist.

2.5 Current Market Trends

The Internet of Things is currently one of the most hyped emerging technologies (see Figure 2.5). Gartner(Gartner, 2014b) estimated that by 2020 the IoT will include 26 billion of devices and generate incremental revenue exceeding $300 billion from products

and services.

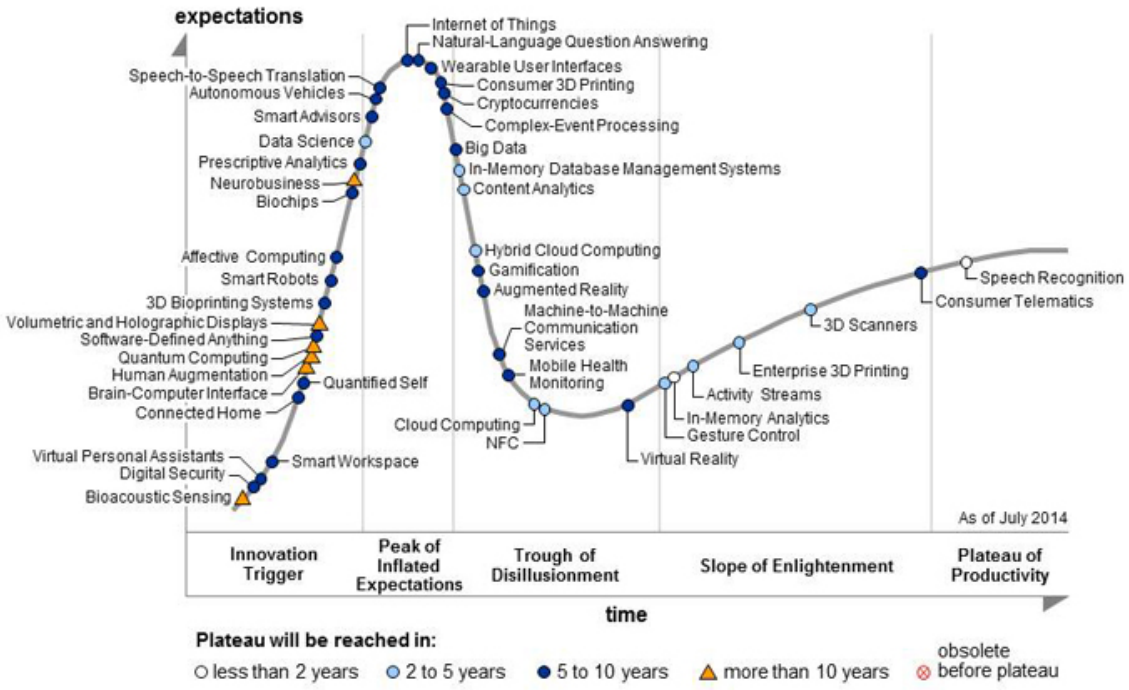


Figure 2.5: Hype Cycle for Emerging Technologies, 2014(Gartner, 2014c)

By 2015 it is expected to have almost 3 billion consumer IoT products installed and 13 billion by 2020 (see table 2.1).

Table 2.1: Internet of Things units installed in millions base by category(Gartner, 2014a)

Category	**2013**	**2014**	**2015**	**2020**
Automotive	96.0	189.6	372.3	3,511.1
Consumer	1,842.1	2,244.5	2,874.9	13,172.5
Generic Business	395.2	479.4	623.9	5,158.6
Vertical Business	698.7	836.5	1,009.4	3,164.4
Grand Total	3,032.0	3,750.0	4880.6	25,006.6

It is expected that consumer applications will drive the major number of connected things, as smart appliances, wearables and other consumer IoT products become more prevalent in the market.

2.6 Current State of Security in IoT

In this section a brief insight is presented on the current state of security for IoT products. The ways in which organizations in the IoT field are preparing themselves to tackle the security and privacy issues are also discussed. This section is based on research done by Capgemini Consulting and Sogeti High Tech (Capgemini, 2014) on more than 100 large enterprises and startups.

The survey focused in the following two aspects:

- A main survey concerning security in IoT
- And a benchmarking assessment of organizations' privacy policies.

2.6.1 Consequences of a Vulnerable IoT

The IoT brings with it the potential to improve the everyday-life of people and business. However it depends on a critical factor: security. In fact, numerous attacks on the IoT market have already taken place.

- **"ThingsBot"**: Proofpoint[1] a security provider uncovered a cyberattack on conventional smart home appliances. More than 100,000 gadgets (e.g. televisions, refrigerators, connected multimedia centers, etc) had been compromised and used as a platform to launch malicious emails (ProofPoint, 2014).
- **"Target's Data Breach"**: Security vulnerabilities in the Target's[2] internet-enabled heating, ventilation and air-conditioning systems (Vijayan, 2014), lead to the theft of 40 million credit card numbers (Townsend et al., 2014).
- **Vulnerable home appliances**: IOActive[3] uncovered multiple vulnerabilities in the Belkin[4] WeMo Home Automation devices (i.e. Internet-enabled plugs, lights, cameras). These vulnerabilities exposed the WeMo products to unauthorized firmware updates, remote monitoring, remote control and access to the internal home networks (IOActive, 2014).

[1]Proofpoint is a security-as-a-service vendor that delivers data protection solutions, `https://www.proofpoint.com`

[2]Target is a US-based retailer, `http://www.target.com/`

[3]IOActive is a provider of specialist information security services,`http://www.ioactive.com`

[4]`http://www.belkin.com/us/`

- **Wireless Carjackers**: Recently two hackers showed how they managed to remotely control a jeep. Without tampering the car, they managed to gain access to the dashboard 'media center' - which provides access to services over the Internet to the driver - of the car which in turn allow them to issue commands to the car's internal computer network. They managed to control all the dashboard functions (i.e. ventilation, radio, etc.) as well as turning off the engine, disabling the brakes and controlling the steering wheel (Greenberg, 2015).

As illustrated by the above examples, the security vulnerabilities of the IoT products can compromise the growth of the IoT, becoming a potential business stopper. Cyber attacks can cause significant damage to the organizations, as proved from Target's data leakage.

"Losses arising from cyber attacks on IoT systems can hit organizations hard. Target faced plummeting sales and saw a 46% drop in profitability as a result of the November 2013 attack. In addition, the company could potentially face a fine of $400 million to $1.1 billion if a government probe finds it guilty of not following industry-specific security standards"(Capgemini, 2014).

More than half of the surveyed companies agreed that security concerns will influence customers purchasing decision for IoT products.

Figure 2.6 shows that industial manufacturing firms have greater concern towards the security threats than Automotive and Home Automation firms, accordingly to the survey.

The number of IoT devices is expected to surge in the near future. As a consequence the Internet-enabled systems will become increasingly attractive targets for cyber attacks.

2.6.2 Security Vulnerabilities in IoT Products

The research of capgemini shows that despite the impact on the trust and growth as a result of cyberattacks, most of the surveyed organizations do not provide adequate security and privacy in their products. Accordingly to an HP study (HP, 2014a), done on ten of the most popular IoT devices, 25 security vulnerabilities on average, were found per device. The devices ranged from connected TVs, door locks, remote power outlets to

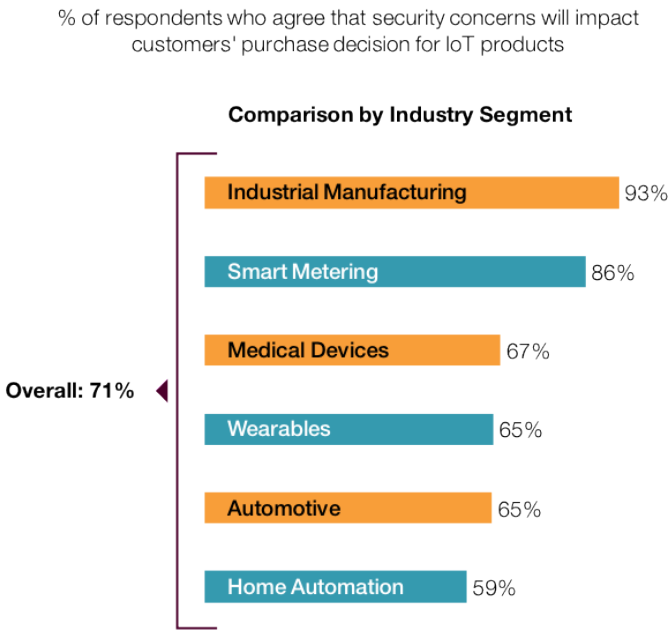


Figure 2.6: Impact of security concerns on customers' purchase decision for IoT products (Capgemini, 2014)

home alarms. The most common security issues found in this study where:

- **Privacy Concerns**
- **Insufficient authorization**
- **Lack of encrypted communications**
- **Insecure web interfaces**
- **Inadequate software protection**

The collection of personal information while using unsecured communication channels, weakly secured websites and devices poses a significant risk to the adoption of IoT products in the future that should be taken seriously by the IoT enablers.

2.6.3 Lack Maturity of Privacy Policies

The majority of the IoT devices with remote accessibility use a third party to bridge the communications over the Internet with a remote user providing any type of services from the data collected. How the gathered data is saved, processed and shared is still not clear to the user. Thus, this opens the door to serious personal privacy threats. Data ownership and transparent privacy policies need to be addressed.

"In addition to securing data to make sure that it does not fall into the wrong hands, issues of data ownership need to be addressed in order to ensure that users feel comfortable participating in the IoT"(Whitmore et al., 2014):

Despite the privacy concerns, almost half of the 100 surveyed companies on the privacy topic, did not provide any privacy related information regarding their products.

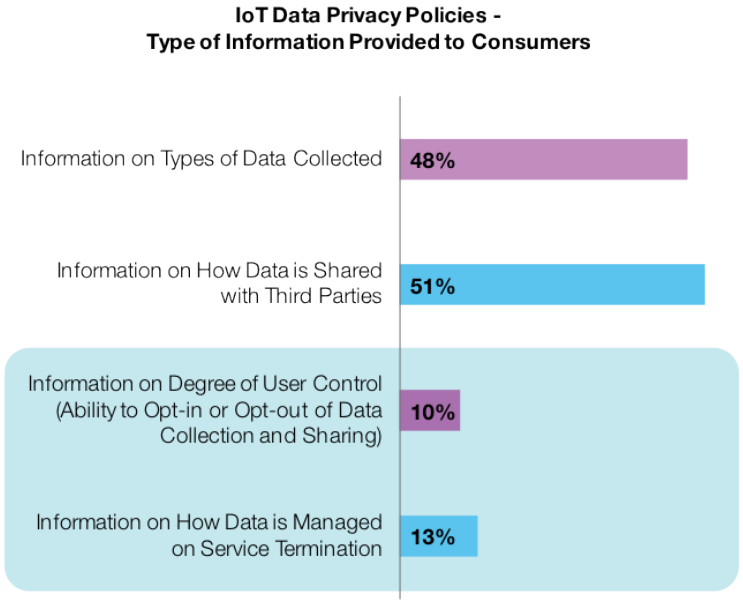


Figure 2.7: Data privacy information provided by organizations, N=100,(Capgemini, 2014)

The ability of the consumer to opt-in or opt-out of data collection and sharing is practiced by just 10% of the companies (Figure 2.7), and only 13% provided any information how consumer data is handled after the service is terminated.

It is important to stress that data privacy policies go beyond the service provider and consumer. IoT is expected in all areas and contexts (e.g. Home Automation, Smart City, Energy, Health, etc), with ubiquitous collection of data.

A clear policy on the ownership of data is important so as to ensure that the users data is used with their full consent. Privacy policies can ensure that the user data is employed in a transparent way and with the full consent of the user.

2.6.4 Securing IoT Products and Systems

The survey on the state of security by Capgemini reveals that the majority of the companies in the IoT sector are lagging behind in securing their IoT products and systems. A number of factors that influence the ability of the companies to secure their products, regarding the surveyed companies include (Capgemini, 2014):

- **Expanded attack surface:** multiple points of vulnerability stem from the embedded software and data residing in the device as well as in the data aggregation platforms, data centers used for analysis of 'sensed' data, apps and communication channels.
- **Inefficiencies in product development:** most of the surveyed companies do not focus on securing their IoT products from the beginning of the product development phase.
- **Weak security architecture:** only half of the companies provided remote updates for their IoT products, relying on consumers to download and install the updates.
- **Lack of specialized security skill-sets:** securing the IoT product requires securing the app, device, infrastructure and the communication channel: despite this the majority of the companies did not acknowledge a shortage of security experts in their organizations as a key challenge in securing their products.
- **Insufficient use of third-party support:** the study also revealed that the vast majority of the companies are not partnering or acquiring specialized security firms as part of their security strategy.

2.7 Research on IoT Security

With concerns for the security and privacy in the Internet of Things, the European Commission has been funding many projects. As an illustration, Table 2.2 presents a list of several projects on security for the Internet of Things.

Table 2.2: Some European funded projects on IoT security

Project	**Funds**	**Description**
Butler	EU FP7	Aims at developing secure and smart life applications for the IoT, emphasizing pervasiveness, context-awareness and security for IoT in a opened architecture
EBBITS	EU FP7	Proposes a framework for business to semantically integrate with the Internet of Things. It supports end-to-end interoperable business applications and an Intrusion Detection System framework for the IoT.
Hydra	EU FP7	Hydra develops middleware for networked embedded systems that can work with any network technology, providing distributed security and social trust components into devices and services.
uTRUSTit	EU FP7	Aims at enhancing the trust of user to the IoT. Working in providing the tools for IoT manufacturers and integrators to present a transparent view to the user of the information being collected and shared between devices and services.
FIRE	EU FP7	Provides an infrastructure to enable research and large-scale experimentation for the Future Internet.
iCore	EU FP7	Proposes a framework with three levels of functionality: virtual objects, composite virtual objects and functional block, for representing the user/stakeholder perspectives equipped with essential security functionalities.

Adapted from (Sicari et al., 2015).

Chapter 3

Enabling Technologies

This chapter provides an overview of the technologies that will help the growth of the IoT concept. Taking into consideration the generic architecture of the Internet of Things cited in the previous section, this chapter can be divided in three parts accordingly. The first part focuses on device technologies such as hardware platforms and lightweight operating systems for small embedded devices. The second part focuses on the networking and communication technologies while the final part makes a reference to technologies that help the integration of the device and communication technologies. A major reference to the openHAB framework is done due to the fact that it will be used in the case study.

3.1 Device Technologies

3.1.1 Hardware Platforms

One of the biggest drivers of the Internet of Things is the increasing number of low-cost sensors arriving on the market (Swan, 2012). This allows for the future *'things'* to sense movement, light, temperature, electrical potential, etc, thereby enhancing their awareness of the environment surrounding them. With the development of technology, the device computational and storage capabilities keep increasing while their size decreases. While sensor and actuator technology provides the ability to feel and act, processing units (e.g. microcontrollers and microprocessors) and software provide the brain and the computational capability for the IoT devices. Hardware platforms provide the means to integrate

the sensing, actuating, processing and connectivity into IoT devices. A variety of hardware development platforms already exist in the current market such as Arduino, Rasperry PI, Intel Galileo, Beaglebone, NodeMCU, CubieBoard, Mulle, WiSense, Z1, FriendlyARM, T-Mote Sky, etc (Al-Fuqaha et al., 2015). These development platforms have different characteristics differing in computational power, memory, connection type, size, etc allowing the development of a vast and diverse variety of IoT devices for numerous application contexts. The number of hardware solutions to develop IoT is continually increasing; a more exhaustive list of hardware solutions for IoT can be found in (Postscapes).

3.1.2 Operating Systems for the Internet of Things

Ubiquitous embedded smart devices that sense and actuate are expected to be deployed in a myriad of areas. The miniaturization of such devices will boost their ubiquitous deployment as devices become cheaper.

Nevertheless, most of the IoT devices are expected to be resource constrained, equipped with the minimum hardware to perform the task that they were conceived for. These features require a tailored operating system for the IoT devices that is easily configurable, programmable and lightweight.

Accordingly to (Dong et al., 2010), the IoT operating systems (OS) should have the following features:

- **Scalable:** to accommodate a wide range of different hardware platforms.
- **Modular:** OS equipped with only the minimum components necessary, allowing the addition of other components if necessary.
- **Connected:** supporting TCP/IP stack protocols and other communication protocols that allow network connectivity.
- **Reliable:** most devices are expected to be deployed once for a long period of time. OS reliability to bugs and easily patchable is of great importance for the correct functioning of the devices.
- **Real-Time Guarantee:** applications such as surveillance and actuation on critical environments require a real-time communication of information, thus timely reposes

should be guaranteed.

- **Lightweight:** capable of running with low memory, limited battery, and small processor as well as being energy efficient.

There are four major operating systems that are discussed by the academia and these are TinyOS, Contiki, RIOT and Linux (Baccelli et al., 2013). A more extensive description of other OSs suitable for IoT can be found in Gaur and Tahiliani (2015); Dong et al. (2010).

Table 3.1: Comparison of different operating systems for IoT (Minerva et al., 2015)

OS	Min RAM	Min ROM	C Support	C++ Support	Multi-Threading	Modularity	Real-Time
Contiki	<2KB	<30KB	P	N	P	P	P
Tiny OS	<1kB	<4kB	N	N	P	N	N
Linux	~1MB	~1MB	Y	Y	Y	Y	P
RIOT	~1.5kB	~5kB	Y	Y	Y	Y	Y

Table 3.1 depicts the major differences between those cited operating systems, where, P means: *supports partially*, N means: *does not support*, and Y means: *support fully.*

3.2 Communication Reference Model

There are two main communication reference models: the Open Systems Interconnection (OSI) and the Transport Communication Protocol/Internet Protocol (TCP/IP) model (Alhamedi et al., 2014; Simoneau, 2011). These are defined by a number of layers, each of them with a particular function in the communication process. The TCP/IP model is currently used to define the protocol stack that rules the Internet. Figure 3.1 shows the relation between these models and gives an example of the protocols commonly used in each layer of the TCP/IP model.

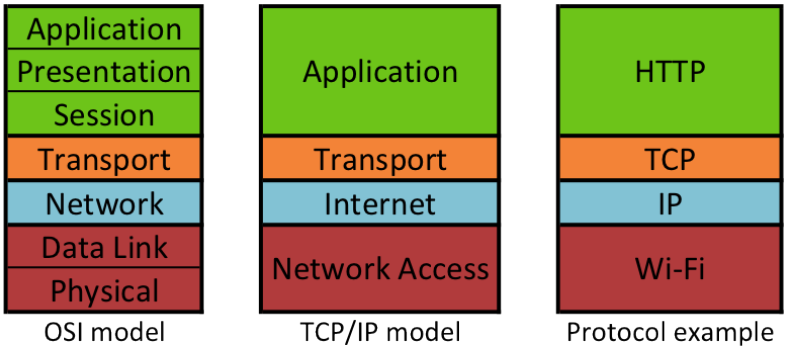


Figure 3.1: TCP/IP model (center) in relation to the OSI model (right) and protocol examples (left) (Alhamedi et al., 2014)

The 4 layers that compose the TCP/IP model be can described as follows:

- **Network Access Layer:** provides the interface with the network medium, it is responsible to convert data streams to signals to be sent through the network medium and vice versa. It manages the access to the network medium and reliability of the communication. In figure 3.1 Wi-Fi can be seen as a network access layer protocol.
- **Internet layer:** is responsible for the delivery of packets end to end and implements a logical addressing scheme to help accomplish this. The Internet Protocol (IP) is the core network layer protocol of the Internet.
- **Transport Layer:** generates communication sessions between applications running on different hosts. It is also responsible for error correcting and reliability of the data packets over the network. The TCP protocol can be seen in figure 3.1 as an example of a transport protocol.
- **Application Layer:** has the responsibility for authentication, data formatting, and it governs the data flow in an optimal scheme for specific applications. A popular application layer protocol is the HTTP (Hyper-text Transfer Protocol) which was created to transfer web content over the Internet.

Although the OSI and TCP/IP models can be seen as a reference for the IoT architecture, the latter represents a wider spectrum of concepts beyond the communication technologies. The reference models help in understanding the specific IoT communication technology roles and position in a communication layered model.

3.3 Network Access Technologies

IoT connectivity can be achieved through cellular technologies (e.g 2G, 3G, LTE) or using radio technologies such as Wi-Fi, Zigbee, 6LowPAN (IPv6 over Low Power Personal Area Networks) or Bluetooth, so as to share data with the Internet (Karagiannis et al., 2015). However, due to the data transmission cost of the cellular technologies, a widespread use of this technology might be difficult to scale within the IoT context. The cited radio technologies provide means to create wireless sensor and actuator networks at a low cost and with a low power demand, thus they are expected to be used in the majority of the IoT application scenarios. The vast number of expected IoT connections will make unfeasible and/or unpractical the use of wired solutions (Mahmood et al., 2015), thus this section will focus on wireless network technologies more specifically the radio technologies.

3.3.1 Network Topologies

To better understand the wireless networks it is important to understand the network topology that they adopt. Routing, security, scalability and other factors vary depending on the network topology adapted.

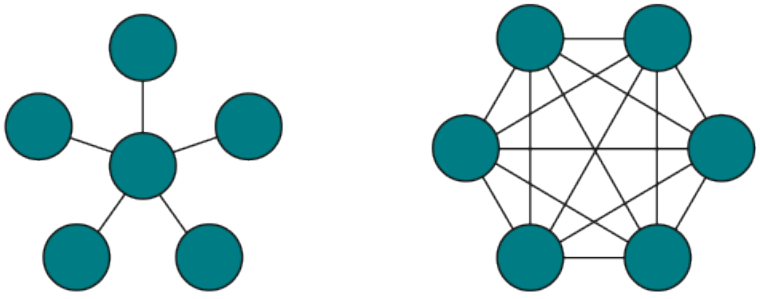


Figure 3.2: Some network topologies, respectively: Star and Mesh (Reiter, 2014)

Figure 3.2 depicts the two commonly used network topologies: Star and Mesh. The following attributes are used to characterise the network topology:

- *Latency:* defines the speed of the network, i.e. the time that takes for a packet of data to travel between the sensor or actuator node and the gateway. The lower the latency the faster the network.
- *Fault resilience:* if a fault occurs in the network to which degree it can recover.

- *Scalability:* defines the number of nodes that can exist in a single network.
- *Number of hops:* defines how many nodes the packet has to pass through to reach its destination.
- *Node range:* The range of a node is the maximum distance of one hop, i.e. from one node to the next node.
- *Network range:* The range of the network is the overall distance a complete network can span.

Star Topology

In a star topology all the nodes are connected directly to a central gateway node. This topology benefits of a low latency due to the fact that the packets have to travel one hop to reach the gateway node. However, if the gateway node stops working, the connectivity between nodes is interrupted, becoming a single point of failure. Furthermore, it has a low scalability compared with a mesh topology, as it depends on the number of nodes that the central can support. This can be a problem as the IoT is expected to deploy a substantial number of nodes. Deployments of nodes with a large distance between them and the central node and the presence of physical obstacles (e.g. walls) are limited by the strength of the signal. The use of a stronger signal may compromise the lifespan of battery powered nodes.

Mesh Topology

In a mesh topology every node can connect to multiple nodes. An advantage of this topology is a high fault resilience as each node has multiple routing routes it can choose from. The mesh topology scales well and it can extend the range of the network through multiple hops, while maintaining a low radio transmission power. However, the mesh topologies are more complex and have a higher latency due to the number of hops a packet needs to get to the destination.

3.3.2 Zigbee

ZigBee is a wireless mesh network built on the IEEE standard 802.15.4 with a low data rate, low-power and low-cost technology ideal for IoT applications. ZigBee benefits from all the advantages of a mesh topology namely high scalability, fault resilience, and long range. This technology has become very popular among home automation, smart energy, and lighting control applications[1], but it can be found in other areas such as health care and retail services. New ZigBee specifications[2] enable battery-free devices to participate in Zigbee networks. In order to enable an Internet connection, a gateway is required to translate the packets from the Zigbee network to the Internet over Ethernet or Wi-Fi. In order to ensure communication security ZigBee uses access control lists or Advanced Encryption Standard (AES-128) to guarantee a high-level security.

3.3.3 Wi-Fi

Wi-Fi is well established technology based on the IEEE 802.11 standard. Wi-Fi access points can be found in every home, office, school, etc, and it is widely supported by the majority of devices in the market from smartphones to game consoles, computers and TVs, which makes it easy to be deployed everywhere. Wi-Fi was not built with constrained devices in mind. It supports high data rates and uses a high radio transmission power that is not ideal for battery powered devices. Wi-Fi networks have a star topology, with the access point being the Internet gateway. Wi-Fi networks normally provide good coverage in most cases; however, the access point does not support a high number of nodes compared with Zigbee. Nevertheless, new low power Wi-Fi chips with better sleep and wake-up management promise longer battery lifetimes. In Tozlu et al. (2012) the authors demonstrate the feasibility of the Wi-Fi technology to enable IP connectivity of battery powered devices.

[1] A list of Zigbee applications can be found in `http://old.zigbee.org/Products/ByStandard/ZigBeeLightLink.aspx`

[2] `http://www.zigbee.org/zigbee-for-developers/network-specifications/zigbeepro/`

3.3.4 Bluetooth Low Energy

Bluetooth is a well known wireless communication system, based on the IEEE 802.15.1 standard. It is widely adopted in smartphones, tablets, computer mouses, headphones etc. It is a proven low power technology that enables battery powered devices to share data in a short range. Recently, the Bluetooth Low-Energy (BLE) short range radio was developed having lower power consumption than the previous version, thereby enabling battery lifetimes up to years in devices powered by coin cell batteries. This makes it an ideal solution for IoT constrained devices. The older version of Bluetooth could connect in a star topology with up to eight devices having a maximum range of 10 meters. The new version can support at least 20 devices with a maximum range of 100 meters. However, Bluetooth does not have the advantage of a native IP-network compatibility such as Wi-Fi, which implies the use of a gateway for protocol translation. An overview of the Bluetooth Low energy in the IoT context was made by Gomez et al. (2012).

3.3.5 6LowPAN

The 6LowPAN (IPv6 low power wireless personal area) protocol was standardized by the Internet Engineering Task Force (IETF) (Hui and Thubert, 2011), in order to provide the transmission of Internet protocol version 6 (IPv6) packets over the IEEE 802.15.4 standard for LowPAN. As the expected number of connected IoT devices is expected to overwhelm the IPv4 address space, the IPv6 comes in so as to provide enough address space for the IoT. The 6LoWPAN reduces the IPv6 overheads making it lighter along with benefits of a mesh topology (Al-Fuqaha et al., 2015). Zigbee is adopting the 6LowPAN in order to achieve IP-functionalities, which facilitates the connectivity with other devices and services over the Internet (Mahmood et al., 2015).

3.3.6 Z-Wave

Z-Wave is a wireless protocol architecture developed by ZenSys and promoted by the Z-Wave Alliance (Gomez and Paradells, 2010). It has a good presence in the IoT market with around 35 million devices deployed (Sigma et al., 2014). Like Zigbee, Z-Wave uses the mesh network topology giving it a good range and scalability compared with other

star networks (e.g. Wi-Fi and BlE). The main advantages of Z-Wave is that it uses IP protocol and operates on the 900 MHz, band avoiding interference of the crowded 2,4 GHz band used by Wi-Fi, Zigbee and Bluetooth.

3.3.7 Remarks

Each of the wireless technologies previously presented provides different advantages that can better fit different deployment contexts. Wi-Fi has the advantage of being widely available but being the less constrained-device friendly compared with the other technologies. The Bluetooth Low Energy (BLE) provides a good range compared with the 'classic' Bluetooth and a good communication protocol for low powered devices. It also has a wide availability as the Wi-Fi. Zigbee, Z-Wave and 6LowPAN adopt a mesh network topology that can accommodate a larger number of nodes and have a greater range and reliability. The Zigbee and Z-Wave operate at low power and have a low cost. A main advantage of Z-Wave compared with the Zigbee is that it natively supports the IP protocol, making it easier to communicate over the Internet.

Table 3.2: Network Access Technologies for the Internet of Things (Mahmood et al., 2015).

	Zigbee	**Wi-Fi**	**BLE**	**6LowPAN**	**Z-Wave**
Network topology	Mesh	Star	Star	Mesh	Mesh
Data rate	40/250 kbps	11-300 Mbps	1 Mbps	20/40/250 kbps	40 kbps
Node range(indoor)	75 m	100 m	100 m	Up to 200 m	30 m
Standard	IEEE 802.15.4	IEEE 802.11	IEEE 802.15.1	IETF RFC 4944	Proprietary
IP support	IPv6 only in SEP2	Yes	No	Yes, IPv6	Yes
Adoption rate	Widely adopted	Extremely high	Extremely high	Medium	Medium
Unique value	Low cost, low power usage, high number of nodes	High speed mature standards	Ease of access, no configuration requirement, secure connection	Benefits of both IP and Bluetooth, low power consumption	No interference from household devices

Table 3.2 summarizes the main characteristics of the network technologies previously referenced.

Adapted from (Mahmood et al., 2015).

3.4 IoT Message Protocols

3.4.1 Introduction

The IoT devices are expected to be, in the vast majority, resource constrained. Thus, lightweight protocols that do not require extensive use of CPU resources are needed. This section provides an overview to the existing data exchange protocols that better suite the IoT, according to (Al-Fuqaha et al., 2015) these are:

- **CoAP:** Constrained Application Protocol
- **MQTT:** Message Queue Telemetry Transport
- **XMPP:** Extensible Messaging and Presence Protocol
- **RESTful:** Representational State Transfer
- **AMQP:** Advanced Message Queuing Protocol
- **DDS:** Data Distribution Service.

In further subsections there will be a brief description of the cited protocols based on (Al-Fuqaha et al., 2015; Karagiannis et al., 2015; Moessner et al., 2013; PrismTech, 2013).

There are two main messaging patterns for data exchange protocols: publish/subscriber model that includes broker-based and bus-based, and request/response model, explained in the following subsections.

3.4.2 Publish/Subscribe Model

Broker-based Architecture

In the broker-based architecture (see Figure 3.3), the broker (i.e. server) controls the distribution of the information to its clients. Each client can switch between publisher and subscriber roles. The publisher sends information to the broker to a specific topic and the subscriber receives automatic messages every time there is a new update in a topic he has subscribed.

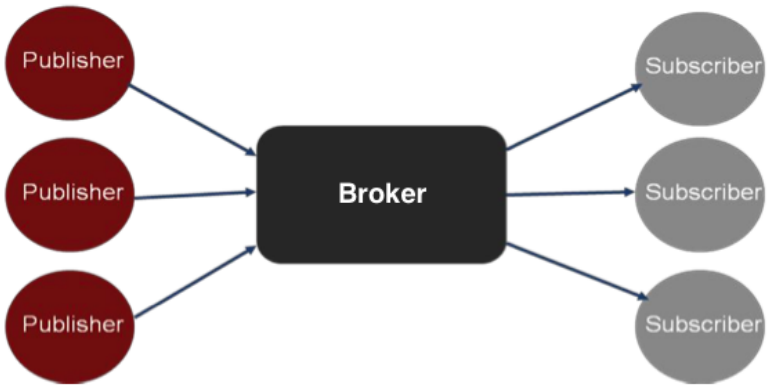


Figure 3.3: Broker based architecture for exchange message protocols(PrismTech, 2013)

Examples of broker–based protocols include Advanced Message Queuing Protocol (AMQP) and Message Queue Telemetry Transport (MQTT).

Bus-based Architecture

The bus-based architecture (see Figure 3.4) is a decentralized broker-less architecture.

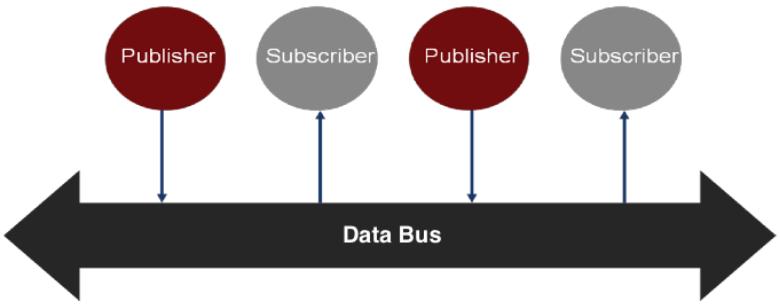


Figure 3.4: Bus based architecture for exchange message protocols(PrismTech, 2013)

Clients publish messages for a specific topic which are directly delivered to the subscribers of that topic.

3.4.3 Request/Response Model

The request/response model is widely use in the Internet. HTTP, one of the most used protocols in the Internet, uses the request/response approach.

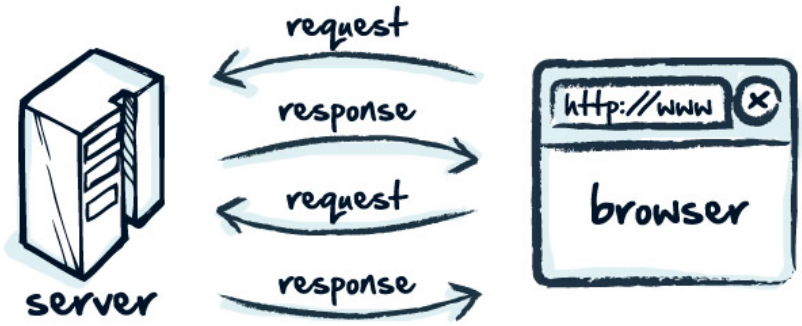


Figure 3.5: Request/response message exchange pattern (Costa, 2011)

A request/response messaging requires the messages to be responded; as a result it consumes more network bandwidth and message processing compared with the publish/-subscribe messaging. As a majority of the IoT devices are expected to be battery powered, a less resource intensive message exchange protocol is preferred.

3.4.4 Data-centric and Message-centric

The data exchange protocols can be also classified as data-centric or message-centric.

- **Message-centric:** Focus on delivering a message regardless of the content. The infrastructure's objective is to ensure that messages get to their intended recipient (e.g. MQTT, CoAP, RESTful, XMPP and AMQP).
- **Data-centric:** Focus on the *"data value"* of the message. The infrastructure's objective is to guarantee that all nodes have the correct understanding of that value (e.g. DDS).

The advantages and disadvantages of the message-centric and data-centric messaging depends on the application context. Beyond the simple delivery of messages, the data-centric approach is better suited for applications that need a data-base model (Schneider, 2012).

3.4.5 CoAP

The Constrained Application Protocol (CoAP) is a synchronous request/response designed by the Internet Engineering Task Force (IETF) to target constrained-resource devices (Shelby et al., 2014). It uses the HTTP commands to provide resource-oriented

interaction in a client-server architecture, where typically the device is a server of information. CoAP uses UDP (User Datagram Protocol) a connectionless protocol and a lighter alternative to the TCP protocol. CoAP utilizes synchronous and asynchronous responses, supporting unicast and multicast, opposed to TCP. As UDP protocol does not ensure the delivery of message, CoAP as the following Quality of Service (QoS) messages:

- *Confirmable:* A request message that requires an acknowledgement (ACK). The response can be sent either synchronously (within the ACK) or if it needs more computational time, it can be sent asynchronously with a separate message.
- *Non-Confirmable*: A message that does not need to be acknowledged.
- *Acknowledgment:* It confirms the reception of a confirmable message.
- *Reset:* It confirms the reception of a message that could not be processed.

In order to ensure secure communications DTLS (Datagram Trasport Layer Security) can be used. DTLS provides authentication, data-integrity, confidentiality, automatic key management and cryptoraphic algorithms. However DTLS adds additional network traffic that can shorten the lifespan of battery powered devices.

3.4.6 MQTT

Message Queue Telemetry Transport (MQTT) is an open-source protocol, standardized by OASIS (OASIS, 2014), and targets lightweight M2M communications. It implements an asynchronous broker-based architecture on top of the TCP stack. The MQTT protocol is optimized to connect constrained devices with enterprise servers and other consumers. It was designed to use bandwidth and battery usage economically, and it is being used by Facebook Messenger[3] for that reason. MQTT provides three Quality of Service (QoS) levels as follows:

1. *Fire and forget:* A message is sent once and no acknowledgement is required. Message loss can occur.
2. *Deliver at least once:* A message is sent at least once and an acknowledgement is required. Duplicates can occur.
3. *Deliver exactly once:* A four-way handshake mechanism is used to ensure the message is delivered exactly one time.

[3]http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger

Although it is uses TCP, it is designed to have a low overhead compared with other TCP-based application layer protocols. In terms of security, the MQTT protocol uses TLS/SSL[4] (i.e. one of the most used security protocols in the Internet) to ensure privacy in its communications. In addition, the broker might use username and password credentials for authentication.

3.4.7 XMPP

The Extensible Messaging and Presence Protocol (XMPP) is a communications protocol that provides an asynchronous publish/subscribe and a synchronous request/response methods. XMPP runs over TCP, implementing small footprint and low latency messages designed for near real-time communications. XMPP is a well established protocol supported all over the Internet and it can implement the publish/subscribe method that is more suitable to IoT applications. However, it uses Xtensible Markup Language (XML) that create additional computational overhead due to unnecessary tags and XML parsing (Karagiannis et al., 2015).

3.4.8 RESTful Services

RESTful is a language and operating system independent architecture style for designing network applications using simple HTTP to connect between machines. It uses the resquest/response method and it only requires HTTP libraries. RESTful is already an important part of the IoT because it is supported by all the commercial M2M cloud platforms and mobile platforms. RESTful services use the secure and reliable HTTP making use of TLS/SSL for security. However, HTTP is not well suited for constrained-communication devices due to its request/response nature. CoAP is a lightweight version of REST as it uses UDP in opposite to TCP, nevertheless it shares the same request/response architecture.

3.4.9 AMQP

The Advanced Message Queuing Protocol (AMQP) is a protocol that arose from the financial industry. It provides asynchronous publish/subscribe communication over TCP

[4]TLS/SSL stands for Transport Layer Security and Secure Socket Layer

stack. Besides TCP, it can support different transport protocols. A main difference from MQTT and AMQP is that the latter was designed for server-to-server communication (Schneider, 2013). The main advantage of AMQP is in ensuring reliability even after network disruptions. Security can be guaranteed with TLS/SSL over the TCP protocol.

3.4.10 DDS

Data Distribution Service (DDS) is a publish-subscribe, bus-based, data-centric protocol developed by the Object Management Group (OMG). DDS uses a decentralised broker-less architecture. Contrary to MQTT and AMQP, the communication between publisher and subscriber is direct, making it ideal for real-time IoT and M2M communications. It can be used with different transport protocols such as UDP, TCP and IP Multicast, being at the same time language and operating system agnostic. It can be secured using DTLS and TLS respectively.

3.4.11 Remarks

Brokered protocols do well in disseminating data to servers, allowing multiple devices to exchange messages with a central server. Request/response messaging such as CoAP and REST provide the means for a representation of data in well known HTTP model (e.g. webpages). MQTT, CoAP, XMPP, and DDS provide lightweight characteristics that allow them to easily be implemented in constrained devices, levering the message exchange among devices, servers and users. While AMQP provides a cross-platform messaging protocol to enhance IoT bussiness to bussiness data exchange. Table 3.3 provides a brief comparison between the cited IoT messaging protocols, where, R/R and P/S mean respectively: Request/Response and Publish/Subscribe, MC and DC mean: Message-centric and Data-centric, and D2S, D2D, S2S mean respectively: Device to Server, Device to Device and Server to Server communication, the latter corresponding to the type of application for these protocols accordingly to (Schneider, 2013).

Table 3.3: Comparison of messaging protocols

Protocols	Transport	QoS	Architecture	Orientation	Security	Application
CoAP	UDP	Yes	R/R	MC	DTLS	D2S
MQTT	TCP	Yes	P/S	MC	TLS	D2S
XMPP	TCP	No	R/R P/S	MC	TLS	D2S
RESTful	HTTP	No	R/R	MC	HTTPS	D2S
AMQP	TCP	Yes	P/S	MC	TLS	S2S
DDS	UDP/TCP	Yes	P/S	MC	TLS	D2D

3.5 Smart Home Gateway

IoT products for Home automation are starting to appear in great number in the current market. The ubiquitous presence of smartphones lets consumers control and monitor anything through their phones. Thus, the sentence *"Control your device from anywhere"* has become a successful marketing slogan for smart home IoT products. Normally these devices can only be controlled by the phone application provided by the manufacturer. A problem with this is that the user can not control all their devices via a unique interface. In addition, IoT products usually keep a permanent connection with the vendor's cloud server in order to send data and listen for commands. Figure 3.6, depicts the common schemes that the IoT products – in the context of the home automation – use to communicate with the user and server.

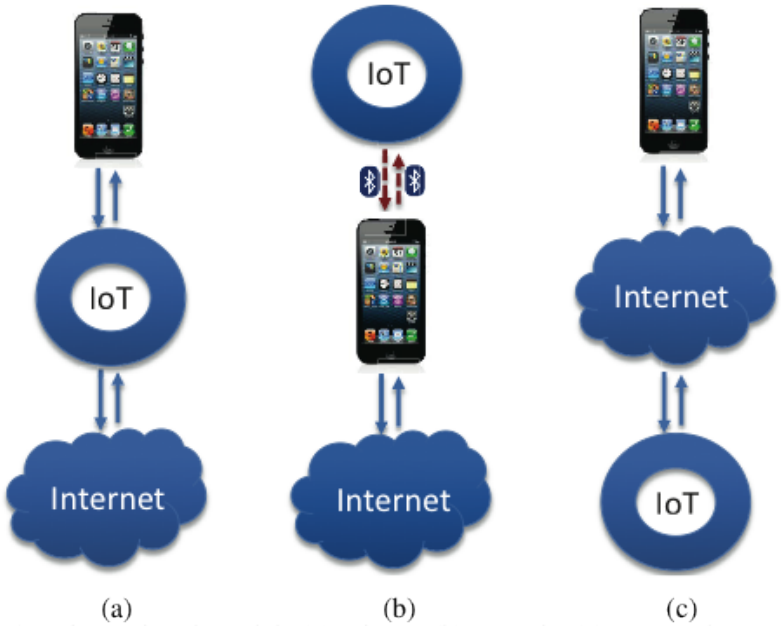


Figure 3.6: Operational model: (a) Direct, (b) Transit, (c) External Server (Notra et al., 2014)

The idea behind the Smart Home Gateway is to enable the control and monitoring of a multitude of smart home products from temperature sensors, smart socket, lighting systems, alarm systems, etc, through a unique interface (Kovac et al., 2014). This section provides an overview of the open-source openHAB platform which fulfils the role of a Smart Home Gateway. The openHAB platform is later used in the demonstration case study (Chapter 5).

3.5.1 OpenHAB Framework

The open Home Automation Bus (openHAB) platform (OpenHAB, 2015) is an open source software project that focuses on integrating all things around home automation within a unique platform. OpenHAB aims to be vendor-neutral as well as protocol and hardware agnostic. It is capable of running in any device system that is able to run a Java Virtual Machine (JVM), e.g. with Linux, MAC or Windows operating system. Providing user interfaces (UIs) for mobile device platforms (i.e. apps for Android and iOS) and web browser. OpenHAB is a pure Java solution based on the Equinox OSGI[5] framework. The openHAB runtime is mainly composed of the Equinox[6] OSGI framework and the Jetty[7] as a web server. It can integrate multiple heterogeneous devices by using bindings. Bindings provide the necessary logic to enable the devices to communicate with the openHAB runtime. The openHAB framework currently supports more than 50 technologies and systems such as the MQTT, TCP, UDP, eBUS, Belkin Wemo smart plug, Phillips Hue Lighting system, etc, (OpenHAB, 2015).

Summarizing, the openHAB platform has the following features:

- *Rules:* OpenHAB allows the use of rules that can be triggered with different types of events such as device status changed, time or system events.
- *Sitemaps:* sitemaps are files that are used to configure the UIs.
- *User Interface:* Multiple and custom UIs can be used at the same time in the openHAB runtime.
- *Authentication:* Username and password can be used to authorize the access to the

[5]The OSGi technology is a set of specifications that define a dynamic component system for Java, http://www.osgi.org/Technology/WhatIsOSGi

[6]Equinox implements the OSGI framework in the ecliplse project, http://www.eclipse.org/equinox/

[7]Jetty is a Web Server is a project of the Eclipse foundation, http://www.eclipse.org/jetty/about.php

different UIs.

- *Confidentiality:* HTTPS (i.e. TLS over HTTP) can be used to secure the communications between the user and the openHAB server.
- *Development:* OpenHAB provides an IDE (i.e. openHAB designer) to configure rules, sitemaps and main runtime configurations.

Communication Mechanism

OpenHAB has two different internal communication channels, namely an asynchronous event bus and a stateful repository, which can be queried. The event bus is the base service of openHAB and it is used to provide a common channel to all parts (bundles) of the openHAB runtime to publish and subscribe to events, i.e. commands and status updates. The stateful repository keeps track of the current status of all items for the eventual use from the UI or the automation logic execution engine. Figure 3.7 illustrates the generic openHAB communication mechanism.

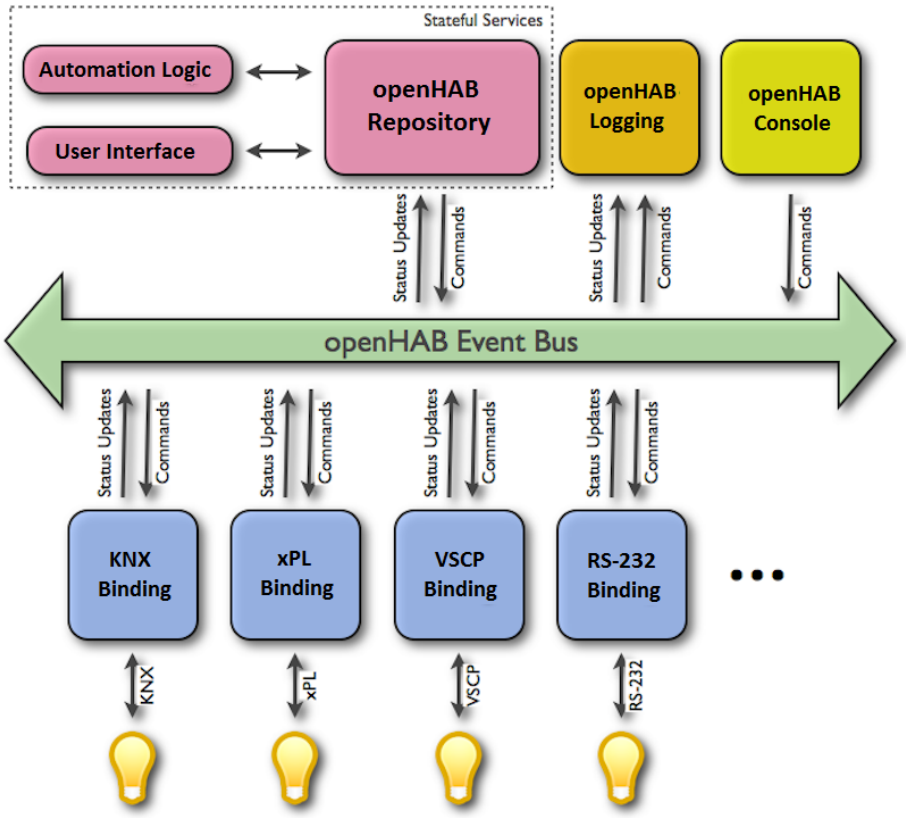


Figure 3.7: Generic openHAB communication mechanism (OpenHAB, 2015)

3.5.2 Similar Projects

OpenHAB is one of many projects that aim to provide a universal solution to integrate all smart home systems. Recently, openHAB provided its core framework to the Eclipse Foundation forming the Eclipse SmartHome. It provides a flexible framework based on the openHAB project[8] to build smart home solutions. In the following table 3.4 a number of smart home solutions are depicted.

Table 3.4: Smart Home Gateway platforms

Project name	**Project type**	**Platform type**	**Programming language**	**Technology support**	**Description**
openHAB	open source	software	Java	50+	`www.openHAB.org`
Domotiga	open source	software	Gambas	50+	`www.domotiga.nl`
SmartThings	proprietary	gateway	n.d.	Z-Wave, Zigbee	`www.smartThings.com`
yetu	proprietary	gateway	Java	50+	`www.yetu.com`
QIVICON	proprietary	gateway	n.d.[9]	50+	`www.qivicon.com`
Home Assistant	open source	software	Phython 3	50+	`www.home-assistant.io`
The Things System	open source	software	JavaScript	50+	`www.thethingsystem.com`
Ninja Blocks	open source	hardware/ software	JavaScript	20+	`www.ninjablocks.com`
Vera	proprietary	gateway	n.d.	only Z-Wave enabled devices	`www.getvera.com`

[8] `http://www.eclipse.org/smarthome/`
[9] not defined (n.d.) in the description of the system.

3.6 Cryptography in the IoT

3.6.1 Public Key Cryptography

Public key cryptography or asymmetric cryptography consists of a pair of keys mathematically related, namely a public and private key. The private key is kept secret while the public key can be freely distributed. If the keys used are large enough, it is infeasible to derive the private key from the public key. Thus, the security of the encryption scheme relies on the key length and the computational work in breaking the ciphertext (i.e. encrypted text). Data can be encrypted with the public key of the destination device with the assurance that only the destination device that is in possession of the correspondent private key is able to decrypt it (Mendes, 2013).

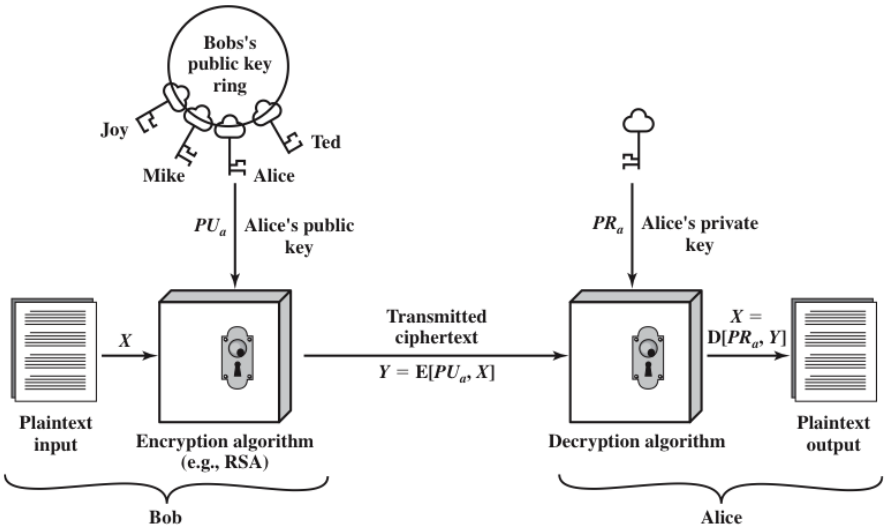


Figure 3.8: Encryption with public key (Stallings, 2010).

As illustrated in Figure 3.8, Bob encrypts the plain text X (i.e. unencrypted text) with Alice's public key and thus only Alice can decrypt it. The message's confidentiality is provided but not authenticity as Alice's public key is available to everyone including eavesdroppers that could forge/falsify Alice's public key. Certificate Authorities (CA) can be used to verify the relationship between the public key and identity of Alice (Nguyen et al., 2015). If Bob creates a message digest[10] from the message to be sent and encrypts it with his private key, Alice will only be able to decrypt it with Bob's public key. The

[10]Is the product of a message that passed through a one-way (hash) function, represented by a string of digits.

integrity of the message can be verified by Alice by decrypting the message digest and comparing it with the message digest produced from the received message. If they match, the message is guaranteed to be safe from alterations. Therefore, the message is authenticated both in terms of source, hence Bob is the only one that could encrypt that message, and in terms of data integrity, as to be able to alter the message, access to Bob's private key is needed. This process is known as digital signature (Stallings, 2010).

The public key algorithm's computational complexity and energy consumption make it undesirable to be applied to resource-constrained devices. Public key algorithm techniques, such as the Diffie-Hellman (DH) key agreement protocol or RSA (Ron **R**ivest, Adi **S**hamir and Leonard **A**delman) signatures are computationally intensive and require tens of seconds up to minutes to perform encryption and decryption operations in resource constrained devices (Sen, 2010). Nonetheless, recent studies (Liu et al., 2012; Ye et al., 2014), have shown that it is feasible to apply public key cryptography in resource constrained devices. The Elliptic Curve Cryptography (ECC) is a discrete logarithm-based cryptosystem (Lauter, 2004), compared with the RSA offers the same security level for a smaller key size. For example, RSA Security recommends for data protection beyond the year 2010 a minimum key size of 2048 bit key for RSA, which is the equivalent to ECC with 224-bit key (Sen, 2010).

3.6.2 Symmetric Encryption

Symmetric encryption, also referred to as single-key encryption, is the simplest forms of encryption. It involves the use of a single shared key between the two communicating nodes which is used both for encryption and decryption. Advanced Encryption Standard (AES) is the most common algorithm used in symmetric encryption. One of the main challenges of the single-key encryption is how to securely distribute the shared key between the two communicating hosts (Sen, 2010). Furthermore, if a node/host gets compromised an attacker can get access to all the other nodes which share the same key. Key management solutions need to be adapted to the constrained nodes of the IoT in order to provide a secure distribution of secret keys among the nodes (Roman et al., 2011). Nevertheless,

due to its simplicity and small amount of calculation, symmetric encryption becomes more suitable for devices with limited computational power and storage capacity (Jing et al., 2014). Table 3.5 compares the symmetric and asymmetric algorithms strength accordingly to their key size, based on a NIST[11] recommendation, (Barker et al., 2012).

Table 3.5: Key sizes for equivalent security levels (Barker et al., 2012).

	Algorithm	**Key lenght (bits)**		
Symmetric key	AES	128	192	256
Public key	ECC	256	384	512
	RSA	3072	7680	15360

The Table 3.5, shows that the key size of AES and ECC algorithms scale linearly, while RSA does not. As the computational power of the computers increases, so as the demand for a bigger key size, therefore the use of RSA in resource-constraint devices may not be feasible. Table 3.5, demonstrates that a symmetric key algorithm is better suited for constraint devices than the public key algorithms, and within the latter, the ECC offers the smallest key sizes. As the majority of the IoT devices are expected to be resource-constraints the use of a lightweight encryption algorithm is preferred.

[11] National Institute of Standards and Technology

Chapter 4

Security in the IoT

4.1 Security Challenges

This section aims to point out the main challenges to security in the IoT context. Due to their characteristics, traditional security methods can be difficult to apply due to the following issues.

Resource Constraints: The majority of the devices that will constitute the IoT will be designed to work with reduced-size hardware and low-power. Thus, IoT devices are expected to be severely constrained in terms of memory, computing capability as well as power capacities (Lee et al., 2014; Petersen et al., 2014). Due to these constraints, it is difficult to directly employ the conventional security mechanisms. In (Keranen et al., 2014), the IETF (Internet Engineering Task Force) provides a number of basic terms in order to classify the constrained-node networks and help in the standardization of the constrained devices.

Heterogeneous Communication Protocols: One major limitation for the end-to-end security between end-devices and the Internet applications, is the use of different protocols that interconnect the end devices. This heterogeneity of the networks may pose a threat to the establishment of security protocols, making it harder to apply end-to-end security solutions, coordination and interoperability (Lee et al., 2014; Heer et al., 2011).

Unreliable communications: The majority of the communications in the IoT will be

based on connectionless protocols which do not guarantee reliability of packet delivery (e.g. UDP). Therefore the packets may need to be retransmitted and error handling schemes need to be employed, leading to a higher overhead of the communication in constrained-node networks (Lee et al., 2014; Sen, 2010).

Energy Constraints: Depending on the type of application, IoT devices are expected to operate with battery power, consequently limiting the energy available for their communication, computations, and storage. This rises the risk of exhaustion attacks that force devices to stay awake consuming resources. Furthermore, the use of security methods adds computation, storage requirements, and communication overheads, which drastically compromise the available energy (Lee et al., 2014).

Physical access: Devices in the IoT are likely to be left unattended most of the time, making them vulnerable to physical attacks that may include tampering the device or destroying it (Lee et al., 2014; Abomhara and Køien, 2014).

Data privacy: The Internet of Things will enable a wider collection of personal data. In the traditional Internet, privacy threats are mostly directed to the Internet user, while in the IoT the threats can come up for people not using any IoT service (Atzori et al., 2010). The collected data can hold information about a person that can lead to the inference of his/her habits. As information storage costs decrease, it is likely that the data collected will be indefinitely stored, making it difficult to control the disclosure of personal information. As there are no clear policies on data handling for the IoT, the user might not know if is data is being stored indefinitely, deleted, protected, anonymised or shared with third party services or business. The IoT raises the need for new solutions to ensure the privacy of their users for which further research effort is needed.

4.2 Security Requirements

The IoT environments will collect, monitor, and analyse sensitive information about people and business, as well as in some cases actuate physically in the environment. The communication medium through which information travels must be secured from any attempt to disclose, alter, or damage the user's information or environments they are

linked to. The IoT communication security implies the provision of security services such as confidentiality, authentication, integrity, authorization, freshness, and availability (Heer et al., 2011).

4.2.1 Confidentiality

Confidentiality prevents the disclosure of information to unauthorized persons, parties or systems. Confidentiality is an important desired security feature in IoT although it may not be mandatory in scenarios where the information is already public (e.g. temperature of a street) (Lopez et al., 2009). The confidentiality of data can be ensured with the use of data encryption mechanisms, such as public key cryptography (e.g. RSA, ECC) or symmetric key cryptography (e.g. AES) (Suo et al., 2012). Although these mechanisms can ensure the confidentiality of sensed data, the confidentiality sensitivity at the sensor node can be considered relatively low. Hence an attacker can place his own sensor and get the same data (Mayer, 2009).

4.2.2 Integrity

Integrity prevents the data transmitted from falsification and modification by unauthorized persons. Message integrity is, in most cases, a mandatory security step to provide reliable services to IoT users (Abomhara and Køien, 2014). The integrity of device software is also important. The modification or falsification of the local or transmitted information can lead to the damage of control systems, facilities, and human lives (Islam et al., 2012).

4.2.3 Authentication and Authorization

In order to ensure that the communications are being established between two trusted peers (e.g. client and server), authentication and identification is desirable (Roman et al., 2011). Two peers with shared keys are the simplest form of authentication as only these peers can decrypt the other's message. Authentication can be a viable measure against attacks (e.g. fake node) that aim to impersonate one of the peers (e.g. server, thing, user). Authorization allows that only authenticated entities can access the resources that they

have been authorized for.

4.2.4 Availability

Availability ensures that authorized users and devices always have access to the services they need. Malicious attackers can prevent the access of authorized entities to their sources; this type of attack is also known as denial-of-service. The non availability of certain critical systems can lead to the economic damage as well as affect the safety of individuals when timely actions are crucial (e.g. hospital devices) (Islam et al., 2012).

4.2.5 Freshness

Guarantees that the data received is recent, ensuring that no adversary replayed or generated old messages. Without the guarantee of data freshness, an intercepted command message (i.e. a message with an order for an action) could be resent by an attacker and still be considered a valid message by the recipient (Lopez et al., 2009).

4.2.6 Privacy

Privacy policies ensure the protection of information as well as to what extent the information should be used or shared with third parties (Weber, 2010). Privacy of collected data can be ensured by means of minimizing the data collected, anonymizing the data, periodic deletion of data and provision of transparent policies on how and which data are being collected and stored for each user.

4.3 Locating Risks in the IoT Architecture Layers

The security threats can be associated to the different layers of the architecture of the IoT: device layer, network layer, and application layer. In the following sections, an attempt to organize threats per layer was done.

4.3.1 Device Layer Risks

The Device layer consists of all kinds of devices (e.g. sensors and actuators), some with the purpose of collecting data from the environment, control and transmission of

collected data from different nodes to the network layer. The majority of these devices do not have a human interface and are often deployed in unmanned monitoring sites. In this respect, attackers can easily compromise security gaining access to the equipment, control or physically damage them (Zhao and Ge, 2013).

The following text enunciates several kinds of attacks that can occur in the Perception Layer (Zhao and Ge, 2013):

Jamming: it is a type of attack that consists of the emission of radio signals with a goal of interfering with the radio frequencies that a network's nodes are using (Wang et al., 2006; Sen, 2010). A powerful jamming source can potentially disrupt the communications in the entire network. As the data transmission gets disrupted, the node might have to retransmit the failed messages, if an attacker is persistent enough it could be possible to compromise the battery of the target devices. Thus, jamming could lead to a serious denial of service (DoS) of the targeted network (Lee et al., 2014).

Tampering: given physical access to devices, an attacker can inject malicious code through the debugging interface of the device and extract security information such as pre-installed encryption keys or other data on the device and/or duplicate a device which the attacker controls (Lee et al., 2014). This type of attack not only compromises the targeted device but also compromises the security of other devices within the network.

Collision: occurs when two nodes attempt to transmit on the same frequency at the same time (Wang et al., 2006). The collision can result in the data packets being altered and later discarded as invalid. An attacker listening to the communication medium can send data packets at the same time as proper message is being transmitted causing a collision. Repeated collisions can lead to a DoS attack affecting the availability of the network (Ashraf and Habaebi, 2015).

Exhaustion: results as a consequence of the previous attacks. Repeated collisions and jamming attacks lead to the disruption of the communication medium causing the retransmission of data packets and depletion of the energy resources for battery powered nodes (Ashraf and Habaebi, 2015).

Cryptanalysis attacks: focus on deciphering encrypted communications more specifically the ciphertext (encrypted message), i.e. finding the encryption key to obtain plaintext (unencrypted message) containing the information of interest, e.g. Man-in-the-middle attack, Ciphertext-only attack, Known-plaintext attack, etc (Babar et al., 2011).

Side Channel Attack: it is based on the fact that logic operations have physical characteristics that are measurable. A side channel attack makes use of this to extract information in the process of the encryption, such as time consumption, power consumption statistics, radiation of various sorts, sound, etc in order to discover the device key (Zhao and Ge, 2013).

Software Attacks: as some devices/things will be embedded with microprocessors that run a certain OS, the implementation vulnerabilities of that OS can be exploited by attackers through the communication medium, e.g. virus, Trojan horse programs, worms, trapdoors, etc (Babar et al., 2011).

4.3.2 Network Layer Risks

The Network layer provides the communication between the perception layer devices and the application layer services through the Internet. The security issues in this layer are common to the traditional security issues of the Core Network (Internet), but with the development of IoT, this layer will carry huge data traffic. Due to the heterogeneity and complexity of IoT network, the network layer is vulnerable to attacks. The main security issues of the network layer are as follows (Zhao and Ge, 2013):

Routing attack: Targeting the route information while it is being exchanged between nodes is the most direct attack against a routing protocol in any network. Routing information can be spoofed, altered, or replayed, in order to create routing loops, attract/repel network traffic, extend/shorten source routes, etc. Examples of routing attacks are Sinkhole attack, Selective forwarding, Wormhole attack, Sybil attack (Wang et al., 2006).

Selective forwarding: In a multi-hop network an attacker may create a malicious node

to selectively forward some messages and drop others. Another form of this attack is called black hole where all messages are dropped thus compromising network availability (Wang et al., 2006).

Sinkhole: In a Sinkhole attack a compromised node is used to attract the surrounding nodes by forging routing information. This leads to the surrounding nodes routing all the traffic through the attacker's node that can drop some or all traffic like in the previous attack (Wang et al., 2006).

Sybil attack: A single node creates its own multiple identities and presents it to other nodes in the network using them to gain a disproportionately large influence.

Hello flood: Some routing protocols require nodes to broadcast hello messages to announce themselves to their neighbours. A node which receives such a message may assume that the sender is within radio range and therefore attempts to use the route as a communication path. An attacker may use a high-powered transmitter to trick a large area of nodes into believing they are neighbours of that transmitting node. This leads to nodes out of radio range to send packets to the attacker, which are likely to be lost (Singh et al., 2010; Wang et al., 2006).

Eavesdropping: Is the most common attack and it consists of listening to the communications in the network. If the messages are exchange unencrypted an attacker could easily understand the content(Abomhara and Koien, 2014).

Traffic analysis: A step beyond simple eavesdropping is traffic analysis. The IETF defines it as the act of acquiring knowledge of information by inference from observable characteristics of data flow. This includes frequency and size of messages, flow direction as well as the identity of sender and receiver. This could be used, for instance, to identify the data sink (e.g. gateway) in a WSN as target for further attacks.

Man-in-the-middle attack: An attacker impersonates both peers in a conversation, leading the victims to communicate directly to the attacker while assuming that they are communicating with the legitimate peer. The attacker then has full control over what is communicated between the peers and is able to read, alter, or destroy messages (Noack, 2014).

4.3.3 Application Layer Risks

In the application layer different security issues occur for different areas of application; thus security in the application layer becomes more complex and burdensome, but it still can be summed up to some common security problems (Zhao and Ge, 2013):

Unauthorized access: Access by someone trying to impersonate a user or a thing as to gain access control or insert fake data to the system. Thus, it is vital that effective identification and authentication methods should be put in place.

Software vulnerabilities: Non-standard codes written by programmers can raise vulnerabilities creating backdoors (trapdoors) to the application that can be easily discovered or accessed by an attacker with bad intentions (Babar et al., 2010).

Password attacks: Users might have to introduce account name and password in order to authenticate themselves with the application. One might be allowed to use blank or weak password (e.g. 123) or use default credentials given by the service provider. This opens a door for an easy discovery of the user account password and thus getting full access to the devices he/she might own.

Flooding (or Denial of Service) attack: Targets the application server with continuous overwhelming requests, thereby depriving the legitimate user from accessing the service.

Malicious code injection: The application system on the end-user side can be compromised with malicious codes through known vulnerabilities. Malware such as viruses, worms, Trojan horses, spyware, malicious adware, and other programs to interfere with systems (Ning et al., 2013).

4.4 Security Countermeasures

This section aims to summarize existing countermeasures for the attacks that target the different layers of IoT described in the previous section. Table 4.1 summarizes the attacks and some corresponding defence mechanisms in each IoT layer.

Table 4.1: Attacks and Defences in IoT (Sen, 2010).

IoT Layer	**Attacks**	**Defences**
Perception Layer	Jamming	Frequency-hopping spread spectrum (FHSS) Code spreading
	Tampering	Tamper-proof case
	Collisions	Error-correcting codes (ECCs)
	Exhaustion	Rate limitation Detect and Sleep
	Side Channel Attack	Randomizing Masking Blinding
	Software Attacks	Update software (i.e. with software patches, firmware patches)
Network Layer	Selective forwarding	Multipath routing
	Sinkhole	Authentication Intrusion Detection System (IDS)
	Sybil	Authentication Encryption
	Hello flood	Puzzle schemes Pairwise authentication Geographical routing
	Eavesdropping	Message encryption
	Man-in-the-middle	Pairwise authentication Message Encryption
Application Layer	Unauthorized Access	Authentication Access Control Intrusion detection systems
	Password Attacks	Strong passwords Password policies
	Flooding (DoS)	Puzzle schemes
	Malicious code injection	Anti-virus programs Firewalls Intrusion detection systems

The following subsections explain in detail the countermeasures listed in Table 4.1.

4.4.1 Device Layer Measures

Jamming: A typical defense against jamming involves the variation of spread-spectrum communications as frequency hopping and code spreading (Wang et al., 2006).

Frequency-hopping spread spectrum (FHSS) is a method of transmitting signals by rapidly switching a carrier among many frequency channels using pseudo-random sequence known to both the transmitter and the receiver. This makes it impossible for a potential attacker to predict the frequency selection sequence, consequently preventing the jamming of the frequency being used at a given point of time (Sen, 2010). However, the level of protection of this method is low due to the fact that an attacker may jam a wide section of the frequency band compromising any selected frequency.

Code spreading is another technique for defending against jamming attacks. However, it requires greater design complexity and energy and thus it is not very suitable for constrained resource devices. Generally, the implementation of anti-jamming measurements have a high cost for the wireless nodes resources. In order to maintain low cost and low power requirements, the wireless devices are limited to single-frequency use and are therefore highly susceptible to jamming attacks.

Tampering: Assuming that the manufacturers of the device are trustful, and the device is not counterfeit, tamper-proof case (i.e. the node's physical package) is one defense to this attack. However, due to the additional cost involved, it is usually assumed that the devices are not tamper-proofed (Wang et al., 2006). The protection level of a tamper-proof case is low due to the fact that the device can be compromised in its production and or be tampered before it gets to the final destination. Nevertheless, the probability of a device being tampered once deployed can vary depending on its accessibility, thus being higher in public spaces compared to private spaces.

Collision: A common defence against collisions is the use of **error-correcting codes (ECCs)**. These codes work better for low level of collisions caused by environmental or probabilistic errors. Since collisions caused by an attacker do not follow certain patterns, it is reasonable to assume that an attacker will always be able to corrupt more than what can be corrected (Wang et al., 2006), consequently providing a low

level of protection. Despite the low level of protection these codes also add additional processing and communication overhead.

Exhaustion: Can be done by applying **rate limits** to the MAC (Media Access Control) admission control, in order to limit the amount of requests one can admit thus saving the impact in battery-powered devices. A second measure is to use **time-division multiplexing** where each node is allocated a time slot in which it can transmit automatically limiting the rate at which a node can transmit. However, it is still vulnerable to collisions.

Side Channel Attacks: In the context of the Smart Homes, side channel attacks are unlikely to happen, due to the high expertise (skills) and physical presence needed from the attacker. However in other applications (e.g. Industrial domains) it might be easier to perform such attack. Common measures against this attack are **Randomizing, Blinding**, and **Masking**. The actual solutions try to make the power consumption of the cryptographic device independent of the signal values at the internal circuit nodes by either randomizing or flattening the power consumption. However such techniques do not provide perfect security and they increase the required number of measurements (Zhou and Feng, 2005).

Software attacks: Due to the heterogeneity of software that will be implemented in the end-devices, it is harder to come up with a specific countermeasure. **Software patches** can prevent novel attacks by correcting known vulnerabilities of the software that a device has. Therefore keeping the end-devices up to date with the latest security software patch is essential to the security of the end-devices.

Feasibility of protection mechanisms for the device layer

Table 4.2 summarizes the feasibility in terms of cost and protection level of the countermeasures for the device layer threats.

Table 4.2: Feasibility of protection mechanism in terms of computational cost and protection level at the device layer.

IoT Layer	Protection Mechanism	Cost	Protection Level	Description
Device	Frequency-hopping spread spectrum (FHSS)	High	Low	Additional processing Unable to protect the whole frequency spectrum.
	Code spreading	High	Low	Complex design and extra energy consumption. Unable to protect the whole frequency spectrum.
	Tamper-proof case	High	Medium	Adds production costs. Can always be compromised.
	Error-correcting codes (ECCs)	High	Low	Additional processing and communication overhead. Poor defence against high levels of collision.
	Rate limitation	n.d[1]	n.d.	
	Detect and Sleep	Low	Medium	Detection should be easy to implement and the sleep mode saves battery. Protects de node from exhaustion by sleeping, thus avoids the attack and saves battery by becoming inactive.
	Randomizing	n.d.	Low	Additional processing overhead Increases the number of measurements an attacker has taken.
	Masking	n.d	Low	Increases the number of measurements an attacker has taken.
	Blinding	n.d	Low	Increases the number of measurements an attacker has taken.
	Up to date software	n.d	High	Software vulnerabilities can be mitigated or erased as soon as they are discovered.

4.4.2 Network Layer Measures

Selective forwarding countermeasure: Is the use of multipath routing which sends data over **multiple paths** increasing the chances of reaching its destination. However this technique wastes power on redundant paths and consumes additional network bandwidth.

Sinkhole countermeasure: Can be done by applying an **intrusion detection system**

[1] "n.d." means not defined or found within the literature.

(IDS) that verifies the route quality (Ashraf and Habaebi, 2015). This method can be applied in high power devices such as the gateways, thus making it viable. An IDS that aims the detection of sinkhole attacks is described in (Krontiris et al., 2008). **Authentication methods** can also be used to disclose the compromised node.

Sybil countermeasure: Authentication and encryption can be used to prevent sybil attacks in the network (Padmavathi et al., 2009). Even if the Sybil is able to impersonate a node and stay undetected, it has no immediate way of accessing the corresponding public / private key pair (Noack, 2014).

Hello flood countermeasure: Includes **authentication mechanisms**, and **puzzle schemes** which impose that a connecting node solves a puzzle with a certain difficulty. The idea is to discourage unnecessary connections as the requester needs to waste resources to solve the puzzle. The problem occurs if the attacker does not rely on constrained resources but in a more powerful node (e.g. laptop). (Koh et al., 2013) propose a mechanism to increase the difficulty of the puzzle for malicious nodes. **Geographical routing protocols** require each node to know its location and be able to communicate that location to other nodes. This lets the nodes drop hello messages that are not within their communication range (Raymond and Midkiff, 2008).

Traffic Analysis countermeasure: Even if the messages are encrypted, an attacker can deduce significant information by monitoring traffic volume and traffic path information finding possible targets in the network. Countermeasures are **hiding the identity of users** and **obfuscation of message characteristics** (Noack, 2014), (Deng et al., 2005).

Eavesdropping countermeasure: An immediate measure is **encryption of messages**. Even if the eavesdropper can get the message, the cryptographic algorithms make it infeasible to recover the message content (Noack, 2014), (Xiaohui, 2013).

Man-in-the-middle (MITM) countermeasure: Message Integrity prevents that MITM attack modifies the message content by ensuring mechanism that verify if the message has been altered in the way. **Message Confidentiality (Encryption)** ensures that the message content is protected (e.g. session keys). **Mutual**

Authentication mechanism ensures that both communicating nodes are who they claim to be.

Feasibility of protection mechanisms for the network layer

Table 4.3 summarizes the feasibility in terms of cost and protection level of the countermeasures for the network layer threats.

Table 4.3: Feasibility of protection mechanism in terms of computational cost and protection level at the network layer.

IoT Layer	Protection Mechanism	Cost	Protection Level	Description
Network	Multipath routing	Medium	Low	Consumes additional network bandwidth and power in redundant paths Not valid if no more than one routing path is available
	Intrusion Detection System (IDS)	Low	n.d.[2]	IDS are expected to be deployed in rich resource devices
	Puzzle schemes	High	Low	Adds computational overhead. Not effective against rich-resource attackers.
	Geographical routing	n.d.	n.d.	
	Message Encryption using ECC	Medium	High	Message Encryption with asymmetric key consumes a considerable amount of resources compared with a symmetric key Depending on the key size it is infeasible to break
	Message Encryption using SKC	Low	High	Symmetric key encryption requires few computational resources Depending on the key size it is infeasible to break

4.4.3 Application Layer Measures

Unauthorized access countermeasure: Authentication can be used to prevent unknown individuals from accessing protected resources. **Access control** policies guarantee that unwanted actions take place, limiting the actions an attacker can

[2] "n.d." means not defined or found within the literature.

perform if the first barrier is successfully broken. Finally **Intrusion Detection** can monitor and detect suspicious activities (Farooq et al., 2015). A more detailed survey on the different intrusion detection models is provided by (Patcha and Park, 2007).

Password Attacks countermeasure: One measure would be to define a minimum password length and complexity (e.g. at least one number, one capital letter, etc) in order to ensure that the user defines a **Strong Password**. Second, define policies to limit the amount of retry attempts that can be used to guess the password.

Flooding (or Denial of Service) attack: As in the Hello flood attack this type of DoS attack can be mitigated using **puzzle schemes** (Wang et al., 2006; Heer et al., 2011).

Malicious code injection countermeasures: Known countermeasures against malicious codes are **antivirus software**, **firewalls** and **intrusion detection**.

Feasibility of protection mechanisms for the application layer

Table 4.4 summarizes the feasibility in terms of cost and protection level of the countermeasures for the application layer threats.

Table 4.4: Feasibility of protection mechanism in terms of computational cost and protection level at the appplication layer.

IoT Layer	Protection Mechanism	Cost	Protection Level	Description
Application	Authentication	Low	Low	Application layer is a rich resource layer Good protection against any user/device impersonation
	Access Control	Low	High	Application layer is a rich resource layer
	Intrusion Detection System (IDS)	Low	n.d[3]	IDS are expected to be deployed in rich resource devices
	Strong passwords	Low	High	A strong password makes it infeasible for an attacker to break it, an to get access to vital action or information
	Puzzle schemes	Low	n.d	
	Anti-virus programs	low	High	Provides a good and updated protection against malware.
	Firewalls	High	Low	Provides a good protection against unauthorized users

[3]"n.d." means not defined or found within the literature.

4.5 Secure Gateway

Due to the major difficulties in implementing traditional security measures into the constrained nodes of the IoT, a solution lies in the use of a rich-resource device such as a gateway. The gateway, as a rich-resource device, can easily implement existing security standards such as TLS (Transport Layer Security) protocol; which makes use of public key and symmetric key cryptography to secure the communication channel. The communications between the Internet host and the gateway can be secured by the use of traditional security measures and the interactions between the gateway and the IoT nodes could be secured with the use of simpler security approaches (e.g. symmetric encryption) (Hyuk Park et al., 2009).

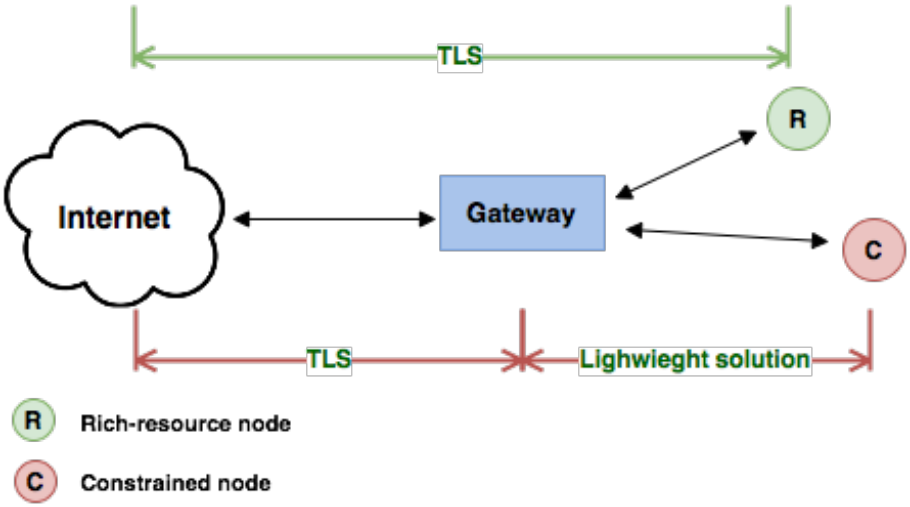


Figure 4.1: Gateway solution to secure constrained nodes.

Figure 4.1 illustrates the gateway use to secure the communications to constrained nodes. This solution aims IoT applications where devices resort to the use of gateways to access the Internet (e.g. Smart Home), for applications where devices have a great mobility this approach might not be applicable.

A problem with this approach is that the gateway becomes as single point of failure. If an attacker manages to gain access to the gateway all the device network information will be compromised, moreover if the gateway fails all devices lose their connectivity to

the exterior. Nevertheless, the gateway can provide other services such as user authentication, authorization, storage of data, and management of the sensor/actuator nodes. User authentication can be achieved with the use of public key certificates, user/password credentials or other complex authentication mechanisms and authorization can be applied with the use of access control rules to define that only authorized users have access to the devices. The gateway solution can provide a feasible security solution to protect the constrained nodes of the IoT by taking care of the complex and heavyweight security measures of the traditional Internet. Adapted from (Hyuk Park et al., 2009).

Chapter 5

Demonstration Case Study

5.1 Introduction

The objective of this chapter is to demonstrate the vulnerabilities that the IoT devices might have and propose tools to secure them within the context of a Smart Home. A second objective is to demonstrate how different devices built with different technologies can coexist and interoperate under the same framework rather than working solely on proprietary systems. In this chapter an analysis is done of the network behaviour of an *"on market"* IoT device, namely the Orvibo Smart Socket, highlighting the lack of security and demonstrating how an attacker can compromise this device. Secondly the openHAB framework is used to secure the smart socket. Finally, a second device is implemented and integrated with the openHAB framework.

5.2 Orvibo Smart Socket

For the security analysis of an *"on market"* IoT product, the Orvibo WiFi Smart Socket model WiWo-S20 was chosen. One can find this socket in Amazon, eBay, Aliexpress or other online shops. It is a popular device among other smart devices for home automation as it can be found in the Amazon's webpage[1],thus it becomes a good sample for a security analysis. The power socket can only be controlled by a smart phone or

[1] `http://www.amazon.com/gp/bestsellers/hi/6478741011/ref=pd_zg_hrsr_hi_1_4_last#2` Access date: 28/08/2015

tablet device through the WiWo App[2]. The phone application discovers the socket and after the user types the password of the local network, the socket connects to the router. The user can then turn it on and off and set timers. The socket has also the ability of being controlled remotely, i.e. outside from the local network.

Figure 5.1: Orvibo WiFi Smart Socket, Model: WiWo-S20.

The used Orvibo Smart Socket has the following specifications:

Table 5.1: Orvibo WiFi Smart Socket specifications

General Specifications	
Model	S20
Voltage Range	AC 100-240V
Max load power	2000W
Wifi	802.11, b/g/n
Security protocols	WPA-PSK/ WPA2-PSK/ WPA/ WPA2/ WEP/ WPS2/ WAPI
Encryption Type	WEP / TKIP / AES
Communication Protocol	UDP (User Datagram Protocol)

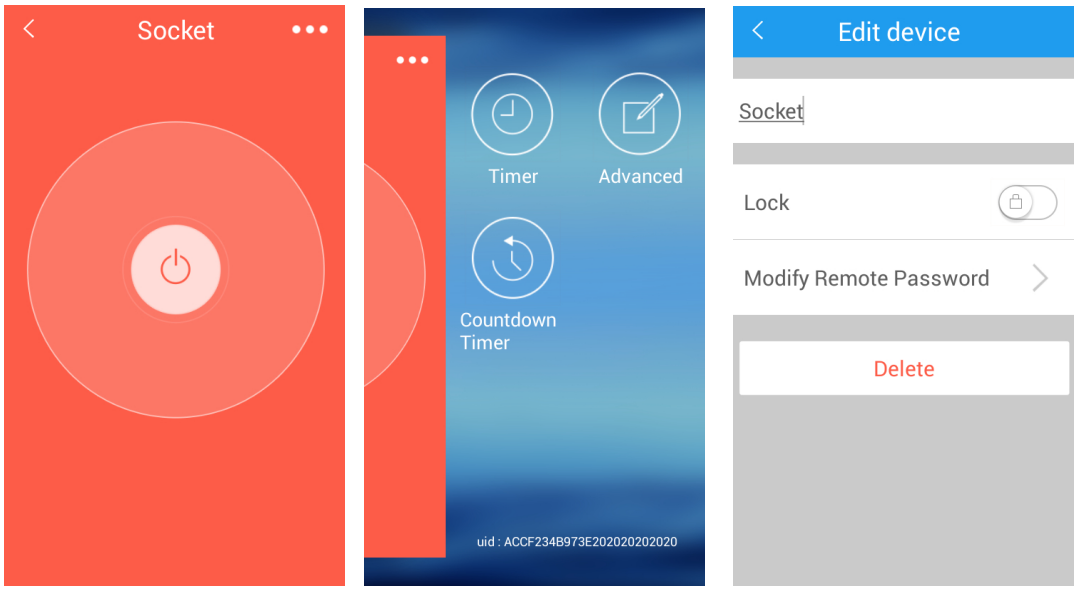
5.2.1 Controller Application

To better understand the security aspects of the power socket it is important to understand the functionalities of the controller application (i.e. WiWo App) explained as follows:

- The application (App) can discover any socket that is in the same network as the App.

[2] `https://play.google.com/store/apps/details?id=com.orvibo.irhost`, access date: 1/08/2015.

- Multiple users can control one socket.
- The application comes with a lock function that prevents other users from finding the socket.
- A remote password, that can be modified, prevents users that previously gained access to the socket from being able to control it from an outside network.
- Every device comes with a default remote password equal to 888888.



(a) On/Off button (b) Options menu (c) Advanced settings

Figure 5.2: WiWo App interface with (a) On/Off button, (b) Options menu, (c) Settings where is possible to set the Lock function and Remote Password.

In the Figure 5.2(b) we can see identified as *uid* the MAC address[3] (i.e ac cf 23 4b 97 3e) of the device, that is present in most of the communications with the device.

Summing up the security measures implemented on the smart socket are (Figure 5.2(c)): a) the ability to hide the socket from future discoveries with the lock function, and b) prevent unwanted remote control (i.e from an outside network) through the use of a "remote password", that the user is encouraged to modify.

[3]Media Access Control address is a unique identifier.

5.2.2 Socket Communication Scheme

In order to better understand the communication between the user (i.e. App) and the socket, Wireshark[4],(Wireshark), was used to capture and analyse the network traffic. After a brief analysis of the captured traffic, a pattern could be seen as follow:

- All messages are sent in cleartext (i.e. not encrypted).
- The socket periodically broadcasts messages with what appears to be its status.
- The remote access to the socket is made through a server for that purpose.
- The MAC address of the socket is always present in any message except for device discovery message.
- All packets are sent over UDP through the port 10000.

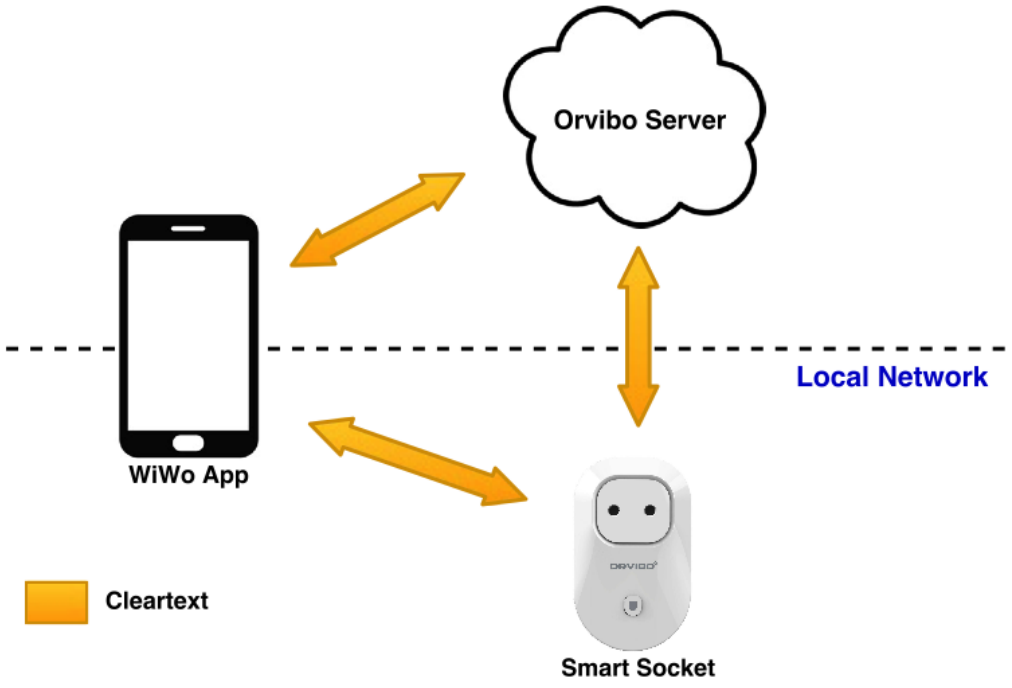


Figure 5.3: Orvibo Smart Socket communication scheme

As illustrated in Figure 5.3, the user either communicates directly with the Smart Socket within the local network or communicates with the Orvibo Server for remote access to the Socket. All messages are exchanged in cleartext.

[4]Wireshark is a network protocol analyser, that can be used to capture and analyse packets being transmitted in a local network.

Local Control

When first deployed, the smart socket has to be configured using the WiWo App in order to connect to the local network. In this phase the user assigns the password of the local network to the smart socket that later connects to the router. Figure 5.4 illustrates the exchanged messages between the power socket and the controller application within the local network.

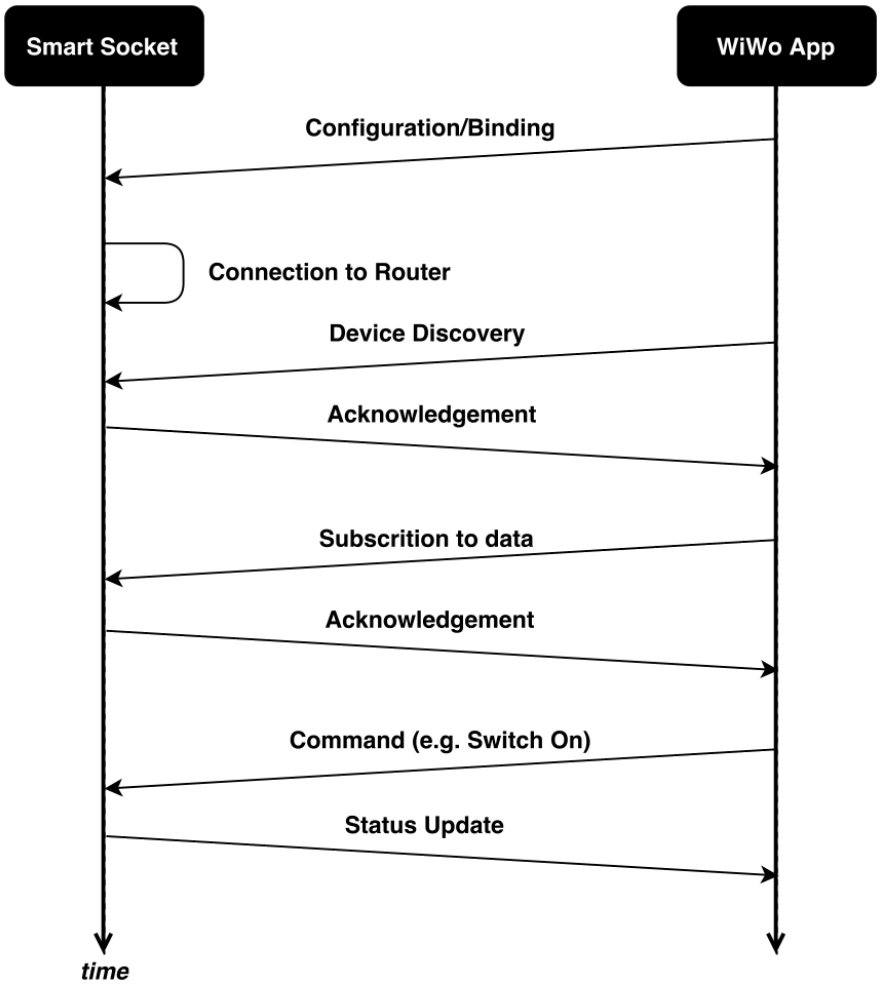
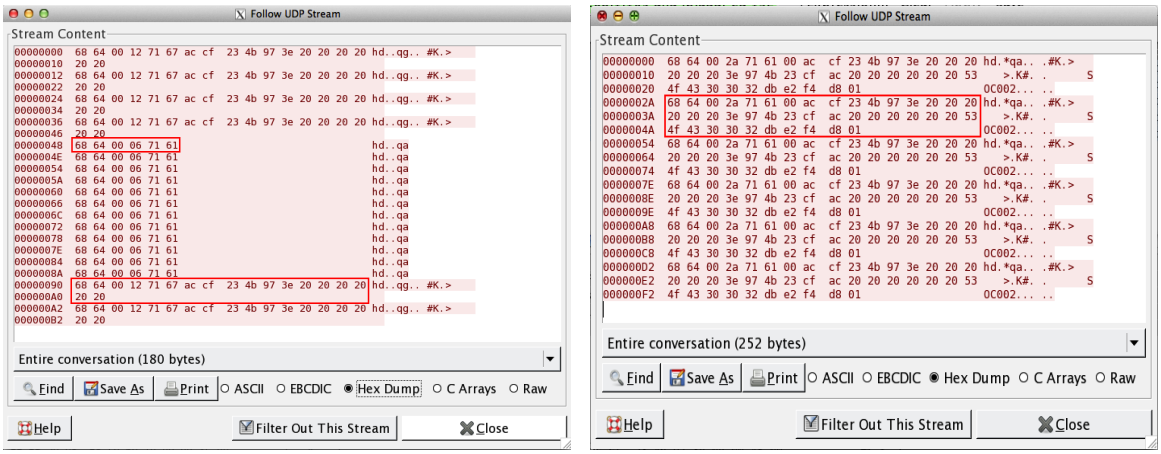


Figure 5.4: Communication diagram of exchanged messages between the Wiwo App and the Smart Socket in the local network

Following the configuration phase, the WiWo App sends a message to discover the device in the local network, message that is replied by the socket.



(a) Device Discovery (b) Reply to Device Discovery packet

Figure 5.5: Wireshark capture of Local Communication between WiWo app and Socket (a) Device Discovery (b) Reply to Device Discovery packet

In Figure 5.5(a), two different *device discovery* messages are apparent. The first is used to discover any socket present in the network while the second is targeted to a specific device; hence it contains the MAC address of that device.

Following this, the WiWo App subscribes to receive data from the socket, which is followed by an acknowledgement message depicted in Figure 5.6, where packets from the WiWo App are marked as *(App)* and as *(Soc)* for packets from the socket.

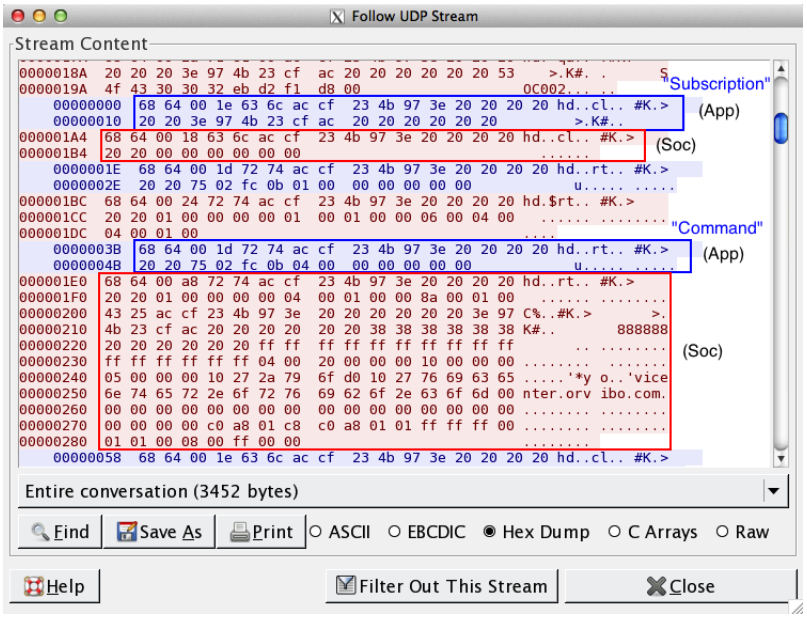


Figure 5.6: Wireshark capture of the Socket (Soc) communicating with WiWo App (App)

Finally the App can send command messages that can be a switch for On, Off, or a

data request (e.g. information about timers) as shown in the Figure 5.6.

In Figure 5.6 the second message from the App (*"Command"*) is some kind of request for data due to the sockets response. In the socket response there is the number *'888888'* that is the default password and *'vicenter.orvibo.com'* that corresponds to the remote server used for remote control of the socket.

Figure 5.7 shows the captured message when the WiWo App issued a command to turn on and off the socket.

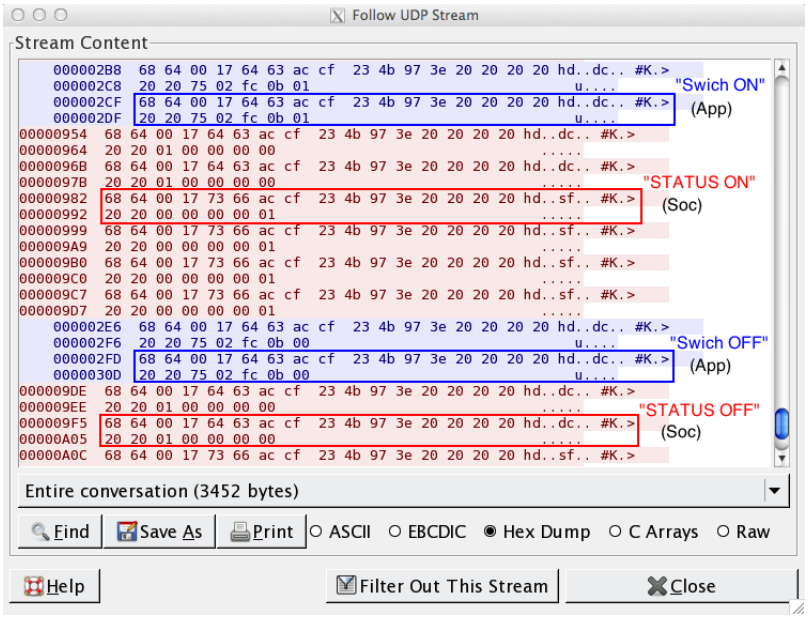


Figure 5.7: Wireshark capture of Socket (Soc) communicating with WiWo App (App)

Note that these messages are exchanged within the local network using UDP packets through port 10000, with both Smart Plug and WiWo App being connected to the same network.

Remote Control

When the user is in a remote location (i.e. in a different network than the power socket) the communications between the user application and the socket are done through a third-party, in this case the Orvibo's server (whose IP address is 42.121.111.208).

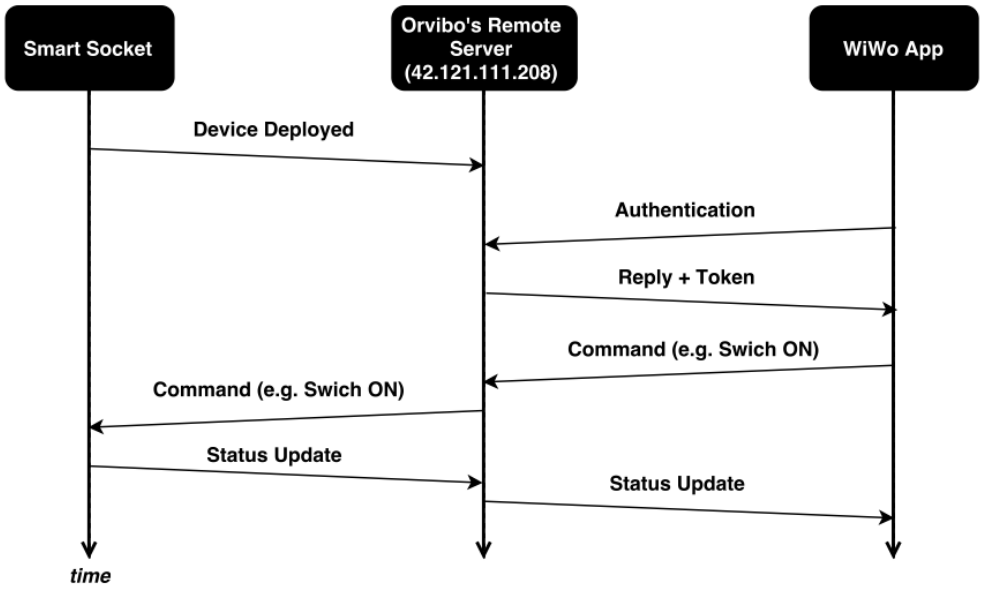


Figure 5.8: Communication diagram of exchanged messages between the Wiwo App, the Smart Socket and the Orvibo's server

At the deployment stage the socket transmits a packet to the server with information about the device. At this point it is assumed that the server collects the IP address of the socket and other relevant information in order to connect to the socket.

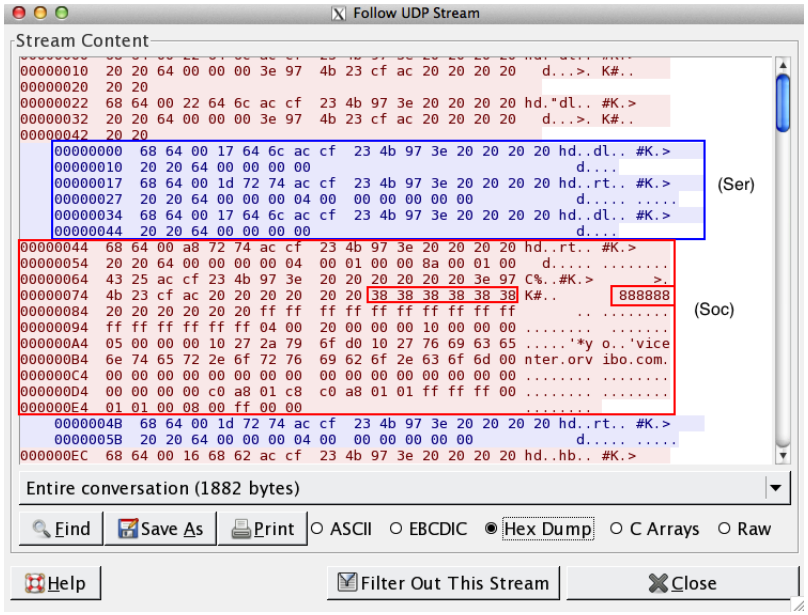


Figure 5.9: Wireshark capture of the Socket (Soc) communicating with the Orvibo's server (Ser)

Figure 5.9 depicts a capture of the communication between the socket and the server. Highlighted inside the socket's message is the remote password (i.e. 888888) that is transmitted in clear text to the server as well as the other messages.

When the user is trying to control the socket remotely, the application sends an authentication message with the MAC address of the target socket and the respective remote password to the server. The server then responds with another message as it can be seen in the Figure 5.10, where the message marked as *(App)* is from the mobile App and as *(Ser)* from the server.

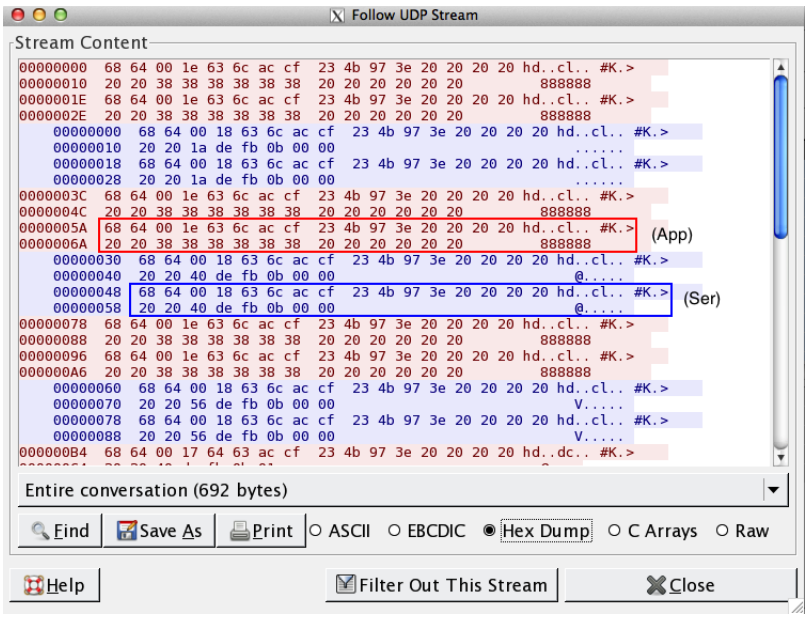


Figure 5.10: Wireshark capture of Wiwo App (App) communicating with the Orvibo's server (Ser), authentication message

Figure 5.10 shows that the communications between the server and the App are done in clear text as well, regardless that both have the computational power to apply encryption algorithms in their communications. Neither the less to say that the remote password is sent in clear text. Someone sitting in the same network as the user just has to sniff[5] the network traffic to get all the information it needs to control the socket remotely, as it will be demonstrated further on.

After the authentication is done, the user can control the socket remotely with the server as the middle-entity. In Figure 5.11 we can see the exchanged control (i.e. "Switch On" and "Switch Off") messages.

[5]The act of monitoring the network traffic, this can be done using Wireshark.

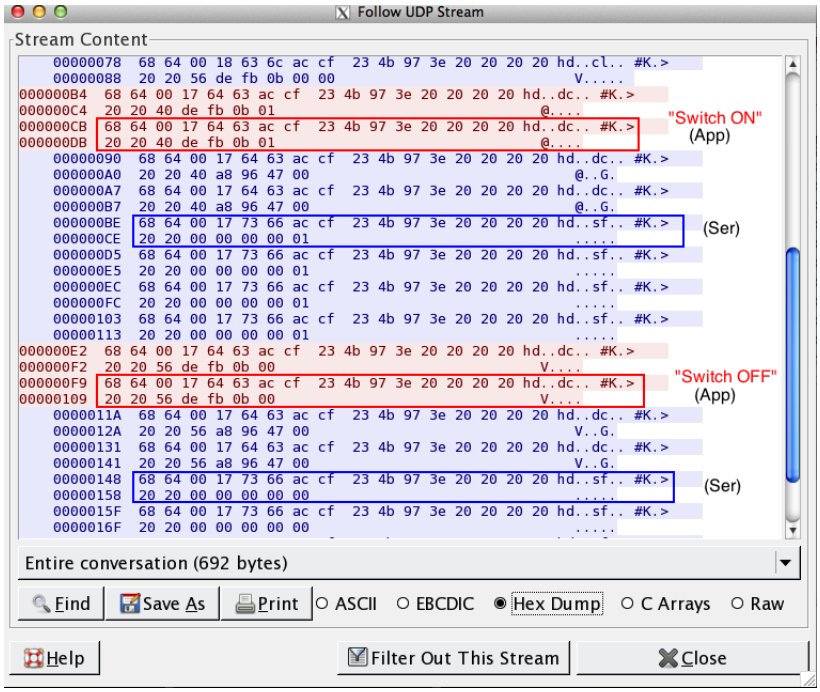


Figure 5.11: Wireshark capture of WiWo App (App) communicating with the Orvibo's server (Ser), control messages

From the messages exchanged in Figure 5.11, one can infer that the last octet of the exchanged messages means the following, *'00'* socket off and *'01'* socket on.

5.2.3 Security Vulnerabilities

After analysing the local and remote communication patterns of the smart socket, it becomes clear that some important security aspects were not taken into account when developing this product. The key issue is the fact that there was no attempt to encrypt the communications, which makes it easy to an attacker to gain control. In the following bullets the protection mechanism offered by the manufacturer and their vulnerabilities are described.

- **Lock Function:** The main purpose of the lock function is to prevent any further device discoveries by other WiWo application to new devices. This means that any new user cannot connect its WiWo app to the socket. However this protection measure has its flaws/vulnerabilities. One can impersonate the IP address of an user that is already subscribed and successfully send control messages to the socket.
- **Remote password protection:** Is meant to protect the socket from being remotely

controlled, but due to the unencrypted communications this password can be easily disclosed.

The security mechanism implemented by the manufacturer works well for individuals that limit themselves only to the use of the WiWo application and have no intention of making an analysis of the traffic generated by this system. The security vulnerabilities that this socket has, makes it easy to be controlled by a low skilled attacker. Table 5.2 summarizes the main vulnerabilities found in the Orvibo Smart Socket.

Table 5.2: Orvibo Smart Socket main vulnerabilites

Vulnerabilities	**Description**
Lack of encryption	The message composition and content can be easily understood by analyzing the communications of the socket.
Unauthorized remote control	The remote password and MAC address of the device can be easily obtained by an attacker whenever the device or application communicates with the server.
Unauthorized local access	One can impersonate the IP address of a subscribed user and successfully gain access to the socket regardless of the lock function.
Privacy issues	The manufacturer does not present any privacy policies about the usage of the data collected or which data is collected. Furthermore, there is no information about the use of a server to which the socket and the application exchanges data.

5.2.4 Hacking the Smart Socket

In this section it will be demonstrated how the Orvibo socket can be controlled by an attacker. From the knowledge acquired from the network behaviour of the socket the Table 5.3 was elaborated with the relevant control messages that allow one to control the socket without the use of the WiWo App.

Table 5.3: Control UDP packets sent by the WiWo App

Command	**Message content (HEX)**
Device Discovery	68 64 00 06 71 61
Data Subscription	68 64 00 1e 63 6c ac cf 23 4b 97 3e 20 20 20 20 20 20 3e 97 4b 23 cf ac 20 20 20 20 20 20
Authentication with Server (Data Subscription)	68 64 00 1e 63 6c ac cf 23 4b 97 3e 20 20 20 20 20 20 38 38 38 38 38 38 20 20 20 20 20 20
Power On	68 64 00 17 64 63 ac cf 23 4b 97 3e 20 20 20 20 20 20 75 02 fc 0b 01
Power Off	68 64 00 17 64 63 ac cf 23 4b 97 3e 20 20 20 20 20 20 75 02 fc 0b 00

Where the majority of the messages follow a similar pattern (Stikonas, 2015):

Magic Key + message length + command id + socket MAC address + rest of the message

The Magic Key[6] is always 68 64 (hexadecimals) that can be used to distinguish these UDP packets from other packets sent over UDP port 10000.

In (Stikonas, 2015)[7], a more detailed analysis of the meaning of each command and how to reverse engineer the socket was done by the author. This section focuses more on how the socket can easily be controlled with the minimum analysis of the packets.

Controlling the Socket

In order to control the socket it is required to know the MAC address of the device so as to be able to control it locally and additionally the remote password in order to control it remotely. To get the MAC address an attacker can issue a Device discovery message that will be replied by any socket in the network. The replied message contains the MAC address needed to compose the remaining messages (see Table5.3).

To start controlling the device the Data Subscription message is required to be sent first. After sending the subscription message and receiving the reply message one can just issue the Power on and Power off commands and successfully be able to control the socket.

Gaining Remote Access

From the analysis of the remote communication it was demonstrated that the WiWo app authenticates (subscribes to data) to the Orvibo's server by sending the MAC address and remote password of the targeted device. One could picture the situation were an individual is sitting in a public network and is trying to control his socket. An attacker eavesdropping[8] that same network would get all the information needed to subscribe to data and control the device. In the following step it will be demonstrated a successful

[6]Term used by Stikonas (2015) to define the first two octets of the UDP packet used in the socket's communication

[7]Referenced by the author a more detailed analysis of the commands and reply packets can be found in `http://pastebin.com/LfUhsbcS#`

[8]Consists in listening to the communications in the network.

attempt to control the socket remotely. For the intended purpose the Packet Sender[9] was used to send UDP packets to the server. By sending the captured authentication message (see Table 5.3) the server replied with the following message (Figure 5.12).

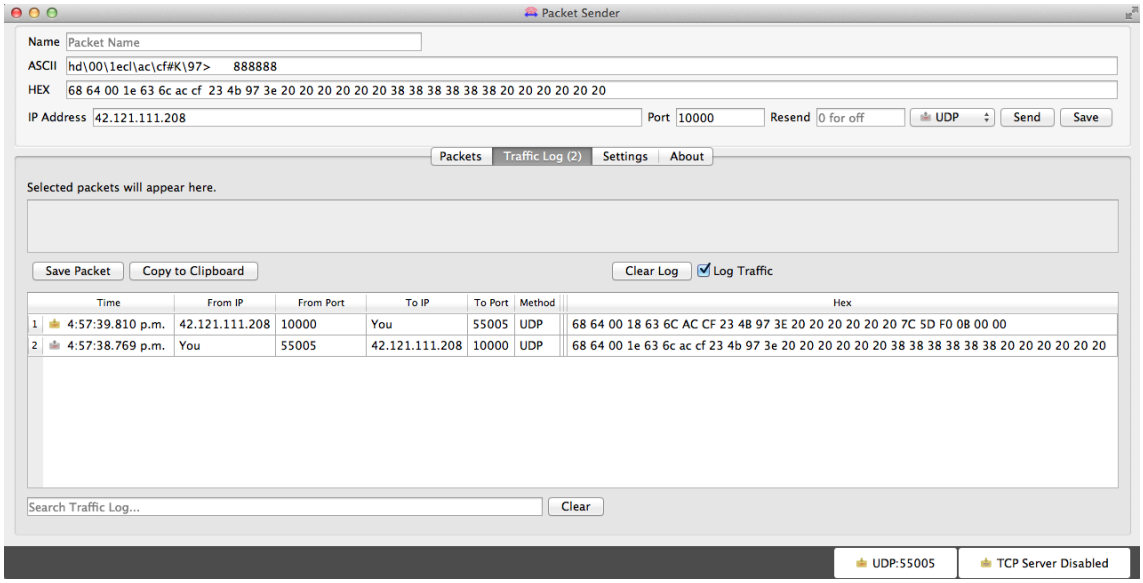


Figure 5.12: Authentication to the Server and respective answer sent over UDP using the Packet Sender

The server reply was understood as the following:

68 64 00 18 63 6C AC CF 23 4B 97 3E 20 20 20 20 20 20 7C 5D F0 0B 00 00

- 68 64 - Magic Key,
- 00 18 - message length,
- 18 63 6C - command id,
- AC CF 23 4B 97 3E - socket's MAC address,
- 7C 5D F0 0B - session token,
- 00 - unknown,
- 00 - socket status, '00' for OFF and '01' for ON.

It was learned that in order to issue Power On/Off command one has to add the session token to the message. With that said, as the socket was OFF we sent a Power On command to the server.

[9] A cross-platform program that allows to send and receive UDP and TCP packets. More information about this tool can be found in https://packetsender.com/

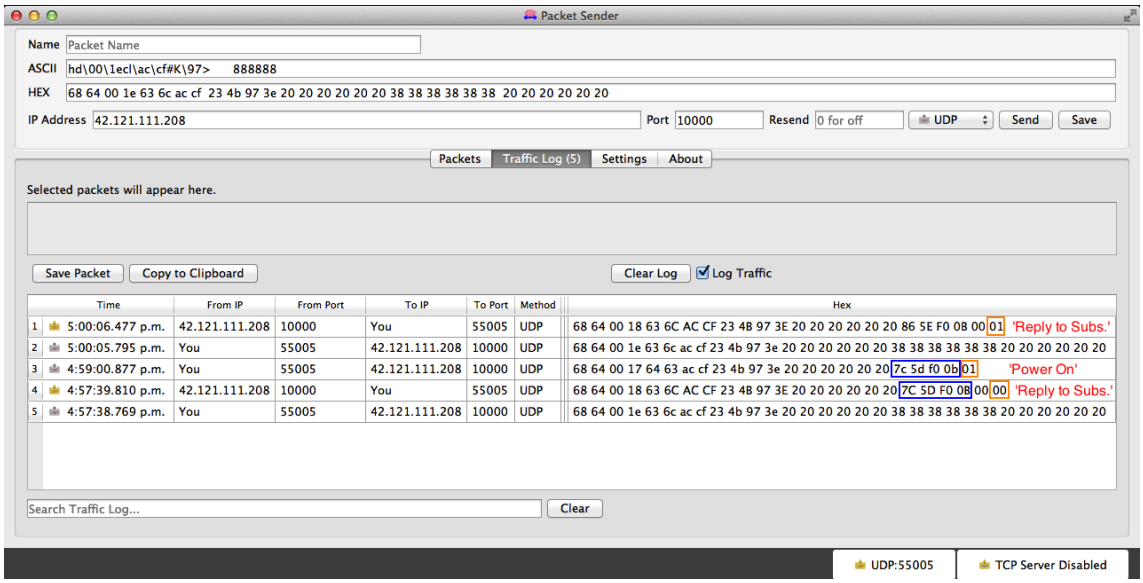


Figure 5.13: Power On and Authentication messages sent to the Server using the Packet Sender

This command made the socket to turn on. Following this, a subscribe message was sent again in order to check the change in state of the socket has it can be seen in Figure 5.13.

5.2.5 Proposed Solution

Faced with the unencrypted communications and how easy it becomes to gather useful data that allows one to control the socket remotely, the blockage of the communications between the proprietary server and the socket is proposed. A set of rules can be put in place where any message coming/going to the server address towards/from the socket is blocked. The common WiFi gateway allows setting of address filtering rules (i.e. firewall settings), that allow one to block any communications coming and going from a specific IP address. This prevents any attempt to remotely control the socket, even if the MAC address and remote password are disclosed. The control of the socket would be then just done within the local network, making it easier to trace any unauthorized access and avoid any related data to go out of the local network.

The secure control of the smart socket can be achieved with the openHAB (open Home Automation Bus) environment which provides the essential tools to integrate the Orvibo Smart Socket and secure the communications with the users remotely and locally. The openHAB provides communication confidentiality (i.e. encryption), user authentication

and data privacy as it does not need to transfer user or device data to a remote server in order to be able to control the device. Table 5.4 summarizes the proposed solutions to secure the communications with the Orvibo Smart Socket.

Table 5.4: Proposed solutions to secure the Orvibo's Socket.

Proposed solution	**Description**	**Observations**
Access control rules	Blocks any communications between the socket and the vulnerable proprietary server (i.e. Orvibo's server).	Prevents unauthorized access to the socket. Preserves data privacy.
openHAB (secure gateway)	Mediates the communications between the user and the socket, establishing a secure channel with the user.	Prevents eavesdropping of communications, unauthorized local and remote access. Maintains device data within the local network.

5.3 Securing the Smart Home with OpenHAB

In this section it will be demonstrated how the Orvibo Socket can be secured with the openHAB[10] home automation software. Furthermore an attempt to demonstrate how openHAB can secure and manage heterogeneous devices is done by integrating and securing a second device with similar functions as the Orvibo Socket but with a different communication protocol.

One of the characteristics of the IoT devices in the current market is that they are controlled and configured with their own app, thus, leading to n applications to control n devices. With the openHAB environment one app controls and manages all devices. OpenHAB thus becomes the only remote interface with these devices.

[10]http://www.openhab.org/

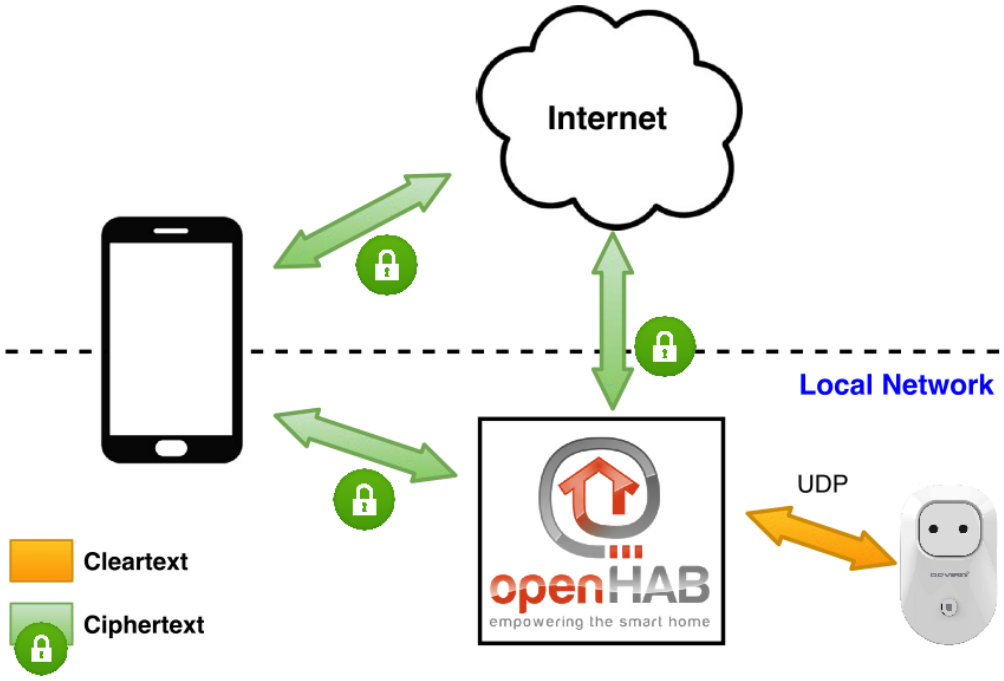


Figure 5.14: Orvibo Smart Socket integrated with openHAB framework, communications chart

With this approach, the users never communicate directly with the device, but with the openHAB server. All communications are secured between the user and the openHAB server (see Figure 5.14).

5.3.1 Security and Privacy with OpenHAB

OpenHAB home automation software provides a set of mechanisms to secure the *Smart Home*, explained as follows:

- **Authentication -** The access to the UI can be secured by the configuration of credentials (i.e. user and password).
- **Authorization -** Multiple user interfaces can be configured and associated to a single user or a group of users.
- **HTTPS -** The confidentiality, integrity and authenticity of the messages exchange between the server and the user are ensured with the use of HTTPS (i.e HTTP over TLS security protocol).
- **Privacy -** The generated data from the devices do not have to leave the private network to an external server, maintaining their functions within the private network.

As seen previously on the analysis of the Orvibo Smart Socket communication, a remote server is used to connect a remote user with the socket. The server then collects the data relative to the socket, which can raise concerns about data privacy. Due to the lack of privacy one cannot know how his/her data are being used. The openHAB framework provides a good protection over the collection of data by providing a full control of the devices without the need of third party servers. Additionally, home network firewalls have to be put in place, blocking any connection with the manufacturer server. Thus enhancing a full ownership of the user data.

5.3.2 Socket Binding

OpenHab provides a set of tools and the environment to integrate any device with its server. Currently openHAB has the ability to integrate to its environment a vast number of IoT devices in the market. This is made possible through some pieces of code that explore the communication protocols of these devices, called *bindings*[11]. Furthermore the openHAB environment can run scripts written in other programming languages, so as for example to help in the binding of a device.

In order to integrate the Orvibo Socket with openHAB the following tools were used:

- **openHAB Runtime:** the server and all the logic to control the home devices. The runtime is basically the core of openHAB were all the core task are performed.
- **openHAB Designer:** the configuration tool that allows one to develop/configure different user interfaces, variables, and rules of the openHAB runtime.

All projects and tools related to openHAB can be found in their site[12]. As learned before, the communications between the Orvibo Socket and the WiWo application are made using the UDP protocol through the port 10000. In order to bind the socket with the openHAB a program written in PERL[13] (Carter, 2014) was used, which creates a UDP

[11] Implement the necessary logic and protocols in order to provide a link to the real hardware devices allowing one to control and configure its device through the openHAB environment as it could do with the manufacturer environment

[12] http://www.openhab.org/

[13] The code was used to send control messages to the Bauhn smart socket which uses the same commu-

socket[14] and sends the necessary packets to the Smart Socket. The message composition was analysed in the previous sections (see Section 5.2.4).

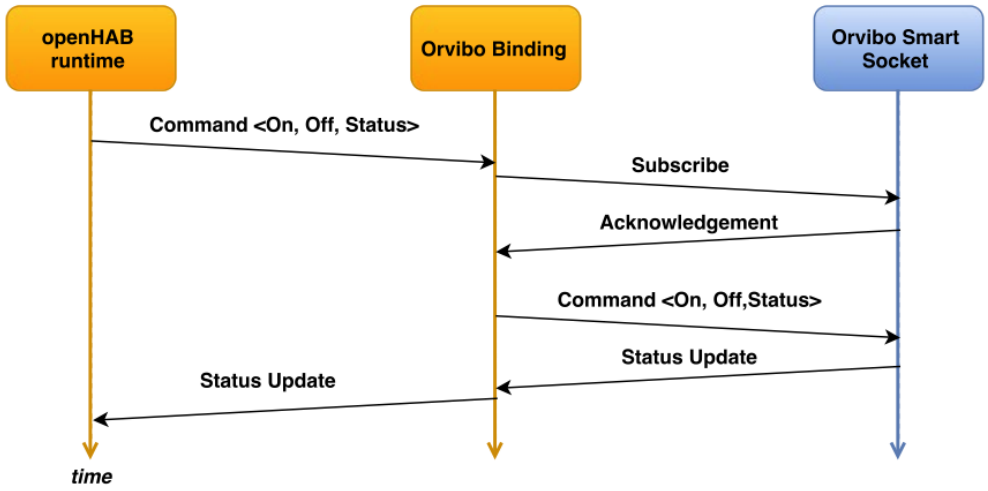
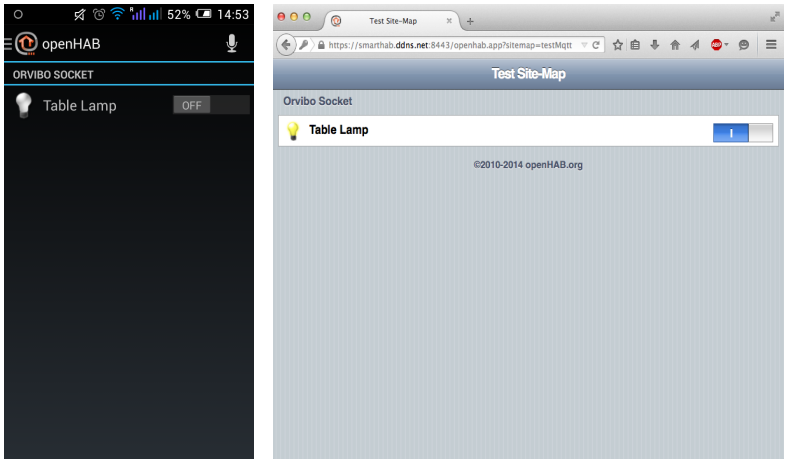


Figure 5.15: Orvibo Smart Socket control with openHAB environment

As illustrated in Figure 5.15, when a command (e.g. button on/off) is triggered in the openHAB runtime (i.e. server), it then runs the *Orvibo Binding* (i.e. the program written in PERL) which subscribes to the Smart Socket and if the subscription is successful it sends the issued command message to the power socket. The power socket then replies with a message from which its state can be read.

nication protocol as the Orvibo Smart Plug

[14] Establishes bidirectional communication between a server and one or more clients. In this case with the UDP protocol.



(a) mobile application (b) browser

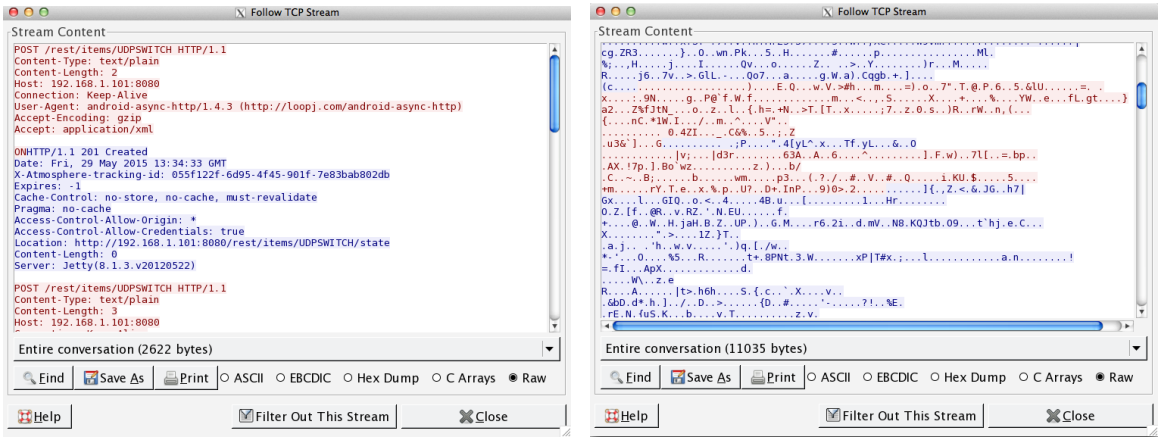
Figure 5.16: Screenshots of openHAB user interface in the a) mobile application, b) browser with the button to control the Orvibo Smart Socket.

The openHAB environment can be accessed by the openHAB's mobile application (Figure 5.16(a)) or simply through a browser (Figure 5.16(b)). In order to access the openHAB server remotely, a dynamic DNS (Domain Name Service) had to be configured, which in this case is *smarthab.ddns.net*[15].

5.3.3 Communications

With the Orvibo Smart Socket linked to the openHAB environment it is possible to see the communicated messages if no security is put in place (Figure5.17(a)).

[15]To access the defined openHAB UI the following url has to be used `http://smarthab.ddns.net:8080/openhab.app?sitemap=testMqtt`.



(a) messages in cleartext (b) messages in ciphertext

Figure 5.17: Wireshark capture of communications between the user and the openHAB server (a) messages in cleartext (b) messages in ciphertext

In Figure (5.17(a)) the security is not enabled and one can see in cleartext the messages exchanged between the user and the openHAB server, making it easy to gather information about the devices that exist and consequently break in further into the server. With the security enabled, the communications are only done through HTTPS (i.e. TLS over HTTP); thus all messages are encrypted as it can be seen in Figure 5.17(b).

5.4 Enhancing the Smart Home

In an attempt to demonstrate the capabilities of the openHAB in integrating different Smart Home devices and keeping them secure, a second device was used. As MQTT is becoming a well known communication protocol in the IoT community due to its simplicity and lightweight, this device was connected through the MQTT communication protocol in order to integrate it with the openHAB environment. The device was assembled in order to behave as a smart socket as the Orvibo socket but with a different communication protocol. The implemented device will be referenced as the Smart Lamp.

Figure 5.18: Smart Lamp composed of NodeMCU devkit, relay, and lamp controlled by openHAB

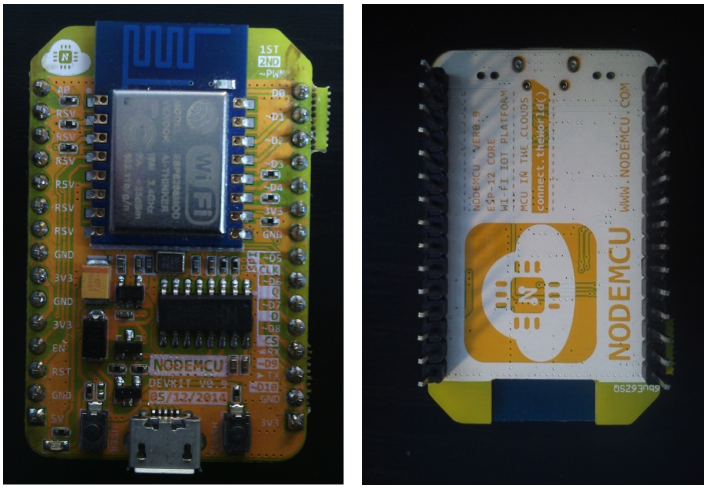
The Smart Lamp main components are:

- **NodeMCU ESP-12:** which is a hardware development platform for the ESP8266 Wifi module.
- **Songle 5V Relay Module:** is basically a relay that opens or closes a circuit, triggered by a 5V input signal.
- **Lamp:** provides the lighting function of the smart lamp.

The most important piece of this device is the Wifi module which conducts all the communications and predefined control actions.

5.4.1 NodeMCU Wifi Module

The NodeMCU is an open-source firmware based on the Lua[16] firmware and development board for the ESP8266 Wifi module (NodeMCU, 2014). With the NodeMCU firmware it is possible to create a HTTP server, UDP client and server, MQTT client, and much more. The ESP8266 WiFi module behaves like an Arduino in the way that it can read from and write to other devices (e.g. sensors, relays, leds, etc). The versatility of this module and its cheap cost is drawing the attention of many IoT developers as it becomes easy to build and integrate IoT devices at a low cost.



(a) front (b) back

Figure 5.19: NodeMCU development board picture from the (a) front, and (b) back of the module

The NodeMCU development board can be plugged directly through USB to the computer from where it can be used to pass the Lua scripts with the functions one expects the ESP8266 WiFi module to perform. For the development of the code in Lua the ESPlorer IDE (ESPlorer, 2014), was used. It behaves like a normal IDE, giving the environment to write Lua scripts and export directly to the module.

[16]Lua is a lightweight, embeddable scripting language developed by PCU-RIo. `http://www.lua.org/` accessed on: 2/09/2015.

5.4.2 MQTT

The communication protocol used in this Module for the purpose of integrating it with the openHAB environment was the MQTT protocol (MQ Telemetry Transport). As explained in chapter 3, the MQTT protocol is a publish-subscribe protocol, in which a client (e.g. the Wifi Module) publishes and or subscribes to a topic (e.g. /WifiModule/BedLightStatus) to a MQTT Broker (i.e. the server). In this study the Wifi Module and the openHAB are both MQTT clients.

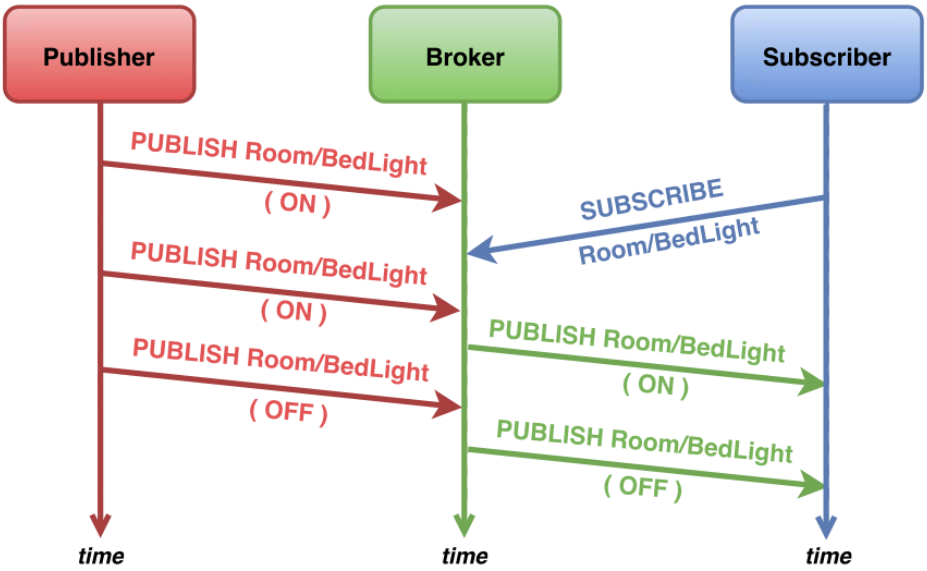


Figure 5.20: MQTT publish-subscribe protocol

When the openHAB wants to change the state of the Wifi Module for example to turn on or off the relay, it first publishes a topic (e.g. Room/BedLight) for which the Wifi Module is subscribed to. When a message is published on that topic by openHAB (e.g. ON or OFF) the Wifi Module will receive that message and perform the action that it was programmed for. The opposite occurs, when the openHAB is subscribed to a topic published by the Wifi Module. Figure 5.21 depicts how the two devices are integrated with the openHAB framework. The MQTT broker can be within the local network or on an outside network (securing with the use of TLS protocol and user/password authentication). This can provide scalability if needed. As an example devices from different locations can publish or be subscribed to by an external MQTT broker.

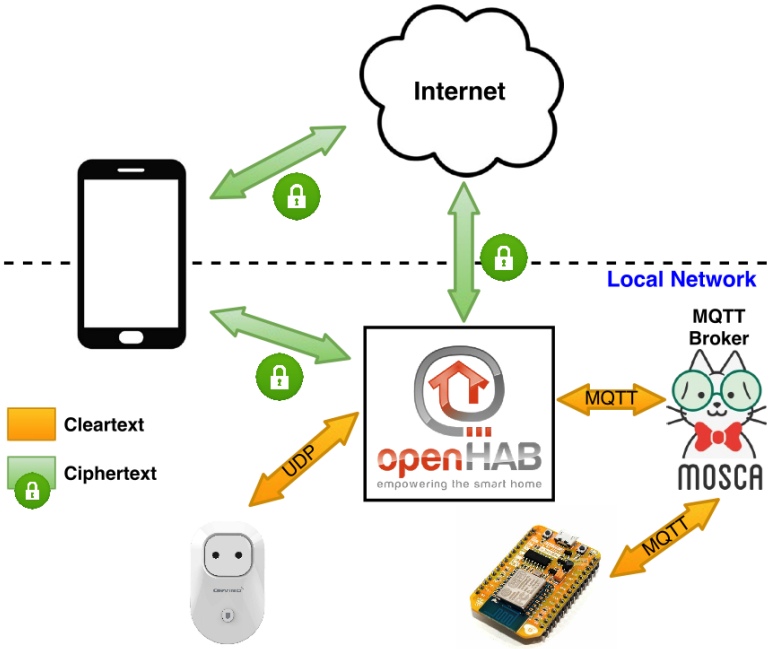


Figure 5.21: Communication diagram of Wifi Module and Orvibo Smart Socket integrated with openHAB

Figure 5.22 illustrates the communications of the Smart Lamp and Orvibo Smart socket with the openHAB framework.

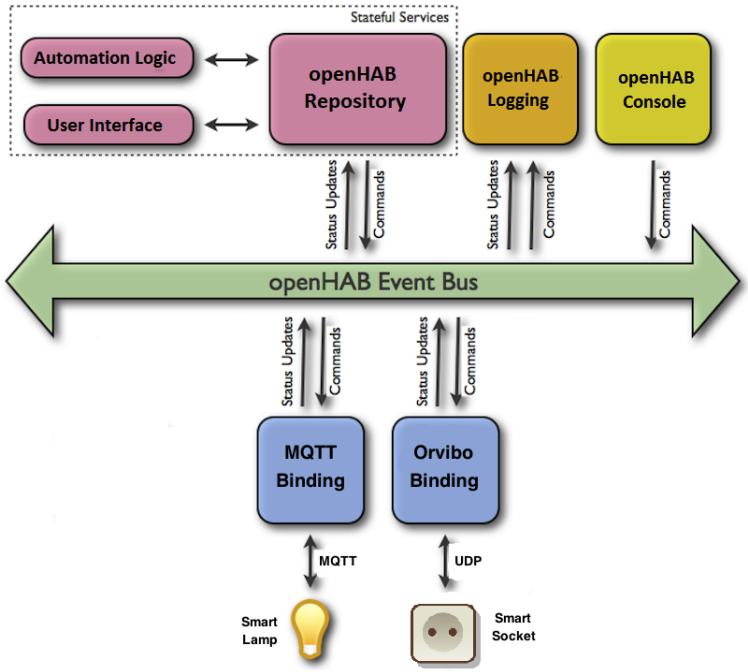


Figure 5.22: Bindings for the Smart Lamp and the Orvibo Smart Socket integrated in the openHAB framework

Furthermore, it depicts the communication links that the "bindings" provide between

the real hardware and the openHAB framework.

5.5 Conclusions About the Case Study

In this case study it was demonstrated how easily it is to compromise a device when basic security measures are neglected. This can lead to additional threats to the security and privacy of the environments where these devices exist in.

The Orvibo Smart Socket is a small sample of the vulnerable IoT devices that are currently on the market. This comes to prove the ease with which the IoT companies secure their products as cited in chapter 2 (Capgemini, 2014; HP, 2014a). The security measures matter, as any vulnerable device can become a backdoor to other devices in the network.

Although the Orvibo Smart Socket comes with some security measures (i.e. the lock function and the remote password) these lost their ability to secure the device as a consequence of the unencrypted communications it uses.

In order to secure the Orvibo Smart Socket, network level security was proposed. The use of rules to block the vulnerable communications between the device and the proprietary server was applied, guaranteeing that no attacker can reach the device remotely through the server. With openHAB the communications with the user were secured by the use of credentials (i.e. username and password) over a communication path secured by the Transport Layer Security (TLS). The TLS helped keep the confidentiality, integrity and authenticity of the messages exchanged between the user and the openHAB environment.

The openHAB environment proved to be a suitable protection barrier for the unsecured devices within the Smart Home. It demonstrated that network level security solutions do well to secure the IoT devices when the implementation of traditional security measures (e.g. cryptographic methods) are not feasible. Another feature of the openHAB is that it allows the integration of different devices with different protocols thus enhancing the potential of the IoT environment. This provides an environment where multiple heterogeneous devices can be monitored and controlled from anywhere at anytime within a secure and private interface.

The privacy of the data is also guaranteed with the blockage of the communication

with the proprietary server. It might be necessary to implement such rules when the data privacy policies are questionable or absent.

The MQTT communication protocol is a good example of a standard that was made with the IoT in mind. It offers a lightweight communication, scalability and secure communications for the IoT devices. It also has the capability of being a secure gateway for the IoT devices that implement this protocol. Another feature of this protocol is that it provides a way for multiple devices to communicate with each other. This can augment the interoperability from thing-to-thing, thing-to-services and service-to-services.

This technology could be used for example to enable the openHAB environment to exchange data/information with other services. The established communication between the openHAB environment and the Smart Lamp provides a good example of a thing-to-service and service-to-thing communication.

The ESP8266 Wifi module proves to be a good hardware platform in enhancing the Internet of Things, providing the accessibility and sufficient versatility to integrate different sensors and actuators into an IoT device although it still does not yet support traditional cryptographic protocols due to resource constraints. Thus, lightweight security solutions need to be achieved so as to provide embedded security to this platform.

In this demonstration case study it was possible to secure the devices from external attacks by securing the communication between the user and the server (i.e. the openHAB). However, the communication between the devices and the openHAB are still vulnerable, due to the fact that none of the devices apply cryptographic protocol. If an attacker manages to infiltrate the local network, these devices would then be vulnerable to malicious actions by the attacker. This is where *"security by design"* enters and could be a last barrier against any attacks, independent of the consumers ability to keep their network safe. The security of the devices should be part of the designing process of any IoT product. IoT devices with weak security measures and privacy policies can hinder the trust that the general public has towards the IoT, thereby slowing down its development and making it harder to patch later on.

Chapter 6

Conclusions and Future Work

6.1 Final Conclusions

The Internet of Things concept is here to stay and it will enhance the interconnection of a broader spectrum of devices and eventually open doors for innovation. The application scenarios for the different areas that the IoT can bring to our society is vast. However, the IoT is still in its early age of adoption and for now the majority of the efforts are in linking the real world with the digital world and providing connectivity to services in the Internet. In a future phase the IoT is expected to be more interconnected and smart by augmenting a context-aware and semantic-oriented approach.

The IoT devices pose different security challenges compared to the common Internet-enabled devices used presently. The miniaturization of the devices lowers their prices and leverages their applicability in numerous scenarios, but it also creates new barriers for the adoption of the pre-established security measures. The Internet of Things does not just involve the sensors and actuators but the user's devices and the remote servers that are used to control and monitor them which need to be secured as well. All these factors expand the "attack surface", which in turn makes the job of securing an IoT system more demanding.

In this thesis it was demonstrated that IoT manufacturers are still relaxed towards applying the necessary security measures and in enforcing the best practices in privacy policies. Also most of the companies have a lack of security experts on their side to advise them with the best practices.

The technologies that will enable the IoT are maturing and a diverse offer exists. The vast application areas will lead to the need of different devices supported by different communication and application technologies. There is a lot of effort in providing technological solutions for the upcoming development of the IoT and the research for the different layers of the IoT is growing. There is currently a lot of hype towards anything related with the Internet of Things from business, hardware solutions manufactures, academia, industry and consumers.

Hardware platforms are being developed with great fanfare and currently are easy to find such solutions that embrace the Internet of Things.

In terms of operating systems there is the need for lightweight solutions compatible with the expected constraints of the majority of the IoT devices, allowing these to become smarter and implement lighter security solutions as well.

Network access technologies such as Zigbee are opening the door for the interoperability and connectivity to reach the most constrained nodes, while the message exchange protocols help leverage the cooperation across 'things', services, business, and people.

In the context of the Smart Home, currently the Internet of Things is mainly *things-oriented* where manufacturers are excited about bringing up their new IoT device that will enrich someone's daily life, but are leaving them locked in silos where interoperability with other IoT devices is not possible.

Solutions such as openHAB come to help integrating the numerous heterogeneous IoT devices that currently are built in silos. The heterogeneity of these devices are a good thing in one way, but cooperation with other technologies and devices must take place.

The analysis of the security applied in one IoT product, the Orvibo Smart Socket, confirmed the relaxed position that the companies adopt toward security. The lack of security measures opens the door for abuse as it was illustrated with the demonstration case study.

In the implemented prototype case study it was possible to reach a solution to partially secure the communications with this device by using the openHAB server and access control rules. The openHAB server was used to mediate the communications between the user and the device and secure the communications with the user. Access control

rules were put in place to block the communication between the Orvibo Smart Socket and the correspondent proprietary remote server, as it turned out to be an open door for unauthorized access. A main vulnerability was left behind which is the communication between the openHAB server and the smart socket. The security of this vulnerability zone depends on how resilient the local network is to an infiltration by an attacker.

Security and privacy standards and solutions need to be formulated and applied to the Internet of Things in order to bring trust and a faster development of the concept. An IoT with weak security measures and privacy policies can hinder the trust and expectations that the general public has towards the Internet of Things. This could slow down its development and make it harder to patch later on.

6.2 Future Work

Lightweight security protocols, i.e. adequate cryptographic technologies for the constrained nodes still needs further research. In Chapter 5, the devices used were left without a cryptographic solution due to resource-constraints and in the case of the Orvibo Smart Socket due to its proprietary nature. Future work shall focus on the research and implementation of lightweight security measures to a resource-constrained IoT device, so as to provide a complete end-to-end security to the IoT network.

Bibliography

Abomhara, M. and Køien, G. M. (2014). Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security*, 4:65–88.

Abomhara, M. and Koien, G. M. (2014). Security and privacy in the internet of things: Current status and open issues. In *Privacy and Security in Mobile Systems (PRISMS), 2014 International Conference on*, pages 1–8. IEEE.

Aggarwal, C., Ashish, N., and Sheth, A. (2013). The internet of things: A survey from the data-centric perspective. In Aggarwal, C. C., editor, *Managing and Mining Sensor Data*, pages 383–428. Springer US.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols and applications. *Communications Surveys Tutorials, IEEE*, PP(99):1–1.

Alhamedi, A. H., Snasel, V., Aldosari, H. M., and Abraham, A. (2014). Internet of things communication reference model. In *Computational Aspects of Social Networks (CASoN), 2014 6th International Conference on*, pages 61–66. IEEE.

Ashraf, Q. M. and Habaebi, M. H. (2015). Autonomic schemes for threat mitigation in internet of things. *Journal of Network and Computer Applications*, 49:112–127.

Ashton, K. (2009). That 'internet of things' thing. *RFiD Journal*, 22(7):97–114.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805.

Babar, S., Mahalle, P., Stango, A., Prasad, N., and Prasad, R. (2010). Proposed security

model and threat taxonomy for the internet of things (iot). In *Recent Trends in Network Security and Applications*, pages 420–429. Springer.

Babar, S., Stango, A., Prasad, N., Sen, J., and Prasad, R. (2011). Proposed embedded security framework for internet of things (iot). In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pages 1–5. IEEE.

Baccelli, E., Hahm, O., GÃ¼nes, M., Wählisch, M., and C. Schmidt, T. (2013). OS for the IoT - Goals, Challenges, and Solutions. In *Workshop Interdisciplinaire sur la Sécurité Globale (WISG2013)*, Troyes, France.

Bandyopadhyay, D. and Sen, J. (2011). Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69.

Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2012). Nist special publication 800-57. *NIST Special Publication*, 800:57.

Bloomberg (2015). Explosion at northern chinese port kills dozens, hundreds hurt, [online]. access date: 27/08/2015. http://www.bloomberg.com/news/articles/2015-08-12/explosion-in-northern-china-shatters-windows-causes-injuries.

Capgemini (2014). Securing the internet of things opportunity: Putting cybersecurity at the heart of the iot, [online]. access date: 11/03/2015. https://www.capgemini.com/resources/securing-the-internet-of-things-opportunity-putting-cybersecurity-at-the-heart-of-the-iot.

Carter, F. (2014). A small perl program to control the bauhn/aldi wi-fi power points, [online]. access date: 2/05/2015. https://github.com/franc-carter/bauhn-wifi/blob/master/bauhn.pl.

Costa, S. (2011). xrtml's blog, [online]. access date: 7/06/2015. http://www.realtime.co/cache$_b$in/XPQlr0QXX291A7SyScZMb1ZKU.jpg.

Deng, J., Han, R., and Mishra, S. (2005). Countermeasures against traffic analysis attacks in wireless sensor networks. In *Security and Privacy for Emerging Areas in Communications*

Networks, 2005. SecureComm 2005. First International Conference on, pages 113–126. IEEE.

Dong, W., Chen, C., Liu, X., and Bu, J. (2010). Providing os support for wireless sensor networks: challenges and approaches. *Communications Surveys & Tutorials, IEEE*, 12(4):519–530.

ESPlorer (2014). Esplorer â integrated development environment (ide) for esp8266 developers, [online]. access date: 3/04/2015. http://esp8266.ru/esplorer/.

Farooq, M., Waseem, M., Khairi, A., and Mazhar, S. (2015). A critical analysis on the security concerns of internet of things (iot). *Perception*, 111(7).

Gartner (2014a). Gartner says 4.9 billion connected "things" will be in use in 2015, [online]. access date: 13/07/2015. http://www.gartner.com/newsroom/id/2905717.

Gartner (2014b). Gartner says the internet of things will transform the data center, [online]. access date: 13/07/2015. http://www.gartner.com/newsroom/id/2684616.

Gartner (2014c). Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business, [online]. access date: 13/07/2015. http://www.gartner.com/newsroom/id/2819918.

Gaur, P. and Tahiliani, M. P. (2015). Operating systems for iot devices: A critical survey. In *Region 10 Symposium (TENSYMP), 2015 IEEE*, pages 33–36. IEEE.

George, J. (2013). In the fukushima fallout, meet the hackers building a sensor network for global radiation, [online]. access date: 27/08/2015. http://motherboard.vice.com/blog/in-the-fukushima-fallout-meet-the-hackers-mapping-the-worlds-radiation.

Gomez, C., Oller, J., and Paradells, J. (2012). Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753.

Gomez, C. and Paradells, J. (2010). Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6):92–101.

Greenberg, A. (2015). Hackers remotely kill a jeep on the highway âwith me in it, [on-

line]. access date: 29/07/2015. http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S. L., Kumar, S. S., and Wehrle, K. (2011). Security challenges in the ip-based internet of things. *Wireless Personal Communications*, 61(3):527–542.

HP (2014a). Hp study reveals 70 percent of internet of things devices vulnerable to attack, [online]. access date: 20/03/2015. http://www8.hp.com/us/en/hp-news/press-release.html?id=1744676.VZuLOEVU2vk.

HP (2014b). Internet of things research study, [online]. access date: 10/03/2015. http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf.

Hui, J. and Thubert, P. (2011). Compression format for ipv6 datagrams over ieee 802.15.4-based networks, [online]. access date: 18/04/2015. https://tools.ietf.org/html/rfc6282.

Hyuk Park, J., Gritzalis, S., Hsu, C.-H., Roman, R., and Lopez, J. (2009). Integrating wireless sensor networks and the internet: a security analysis. *Internet Research*, 19(2):246–259.

IOActive, I. (2014). Ioactive lights up vulnerabilities for over half a million belkin wemo users, [online]. access date: 12/06/2015. http://www.ioactive.com/news-events/IOActive$_a$dvisory$_b$elkinwemo$_2$014.html.

Islam, K., Shen, W., and Wang, X. (2012). Security and privacy considerations for wireless sensor networks in smart home environments. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, pages 626–633. IEEE.

ITU-T (2012). Overview of the internet of things. Recomendation ITU-T Y.2060, International Communication Union. [online]. access date: 13/07/2015. https://www.itu.int/rec/T-REC-Y.2060-201206-I.

Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., and Qiu, D. (2014). Security of the internet of things: perspectives and challenges. *Wireless Networks*, 20(8):2481–2501.

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., and Alonso-Zarate, J. (2015). A

survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, 3(1):11–17.

Karimi, K. and Atkinson, G. (2013). What the internet of things (iot) needs to become a reality, [online]. access date: 10/03/2015. *White Paper, FreeScale and ARM.* http://www.freescale.com/files/32bit/doc/white$_p$*aper*/*INTOTHNGSWP.pdf*.

Keranen, A., Ersue, M., and Bormann, C. (2014). Terminology for constrained-node networks. Technical report, Internet Engineering Task Force (IETF). [online]. access date: 16/04/2015. https://tools.ietf.org/html/rfc7228.

Koh, J. Y., Ming, J. T. C., and Niyato, D. (2013). Rate limiting client puzzle schemes for denial-of-service mitigation. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 1848–1853. IEEE.

Kovac, D., Hosek, J., Masek, P., and Stusek, M. (2014). Keeping eyes on your home: Open-source network monitoring center for mobile devices. 37th International Conference on Telecommunications and Signal Processing (TSP), At Germany, Berlin, June 2014. http://www.researchgate.net/publication/266202005.

Krontiris, I., Dimitriou, T., Giannetsos, T., and Mpasoukos, M. (2008). Intrusion detection of sinkhole attacks in wireless sensor networks. In *Algorithmic Aspects of Wireless Sensor Networks*, pages 150–161. Springer.

Lauter, K. (2004). The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless communications*, 11(1):62–67.

Lee, C., Zappaterra, L., Choi, K., and Choi, H.-A. (2014). Securing smart home: Technologies, security challenges, and security requirements. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 67–72. IEEE.

Liu, J., Xiao, Y., and Chen, C. P. (2012). Authentication and access control in the internet of things. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 588–592.

Lopez, J., Roman, R., and Alcaraz, C. (2009). Analysis of security threats, requirements,

technologies and standards in wireless sensor networks. In *Foundations of Security Analysis and Design V*, pages 289–338. Springer.

Mahmood, A., Javaid, N., and Razzaq, S. (2015). A review of wireless communications for smart grid. *Renewable and Sustainable Energy Reviews*, 41:248–260.

Mayer, C. P. (2009). Security and privacy challenges in the internet of things. *Electronic Communications of the EASST*, 17:12. http://journal.ub.tu-berlin.de/eceasst/article/view/208/205.

McCurry, J. (2011). Iexplosion at japanese nuclear plant, [online]. access date: 27/08/2015. http://www.theguardian.com/world/2011/mar/14/japan-nuclear-explosion-second-reactor-fukushima.

Mendes, J. M. (2013). Security techniques for the internet of things. Universidade de Aveiro. RIA - Repositã^3*rioInstitucionaldaUniversidadedeAveiro.http* : *//hdl.handle.net*/10773/12686.

Minerva, R., Biru, A., and Rotondi, D. (2015). Towards a definition of the internet of things (iot), [online], access date: 15/08/2015. http://iot.ieee.org/images/files/pdf/IEEE$_{I}$*oT*$_{T}$*owards*$_{D}$*efinition*$_{I}$*nternet*$_{o}$*f*$_{T}$*hings*$_{R}$*evision*1$_2$7*MAY*15.*pdf*

Moessner, K., Le Gall, F., Cousin, P., et al. (2013). Internet of things strategic research and innovation agenda. *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*, 7:56–80.

Nguyen, K. T., Laurent, M., and Oualha, N. (2015). Survey on secure communication protocols for the internet of things. *Ad Hoc Networks*, 32:17 – 31. Internet of Things security and privacy: design methods and optimization.

Ning, H., Liu, H., and Yang, L. T. (2013). Cyberentity security in the internet of things. *Computer*, (4):46–53.

Noack, M. (2014). Optimization of two-way authentication protocol in internet of things. Master's thesis, University of Zurich, Department of Informatics (IFI).

NodeMCU (2014). Nodemcu connect things easy, [online]. access date: 3/04/2015. http://www.nodemcu.com/index$_e$*n.html*.

Notra, S., Siddiqi, M., Gharakheili, H. H., Sivaraman, V., and Boreli, R. (2014). An experimental study of security and privacy risks with emerging household appliances. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 79–84. IEEE.

OASIS (2014). Mqtt version 3.1.1 oasis standard 29 october 2014, [online]. access date: 22/07/2015. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

OpenHAB (2015). openhab enpowering the smart home , [online]. access date: 12/03/2015. http://www.openhab.org/.

Padmavathi, D. G., Shanmugapriya, M., et al. (2009). A survey of attacks, security mechanisms and challenges in wireless sensor networks. *(IJCSIS) International Journal of Comput er Science and Information Security.* http://arxiv.org/ftp/arxiv/papers/0909/0909.0576.pdf.

Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454.

Petersen, H., Baccelli, E., and Wählisch, M. (2014). Interoperable services on constrained devices in the internet of things. In *W3C Workshop on the Web of Things.* http://www.w3.org/2014/02/wot/papers/baccelli.pdf.

Postscapes. Internet of things hardware round-up, [online]. access date: 22/08/2015. http://postscapes.com/internet-of-things-hardware.

PrismTech (2013). Internet of things and industrial internet messaging technologies comparison,[online]. access date: 20/07/2015. http://www.prismtech.com/download-documents/1561.

ProofPoint (2014). Proofpoint uncovers internet of things (iot) cyberattack, [online]. access date: 12/06/2015. http://investors.proofpoint.com/releasedetail.cfm?ReleaseID=819799.

Raymond, D. R. and Midkiff, S. F. (2008). Denial-of-service in wireless sensor networks: Attacks and defenses. *Pervasive Computing, IEEE*, 7(1):74–81.

Reiter, G. (2014). Wireless connectivity for the internet of things [online]. access date: 12/06/2015. http://www.ti.com/lsds/ti/wireless$_c$*onnectivity/internet*$-$*of*$-$*things.page*.

Roman, R., Alcaraz, C., Lopez, J., and Sklavos, N. (2011). Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering*, 37(2):147–159.

Schneider, S. (2012). What's the difference between message centric and data centric middleware?, [online]. access date: 21/07/2015. http://electronicdesign.com/embedded/whats-difference-between-message-centric-and-data-centric-middleware.

Schneider, S. (2013). Understanding the protocols behind the internet of things, [online]. access date: 21/07/2015. http://electronicdesign.com/iot/understanding-protocols-behind-internet-things.

Sehgal, A., Perelman, V., Kuryla, S., and Schönwälder, J. (2012). Management of resource constrained devices in the internet of things. *Communications Magazine, IEEE*, 50(12):144–149.

Sen, J. (2010). A survey on wireless sensor network security. *arXiv preprint arXiv:1011.1529*. http://arxiv.org/ftp/arxiv/papers/1011/1011.1529.pdf.

Shelby, Z., Hartke, K., and Bormann, Carsten, O. A. d. . (2014). The constrained application protocol (coap). https://tools.ietf.org/html/rfc7252.

Sicari, S., Rizzardi, A., Grieco, L., and Coen-Porisini, A. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146 – 164. http://www.sciencedirect.com/science/article/pii/S1389128614003971.

Sigma, Designs, and Inc. (2014). Z-wave wireless communica-

tions for smart devices and iot, [online]. access date: 24/07/2015. http://www.embeddeddeveloper.com/documents/zwavewirelesscommunications.pdf.

Simoneau, P. (2011). The tcp/ip and osi models, [online]. access date: 16/08/2015. http://images.globalknowledge.com/wwwimages/whitepaperpdf/WP$_N W_T CP - IP.pdf$.

Singh, D., Tripathi, G., and Jara, A. J. (2014). A survey of internet-of-things: Future vision, architecture, challenges and services. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 287–292. IEEE.

Singh, V. P., Jain, S., and Singhai, J. (2010). Hello flood attack and its countermeasures in wireless sensor networks. *International journal of Computer Science issues*, 7(11):23–27.

Stallings, W. (2010). *Network Security Essentials: Applications and Standards*. Prentice Hall Press, Upper Saddle River, NJ, USA, 4th edition.

Stikonas, A. (2015). Reverse engineering orvibo s20 socket, [online]. access date: 21/04/2015. https://stikonas.eu/wordpress/2015/02/24/reverse-engineering-orvibo-s20-socket/?replytocom=988respond.

Suo, H., Wan, J., Zou, C., and Liu, J. (2012). Security in the internet of things: a review. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, volume 3, pages 648–651. IEEE.

Swan, M. (2012). Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0. *Journal of Sensor and Actuator Networks*, 1(3):217–253. http://www.mdpi.com/2224-2708/1/3/217/htm.

Toma, I., Simperl, E., and Hench, G. (2009). A joint roadmap for semantic technologies and the internet of things. In *Proceedings of the Third STI Roadmapping Workshop, Crete, Greece*, volume 1.

Townsend, M., Coleman-Lochner, L., and Rupp, L. (2014). Target is expected to pursue its first outside ceo, [online]. access date: 12/06/2015. http://www.bloomberg.com/news/articles/2014-05-06/target-is-expected-to-pursue-its-first-outside-ceo.

Tozlu, S., Senel, M., Mao, W., and Keshavarzian, A. (2012). Wi-fi enabled sensors for internet of things: A practical approach. *Communications Magazine, IEEE*, 50(6):134–143.

Vermesan, O. and Friess, P. (2014). *Internet of Things-From Research and Innovation to Market Deployment*. River Publishers.

Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., Jubert, I. S., Mazura, M., Harrison, M., Eisenhauer, M., et al. (2011). Internet of things strategic research roadmap. *O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, et al., Internet of Things: Global Technological and Societal Trends*, 1:9–52.

Vijayan, J. (2014). Target attack shows danger of remotely accessible hvac systems, [online]. access date: 12/06/2015. http://www.computerworld.com/article/2487452/cybercrime-hacking/target-attack-shows-danger-of-remotely-accessible-hvac-systems.html.

Wang, Y., Attebury, G., and Ramamurthy, B. (2006). A survey of security issues in wireless sensor networks. *IEEE Communications Surveys Tutorials â¢ 2nd Quarter 2006*.

Weber, R. H. (2010). Internet of things–new security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30.

Whitmore, A., Agarwal, A., and Da Xu, L. (2014). The internet of things—a survey of topics and trends. *Information Systems Frontiers*, 17(2):261–274.

Wireshark. Network protocol analyzer , access date: 2/04/2015. https://www.wireshark.org/.

Wu, M., Lu, T.-l., Ling, F.-Y., Sun, L., and Du, H.-Y. (2010). Research on the architecture of internet of things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pages V5–484. IEEE.

Xiaohui, X. (2013). Study on security problems and key technologies of the internet of things. In *Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on*, pages 407–410. IEEE.

Yang, Z., Peng, Y., Yue, Y., Wang, X., Yang, Y., and Liu, W. (2011). Study and application

on the architecture and key technologies for iot. In *Multimedia Technology (ICMT), 2011 International Conference on*, pages 747–751. IEEE.

Ye, N., Zhu, Y., Wang, R.-c., Malekian, R., and Min, L. (2014). An efficient authentication and access control scheme for perception layer of internet of things. *Int. J. Appl. Math. Inf. Sci*, 8:1617–1624.

Zhao, K. and Ge, L. (2013). A survey on the internet of things security. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on*, pages 663–667. IEEE.

Zhou, Y. and Feng, D. (2005). Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptology ePrint Archive*, 2005:388.