NTNU – Trondheim
Norwegian University of
Science and Technology

Security in Internet of Things Systems

Christian Dancke Tuen

Master of Science in Communication Technology
Submission date: June 2015
Supervisor: Frank Alexander Krämer, ITEM
Co-supervisor: Colin Boyd, ITEM

Norwegian University of Science and Technology
Department of Telematics

Title:	Security in Internet of Things Systems
Student:	Christian D. Tuen

Problem description:

This thesis will examine the security of protocols and techniques used in the Internet of Things. It will begin with an analysis of existing solutions and technologies on the market today. These analyses can be a purely theoretical one, based on published information, or a practical analysis of existing "things".

An overview of solutions to secure key concepts of security such as confidentiality, integrity, availability and privacy will then be discussed, and will focus on the challenges specific to the Internet of Things. This will also include an analysis of security requirements and some design recommendations to provide an overview of the current state of Internet of Things security, and possible solutions for any current security issues.

The thesis will provide a general conclusion on security concepts within the Internet of Things, how the security within this space is handled today, and possible solutions for improving security within this space for the future.

Responsible professor:	Frank A. Kraemer, ITEM
Supervisors:	Colin Boyd, ITEM & Frank A. Kraemer, ITEM

Abstract

The security in the existing Internet of Things has shown clear weaknesses, and is not near the security of existing computer systems. While the Internet of Things have grown rapidly over the last few years, the focus on security has not kept up. The Internet of Things introduces a plethora of new constraints and challenges that requires security to be focused on in another way than is usual in existing data systems. While today's systems use standards that are easy to implement and works for most forms of communication and storage, there is no such standard solution that will work on every device within the Internet of Things, because of the varied constraints between different devices, resulting in classifications within the Internet of Things.

This thesis lays the foundation on how security should be handled within the Internet of Things, both in present and future systems. Existing devices within different domains and with different technologies, have been analyzed to create a clear, tangible picture of the challenges and solutions that exists in the Internet of Things today.

The three main constraints in the Internet of Things, computation, bandwidth and energy, are described and used to create the foundation for the challenges that are presented. Different possible futures for the Internet of Things, and what challenges that will entail, are also described.

The solutions to the challenges will wary between what resources are available, so an in-depth presentation of possible solutions based on available resources are explained.

To increase the focus amongst developers on important questions regarding IoT security, complete guidelines are presented chronologically from beginning to end of design, development and maintenance of devices in the Internet of Things, targeted towards developers both with and without in-depth knowledge of information security. Possible solutions and alternatives are presented, as well as key questions that will make developers think about the consequences of the choices they make during the process.

Sammendrag

Sikkerheten til Tingenes Internett (Internet of Things) har vist store svakheter, og kan ikke måle seg med sikkerheten til eksisterende datasystemer. Mens Tingenes Internett har økt i omfang de senere årene, har ikke fokuset på sikkerthet klart å holde følge. Tingenes Intrenett fører med seg en rekke nye begrensninger og utfordringer som krever at man fokuserer på sikkerhet på en annen måte enn det som er vanlig i eksisterende datasystemer. Mens man i dagens datasystemer har standarder som er enkle å implementere og fungerer til de fleste former for kommunikasjon og lagring, finnes det ingen standard løsning som vil fungere på alle enheter i tingenes internet. Dette er mye grunnet de varierte begrensingene mellom forskjellige enheter, som fører til en klasseinndeling innen Tingenes Internett.

Denne oppgaven danner et godt grunnlag for håndtering av sikkerhet innen Tingenes Internett både for nåværende, og fremtidige systemer. Eksisterende enheter innen forskjellige domener og med forskjellig teknologi har blitt analysert for å danne et tydelig, håndfast bilde av utfordringene og løsningene som finnes i Tingenes Internett i dag.

De tre hoved-begrensningene som eksister innen Tingenes Internet, komputasjon, båndbredde og energi, blir beskrevet og brukt til å danne grunnlaget for de nevnte utfordringene. Forskjellige mulige fremtider for tingenes internett og hvilke utfordringer det fører med seg blir også beskrevet.

Løsningne på disse utfordringene vil variere utifra hvilke resurser som er tilgjengelig, så en inngående presentasjon av mulige løsninger basert på tilgjengelige resurser er videre forklart.

For å øke fokuset blandt utviklere rundt viktige spørsmål relatert til sikkerhet innen IoT, presenteres det fullstendige retningslinjer, kronologisk fra start til slutt av design, utvikling og vedlikehold av enheter for Tingenes Internett, rettet mot utviklere både med og uten inngående kunnskap om informsjonssikkerhet. Det er presentert mulige løsninger, alternativer og spørsmål som vil få utviklere til å tenke over konsekvensene av valgene de gjør i løpet av prosessen.

Contents

List of Figures

List of Tables

List of Acronyms

AES Advanced Encryption Standard.

AU Authenticated Encryption.

BTLE Bluetooth Low Energy.

CCM Cipher Block Chaining-Message Authentication Code.

CoAP Constrained Application Protocol.

DTLS Datagram Transport Layer Security.

ECC Elliptic Curve Cryptography.

ECDH Elliptic Curve Diffie–Hellman.

HCI Host Controller Interface.

HID Human Interface Device.

HTTP Hypertext Transfer Protocol.

IC Integrated Circuit.

IETF Internet Engineering Task Force.

IoT Internet of Things.

ISP Internet Service Provider.

LCD Liquid Crystal Display.

LED Light Emitting Diode.

LWM2M Lightweight M2M.

M2M Machine to Machine.

MAC Media Access Control.

MIC Message Integrity Check.

NFC Near Field Communication.

NIST National Institute of Standards and Technology.

NTNU Norwegian University of Science and Technology.

OMA-DM Open Mobile Alliance - Device Management.

OSI Open Systems Interconnection.

OTA Over The Air.

PCB Printed Circuit Board.

PKC Public Key Cryptography.

PKI Public Key Infrastructure.

PSK Pre-Shared Key.

QR Quick Response.

RFID Radio-frequency identification.

RSSI Received Signal Strength Indicator.

SD Secure Digital.

SOAP Simple Object Access Protocol.

SSID Service Set Identification.

SSL Secure Sockets Layer.

SyncML Synchronization Markup Language.

TLS Transport Layer Security.

TR-069 Technical Report 069.

VPN Virtual Private Network.

WPA Wi-Fi Protected Access.

WSN Wireless Sensor Network.

XML Extensible Markup Language.

XSS Cross Site Scripting.

Chapter 1
Introduction

The size of computer systems have decreased drastically over the years, from mainframes encompassing whole rooms, through desktop computers, and down to "smart" cellphones. At the turn of the century, a new concept emerged called the Internet of Things, envisioning all "things" in the world connected to a common Internet using tiny computing devices with communication technology. This would allow anything to speak to everything, making everyday life easier for everybody. With everything from lights, cars, washers, watches, kettles and stoves to chairs, helmets, weights, ropes, pacifiers, forks and even socks connected to the Internet of Things, we can safely say that we *really* are in the process of connecting everything to the Internet of Things, whether it actually makes life easier or not.

The largest use case for the Internet of Things today is gathering of data, and responding to the collected data in a useful way. While connecting all our things to the Internet will allow us to gain insight into our lives and environment, we can potentially allow others to gain the same insight if security is not handled correctly. With future envisionings of more things connected to the Internet of Things, the task of securing all these devices will become even more important in the coming years.

1.1 Objective

This thesis will asses the current state of the Internet of Things, look at the causes for the relatively low security standards, provide solutions to the problems troubling the Internet of Things, and work as a guideline for developers designing devices for the Internet of Things. Weaknesses in existing devices is discovered, real world problems encountered, and new ways of improving security presented.

1.2 Motivation

One of the motivations behind the thesis was a report presented by Hewlett Packard on the state of security in the Internet of Things where it was concluded that 80 % of the tested devices had insufficient authentication and or authorization, 80 % showed privacy concerns, and 70 % used unencrypted communication channels [HP14]. Looking at recent research and media reports quickly backs up these claims with examples such as the European standard for smart-grids using home-built and insecure cryptography [JN15], and BMW using no form of cryptography in any of their cars, affecting millions of people worldwide. With the number of connected devices having already reached several billions [Com15] and continuing to rise, IoT security will be affecting an increasing number of people in the coming years.

1.3 Major Findings

With the analyses conducted in this thesis, it is shown that the causes for not securing devices sufficiently in many cases is **oversight on the developers side more than purposefully ignoring security**. Many manufacturers developing devices for the home market will assume that their devices will only be used in a private setting, rely on the users network security, and that anyone connected to the same network should have the same rights to the device as the owner. This is especially the case with audio-visual systems, which often include no security on connections, firmware updates or privacy. This case was proven in a real world encounter at a popular venue, where the audio system of a well known manufacturer was connected to the public Wi-Fi, making it possible to control the audio in the venue, as well as manipulating the associated account for the streaming music service.

Good security mechanisms can be vulnerable through **flawed implementations**. An analysis is made of a popular WiFi connected storage device, that use off the shelf security mechanisms that are part of the 802.11 specification. While the developers of this device used relatively secure mechanisms in their product for authentication, authorization and encryption, their key derivation function showed some glaring faults after it was discovered that the keying material was the MAC address of the wireless network interface, and the function itself was a simple monoalphabetic substitution cipher. This was done to make the set-up procedure easier for the consumer, and ended up exposing their weak key derivation function through the associated Android application.

Introducing **outdated technology** to the Internet of Things is generally not a good idea. Analyzing a new product using old, cheap technology that has received numerous updates to compete with the newer technologies, such as Internet connectivity, showed a glaring lack of security. By eavesdropping the system's communication,

it was possible to take control of the whole system by acquiring a single packet of any type. A device is not more secure than its weakest point, and introducing Internet communication to outdated protocols and marketing them as the future of IoT does nothing but damage the future of security in IoT.

A lot can be shown from the data logged by IoT devices, but the level where this becomes a privacy issue is a matter of opinion. The question of user privacy is tackled throughout the thesis and presented in the analyses of devices, as well as in an analysis on data from a specific user, gathered from a smart-grid meter. The user's daily routines become quickly apparent, as well as when the user has been on holiday, takes a day off work, etc. We show why this is a challenge and possible solutions to the problem where it can be solved, but the overall problem with privacy is defining where exactly something becomes a privacy issue.

1.4 Defining Challenges and System Categories in IoT

Throughout the thesis, three distinct constraints are used to separate devices in IoT from traditional desktop computing, and lays the foundation for the presented challenges, namely the **power** requirement, **bandwidth** requirement and **processing** requirement. These constraints will always be present in IoT, so how one works around them, and to what degree the device is constrained by the different constraints, will define how a device and its security mechanisms are designed and implemented.

The amount of available power will strongly dictate the challenges and possible solutions available. Using the power-constraint as the main differentiator of IoT systems, the IoT space is separated into tree distinct categories: **Severely Constrained**, **Unconstrained**, and **Mixed Systems**. If a device is severely constrained by power, processing capabilities and availability will likely be low as it will not listen continuously to conserve power. One can forget LCD displays, as the common user interfaces in these devices are buttons, LEDs, microphones, or other sensors. Unconstrained devices will less often be constrained by bandwidth, but also less mobile, than any of the other categories. Mixed systems will usually include multiple types of wireless communication, and need more processing capabilities on the unconstrained nodes to process requests from the constrained devices. What approach is taken to secure a device will be decided from what category the product falls under.

1.5 Biggest Challenges Surfaced

Some challenges quickly appear as the most severe, and so fundamental that they will affect all other challenges in the space. One of the hardest challenges within IoT is affected by all three of these constraints, namely the initial paring process. **Key**

distribution on a device with little processing power in itself is a challenge, adding low availability and a lack of user interface makes it even harder.

In the most capable devices, the concept of Public Key Infrastructure (PKI) can be introduced to control authorization, but protocols like Secure Sockets Layer (SSL), Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) will often require too much of an IoT device, resulting in extensive use of Pre-Shared Key (PSK) in the least capable IoT devices today. Many protocols designed for less capable devices will use a direct pairing process, usually requiring user interaction to initiate the pairing. Some solutions require push of a button, and close proximity, which is often a bad solution if the nodes are mounted in out of reach places, while others require the user to input or simply accept a pairing key. Different out of band solutions are presented such as using external ports to connect to a computer or media device, using another wireless radio such as Radio-frequency identification (RFID)/Near Field Communication (NFC), using a photoresistor (light sensor), or simply audio through a microphone.

Post-production management is a hard, and potentially severe problem for IoT. When a product is shipped, and a security vulnerability is discovered, the need to update devices will be urgent. If not, one would have to recall several batches of devices, causing a huge economic loss. We present the problems related to post-production updates, showing the possible alternatives existing today, and the lack of security focus in the existing protocols. Several future solutions are presented depending on the capabilities and physical connections on the devices, both in-band, and out-of-band methods. Emphasis is put on exactly how devastating a flawed or altered update can be, and thus urging developers to make use of authentication and authorization to ensure that the update comes from the manufacturer, and that the update has not been tampered with by a third party.

1.6 Predicting the Future

As the world of IoT is in its early years, an important part of any analysis on the subject is to look into the future. Special emphasis has been put on the importance of a central server or cloud service, and future networking topologies and standards. Some opinions support a clear future of increased cloud processing for smart analysis (the Google-approach), while others envision a future of the Internet of Things where the central server will be of less importance, as data processing, storage, and actions will be processed device-to-device in the local mesh.

1.7 Outline

The main section of the thesis is separated into four parts: analysis, challenges, solutions and guideline.

The **analysis** chapter will evaluate current devices on the market through hands-on analysis of products, as well as analysis of research and media coverage. Four different devices, within four different categories of usage, using four different wireless technologies will be evaluated. These devices will form a basis on how security is handled in the IoT market today, as well as work as examples for challenges, solutions and the guideline.

The **challenges** chapter will highlight the different challenges that exists in IoT today, as well as future challenges. There are some key differences from the challenges that exist in regular desktop computing, which stems from the constraints that exists in IoT, that will affect the way security is handled in IoT devices. Different envisions of the future of IoT will be presented, and show what new challenges will occur in the future.

The **solutions** chapter will present different solutions to the challenges presented in the challenges chapter. What and how challenges are implemented will be dictated by the type of system and device is being designed. Different approaches to solving challenges will therefore be explained in this chapter.

The **guideline** chapter is presented as a guideline with best practices for a developer designing a device for the Internet of Things in relation to security. It will chronologically present different choices that needs to be made, and questions that needs to be considered through the design process, such as:

- How the available power differentiates the type of device and the possible security mechanisms.
- Why security mechanisms should be a part of the hardware design process, and what different solutions exists for devices in every category presented.
- The many different ways to distribute keys to a device is presented, and what should be considered when choosing a key distribution mechanism is discussed.
- What should be though of, and how one can control and update a device, once it has left production.
- The different considerations, questions and some possible solutions related to data storage.

- How to handle privacy-concerns, and questions and considerations that needs to be discussed before collecting private data.

Chapter 2
Background

This chapter will provide the necessary background material needed to understand the technical aspects of this Thesis, as well as explaining the background of the Internet of Things in general.

2.1 M2M

The early years of Internet of Things (IoT) started with Machine to Machine (M2M) communication. While M2M is still an ambiguous term, it is more specific than the IoT term. Machine to Machine communication indicates two machines communicating with each other, usually without human involvement. The communication platform is not defined, and can be both wireless and wired communication.

The term M2M stems from telephony systems. In these systems, different endpoints needed to exchange information between each other, such as the identity of the caller. This information was sent between the endpoints without a human being needed to initiate the transmission. The M2M term is still very much in use, especially in the industrial market, and is commonly regarded as a subset of IoT.

In later years, the M2M acronym has been given alternate meanings such as Machine to Mobile, Machine to Man, Mobile to Mobile, Mobile to Man etc. [JS10], but these are not in wide use.

2.2 IoT

The IoT term is a newer one, originating from a man named Kevin Ashton in the late 20th century. He is regarded as the first person to use the term Internet of Things in a presentation about RFID at Procter & Gamble in 1999 [Ash09]. The IoT acronym is used to define the notion of things connected to the Internet. These things can vary wildly with everything from heart rate monitors to wastewater meters being included in the term.

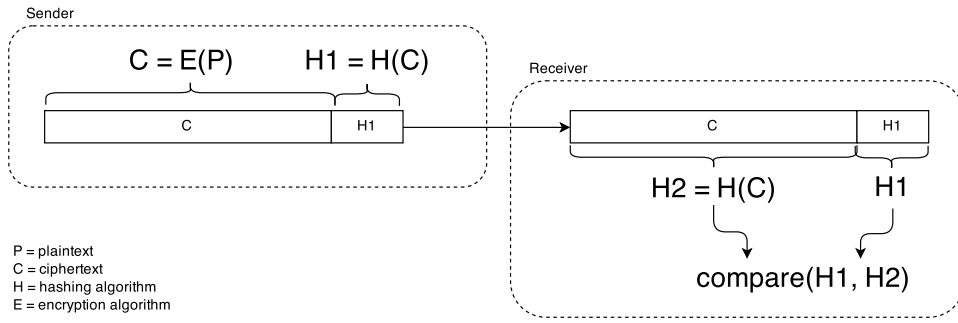

Figure 2.1: Illustration of Encrypt-then-MAC. 1. Sender encrypts plaintext, 2. Sender generates MAC from ciphertext, 3. Sender sends both ciphertext and MAC to receiver, 4. Receiver generates MAC from ciphertext, 5. Receiver compares generated MAC with received MAC

2.3 Authenticated Encryption

Authenticated Encryption (AU) is a method of encrypting and decrypting data while also providing integrity and authenticity validation. This method will include Message Authentication Code (MAC), which is a generated identifier that is unique for the data it has been generated from. One of the possible ways of generating such an identifier is by the use of a keyed hash function. There are three different approaches to authenticated encryption:

- Encrypt then MAC
- Encrypt and MAC
- MAC then encrypt

Mac-then-encrypt is the method used by the popular SSL/TLS protocol, used for example in secure web-browsing (HTTPS). An example of Encrypt then MAC is shown in Fig. 2.1.

2.4 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a cryptographic concept based on elliptic curves. By using point multiplication on points on a curve and utilizing the elliptic curve discrete logarithm problem, one can achieve the same cryptographic strength as algorithms based on the prime factorization problem, such as RSA, using shorter key lengths. Different properties such as resistance to side channel attacks will differ

between the curves used. One of the most common is Weierstrass, while other ones such as Jacobian curves and Edwards curves will improve different properties of ECC. When using hardware-based ECC one is usually restricted to the National Institute of Standards and Technology (NIST)-recommended curves, which has seen some controversy in recent years[DJB13]. There is an ongoing debate about the ownership of and patent rights to ECC which might have slowed the adoption of ECC somewhat[Lab07]. While ECC has many potential use-cases, it has seen the most wide adoption within Public Key Cryptography (PKC).

2.5 Public Key Infrastructure

PKI is a definition of the infrastructure and mechanisms needed to provide secure communication on an insecure network using public key cryptography. It consists of several different parts [Vac04]:

- A Certificate Authority
- A Registration Authority
- Directories
- Certificate Management

The basic relationship within the PKI is as follows: A company applies for a certificate through a RA that will confirm or deny the identity of the requester. If the identity is accepted, the RA will request CA to issue a certificate to the company, and also store the certificate in a certificate directory with its public key. This certificate can in turn be used to validate the identity of the company to any connecting customers or devices using the CA. If needed, the CA can revoke or renew the certificate.

2.6 SSL/TLS

SSL was originally created by Netscape Communications in 1994 (released in 1995) to provide a secure way of browsing the Internet using a web browser [AFK11]. Communication is encrypted on the application level between the server and the client using PKI. SSL has since been released in version 2 and 3, before changing the name to TLS when Internet Engineering Task Force (IETF) took ownership of the protocol [Die14]. SSL version 2 and 3 are both considered insecure, and are being phased out, leaving the SSL name all together.

TLS was first released by IETF in 1999 (v1.0) [DA99], with new version releases in 2006 (v1.1) and 2008 (v1.2). TLS includes a handshaking procedure for the client

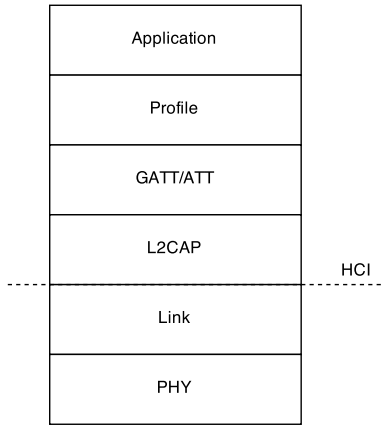

Figure 2.2: The layered architecture of Bluetooth Low Energy, showing the HCI level

and server to be able to negotiate cipher suites and exchange certificates to verify the sender/receiver.

2.7 802.11 WiFi security

The security of the 802.11 specifications has been the target of many attacks over the years. The first wireless security used in WiFi was the Wired Equivalent Privacy (WEP) standard, that secured the network using the RC4 stream cipher. The poor implementation, using short Initial Vector (IV), made it very easy to break the wireless security, access the wireless network and listen to the traffic transferred over it.

The second implementation was stronger, but still had its flaws. The next security standard was called Wi-Fi Protected Access (WPA). The new standard adopted the Temporal Key Integrity Protocol (TKIP), employing a per-packet key and thereby prevents the known attacks of WEP.

The third security standard called WPA2 is the newest one in use today. It swaps RC4 for AES and Cipher Block Chaining Message Authentication Code Protocol (CCMP). While attacks are possible against the WPA2 standard [KTT+12] by eavesdropping the pairing routine, choosing sufficiently long passwords will make rainbow-table attacks costly.

2.8 Bluetooth Low Energy / Smart

Bluetooth Low Energy (BTLE) is a protocol released by the Bluetooth Special Interest group. The technology specifies the full stack for transferring data wirelessly between two devices. The technology is using the publicly available 2.4 GHz frequency band to communicate. Unlike regular Bluetooth which use the concept of Piconets, where one device will act as a master for each piconet and all slave devices need to wait and listen in standby, Bluetooth LE uses a single connection between the master and slave. All slaves will contiually broadcast announcement frames, and the master will only be able to set up a connection on the back of such an advertisement frame. Thus, the slave will be able to control independently when to stand by, and when to sleep. Using this communication method allows BTLE to consume significantly less power than classic Bluetooth [Com13].

Bluetooth LE supports two mutually exclusive security modes, “LE Security Mode 1” and “LE Security Mode 2”, which will secure the Link Layer and the attribute protocol layer (ATT) respectively, shown in Fig. 2.2. These security modes will use Cipher Block Chaining-Message Authentication Code (CCM), 128 bit Advanced Encryption Standard (AES) encryption and Message Integrity Check (MIC) [GOP12]. Payload on the application layer can of course also be encrypted on top of the Link Layer or ATT layer security.

One of the most vulnerable parts of a Bluetooth LE pairing is exchange of the Temporial Key. This key can be distributed out of band, by manually entering a key, or as is usually the case in IoT, using the “Just Works” mechanism. In earlier versions of the Low Energy specification (4.0, 4.1), a Man in The Middle attack could easily be utilized to acquire the TK. The new “Secure Simple Pairing model” included in v.4.2 release of the specification use Elliptic Curve Diffie–Hellman (ECDH) PKC in the pairing process, and thus increases the security of Bluetooth LE drastically [Gro14b]. Which version of the specification is implemented will thus have a large impact on security.

Chapter 3
Hands-On and Literary Analysis of Existing Products

To examine the existing solutions implemented in the IoT market, an analysis of systems currently on the market will be performed in this chapter. The analyses will also serve as examples of security solutions, and how to improve security in this field. The analyses will be based around different use cases and different technologies affecting a consumer's everyday life. We will explain how the results were acquired, and shortly mention the effects of the issues, with a more in-depth discussion in Chapt. 4. The hardware and software used in the analyses are listed in Sect. A.1 & A.2, including descriptions and links to the devices in Sect. A.3.

Domain	**Product**	**Identified Issues**	**Result**
Health	Fitbit One health tracker	Unique ID is broadcasted all the time, and never changed.	Users can be uniquely tracked at all times.
Automobiles	BMW Connected Drive	The only authorization implemented in the in-dash device is source IP address. Information is sent unencrypted.	Doors can be opened, horn set off, lights turned on, AC controlled and GPS position read.
Photography	Eye-Fi Mobi	The key derivation function is a monoalphabetic substitution cipher using the MAC-address as keying material.	Adversary can wirelessly connect to SD cards and download all photos stored on the card.
Home Automation	HomeEasy	The only authorization is source ID, closely resembling the solution implemented by BMW. Information is sent unencrypted.	Adversary can controll all devices, or render the system useless.

Table 3.1: Table of analyzed domains, the product within this domain, the identified issues, and the results of the identified issues.

3.1 Analysis of the Fitbit Health Tracker

Fitbit is an American company producing a range of human activity monitors with the same name, that was first released in 2007, and has since expanded to many different versions which now include health-monitoring, heart-rate monitoring, and phone notifications. This exploratory section will focus on the Fitbit One, released in September 2012, which includes metrics such as step count, floor count, active minutes, distance traveled, calories burned, sleep quality, amongst others. The device can be synchronized with the Fitbit web service for historic storage and analysis of data. Synchronization is conducted using Bluetooth 4.0 either through a compatible smart-phone, or an included USB dongle. The two methods of synchronization have some important differences. While a cellphone will synchronize the Fitbit of the authenticated user, the dongle will synchronize all Fitbits within range, and does not require any form of authentication. Synchronization through the dongle will be the main focus of this section, as it is the most interesting of the two.

3.1.1 Security

All Fitbits connecting to the dongle will use Bluetooth 4.0 for communication between the device and the machine. The host machine will then read the data from the Bluetooth dongle, and then transmit this data to the Fitbit web service using TLS so that the data cannot be read in transit.

Using the Charles Proxy tool [vR15], we are able to inspect the data that is sent from the Fitbit software to the web service, and thus the data that is transmitted from the device to the computer, as the only local storage on the computer is a log file containing information about the Fitbit devices. Detailed historical data for up to two weeks and summary data up to 30 days are stored locally on the Fitbit device until synchronized with the web service.

When inspecting the request data payload, we find base64 encoded data that is encrypted directly on the device, thereby making it hard to read anything reasonable from the request. As the data is encrypted on the device, it needs to have a key stored that the sever knows, and that is associated with this specific device. The specific encryption method used on the device is not known, but data dumps of the USB dongle has clear references to AES [vR14].

When using the Fitbit smartphone application, we are able to set up a connection to a Fitbit device that is associated with our account. When the phone and device has an active Bluetooth connection, we are able to see real-time step count from the device. When looking at the data communicated between the phone and the device, current step-count data can be viewed without more encryption than that provided

Figure 3.1: The Fitbit One activity tracker

JgIAAAEAmkkAAPcasSsoBffScgq4enqr0vLN7VSLhf5Xu2TGOmi
MM00TNGyytUXpijiRpGWT6gcGqukIBfbJb9ZqcljeWSg91Tk/1Yb
ZsplPv+UWCaNcjgEj+37ep2Q4c/ZFKqjC/iOvTwTzrLOhrSLcAnl
Au6h12g+ihCrTR8HUsfs1xEjzEIdjLqDmpihW93zKvtze8/L0KZH
v2ckVucuFCpd1H32oPtLwxK2hGBoeprH/9QPh3loDiMaQTn4CxrU
WzZUV9D+ve53EHWZQ9Zved5w43r3JEhArerce5LHti1Me7bcd8Om
5PmM5m0aVX4lmbG88L3Dg8k+lEcM4r/VtITYSe+CC9KgV4ofBbfE
Pw34KUar8zJZZxlT4uLiWHitLpkkXhcTjmGREvXuuNA8BAA==

Figure 3.2: Example encrypted request payload data in base64 format

by the Bluetooth connection. But this is also the only data that is transmitted "unencrypted", and only when there is an active connection to the device.

3.1.2 Key exchange

The AES encryption inside the Fitbit will need an encryption key that is known both by the Fitbit and the server, so that the data can be decrypted and viewed in the web application. This key can be added to the device in two ways, either during production by referencing the key to the device id, or it can be distributed to the device after it has shipped to the consumer.

The setup procedure of a new Fitbit device starts with installation of the Fitbit synchronization software and insertion of the accompanied USB dongle on a compatible computer. The software will then ask the user to position the device as close to the dongle as possible, and not to have any other devices in near proximity. If the

dongle detects other devices nearby (using the Received Signal Strength Indicator (RSSI) value), the user will be asked to move these devices away from the dongle. Passive Bluetooth sniffers will of course not be detected by using this method, and thus this method works more like an identifier than an actual security feature. The user will then be asked to input their credentials to associate the device with the account.

While we cannot with certainty prove which method is in use with the Fitbit, both methods are viable and have their separate features and drawbacks.

- If the encryption key is stored on the device during production, one has to store the keys safely as the keys in the devices cannot be updated, and all existing devices are useless if this database was to get lost or corrupted. An upside to this method is that the key will never be sent over any wireless transmission protocols outside the factory, thereby shielding the key from eavesdropping.
- If the encryption key is sent to the device after it has reached the consumer, the production procedure can be simplified as there is no need to store a key in the database during production, and the key can be updated in the case of a database breach or loss. A clear drawback of distributing the key after it has reached the consumer, is the possibility of someone eavesdropping the transmission containing the key. Even though this would involve either breaking the TLS/SSL transmission, or the Bluetooth connection between the computer and device, it is still theoretically possible.

As the Bluetooth connection is a possible point of attack, the connection-procedure of the devices are obscured from the published standard [BC14]. While this makes it harder to use publicly released tools to crack the Bluetooth connection, security by obscurity is not regarded as an effective security measure.

The dongle communicates with the host computer using serial commands, and advertising itself as a Human Interface Device (HID), this information is clearly readable, and could expose the key during setup if the key is distributed this way. But attacking the USB serial communication is not regarded as the point of least resistance in the communication channel between the Fitbit servers and the device.

3.1.3 Privacy

As mentioned in Sect. 3.1.1, the Fitbit activity and health-monitors use the Bluetooth 4.0 and Low Energy version of the Bluetooth specification. A requirement, and inherit functionality of BTLE, is transmission of announcement frames every X seconds to announce the presence of the device. This announcement frame contains a uniquely

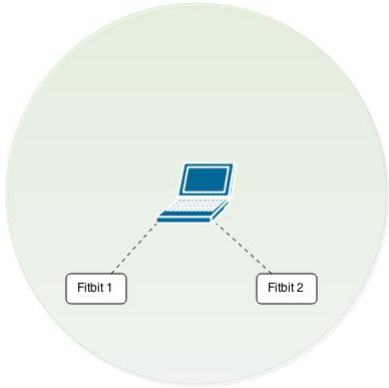

Figure 3.3: The Fitbit key distribution device discovery. The setup-procedure would not be allowed to continue in this situation

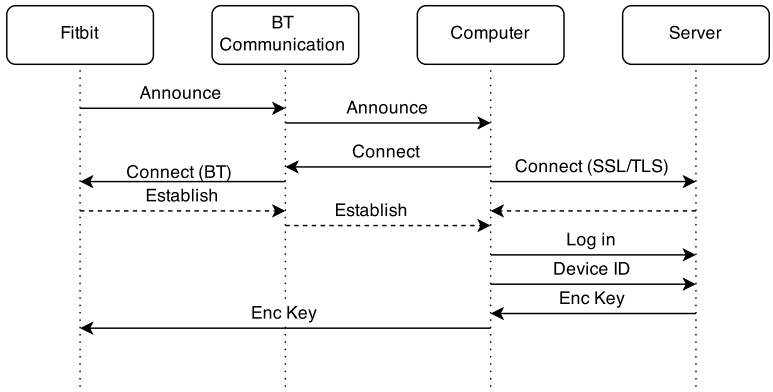

Figure 3.4: The Fitbit key distribution

identifiable address that does not change. There have been reports claiming that the trackers will change the address at certain intervals, but this has not ever been observed with the "Fitbit One" tracker used in this thesis, or in other studies [BC14].

By using a cheap, off the shelf USB Bluetooth 4.0 dongle, we are able to listen for BTLE announcement frames in the near vicinity of the dongle, and we are thus able to track all devices within the range of the Bluetooth radio. The announcement frame also includes the RSSI, meaning we also have some sort of concept of the distance between the computer and the device. While this value varies too much to precisely locate a persons distance from the sensor, we can assume whether the person is in the same room as the sensor or not. With inclusion of more sensor, we would be able to more closely position the BTLE device. The result of a few days

tracking can be seen in appendix B, showing anonymized addresses, RSSI values, and the timestamp for first encounter with the address.

3.1.4 Security Implications

The Fitbit range of trackers are secured with good on-chip encryption between the device and the servers, as well as using strong encryption during transmission of the encrypted data between the computer and server.

The privacy of a user wearing the Fitbit all day is poor. By using a network of cheap receivers, the position of a user can be tracked and pinpointed down to a small area. While the privacy implications are not as severe as leaking GPS data, the privacy concerns can be as relevant when in close proximity to the user.

Implementing randomized Bluetooth-addresses will solve the security-issues in this particular case.

3.2 Literary Analysis of the BMW Connected Drive System for Automobiles

Connected cars are becoming more and more common, and they are incorporating more and more features over the network. A German motoring association called Allgemeiner Deutscher Automobil-Club (ADAC) found a vulnerability affecting 2.2 million cars worldwide from BMW, Mini and Rolls Royce using the BMW Connected Drive system [1]. The door-locks, headlights and horn could be controlled remotely, the current state of the car sensors such as GPS location, current speed and door lock state could be viewed remotely, the emergency numbers could be changed remotely, and all private network communication could be eavesdropped remotely. The on-board computer checked that the data source was one of the BMW servers, but as IP-spoofing is relatively easy, this is not considered adequate security.

3.2.1 Security

The security in this case has been virtually non-existent, and has been reliant on no one eavesdropping the mobile network. Authentication of transmitted data have also been virtually non-existent with only simple IP origin check for authentication. The car's sensor data could be read and data could be written to the car over the mobile network in clear text. A replay-attack allowed the doors of the car to be opened and headlights blinked amongst other things.

The system also allowed for modification of emergency-numbers that would automatically be phoned after an accident. The only requirement for this modification

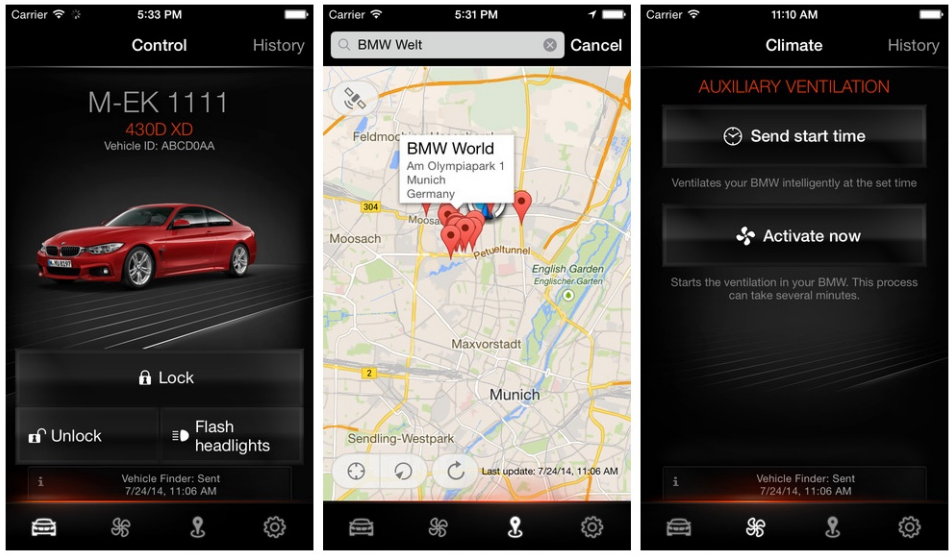

Figure 3.5: BMW Connected Drive Application showing the features that were accessible by anyone. Credit: Bayerische Motoren Werke AG

request was that it came from the IP of a BMW server. Using IP spoofing, a falsified number could be inserted into the cars.

3.2.2 Privacy

As the car's current GPS location was transmitted unencrypted over the network, the privacy implications could be huge. Even though the cars position is not clearly user specific, it will in many cases give a relative location such as the user's home-address or work location.

3.2.3 Security Implications

The GPS location of all the cars could be read, having huge implications on the privacy of the user. The doors could be opened remotely, so that items stored in the car could be retrieved without sounding the alarm of the car. The emergency numbers could be altered so that the owned could think that the emergency services had been contacted, when in fact they were not.

This issue was resolved by BMW by adding SSL/TLS through a wireless system update.

Figure 3.6: The EyeFi Mobi SD card

3.3 Analysis of the Eye-Fi Internet-Connected SD-Card for Digital Cameras

EyeFi is a Secure Digital (SD) card that is specifically targeted towards photography. The SD card contains a wireless access-point and web-server in addition to the normal flash storage. These SD cards come in two different versions, one targeted towards professionals where the card connects to a wireless network, and the other targeted towards normal consumers that acts as a base-station for phones and laptops to connect to. Throughout this analysis, the SD card targeted towards normal consumers, called EyeFi Mobi, will be the target of analysis.

The card comes with an accompanying app for mobile devices, to be able to download pictures straight from a camera to the mobile device wirelessly. This is made possible by the Wi-Fi-module in the SD card, acting as a Wi-Fi Protected Access (WPA) or WPA2 base station depending on the card's revision version. To connect to the card, the user needs to provide an access code in the app which is written on the SD card's packaging. This access code is in reality the key for the wireless network. The typical Service Set Identification (SSID) of these card will be in the following form:

SSID: Eye-Fi Card 606643

Where the 6 last digits in the SSID are the last 6 digits of the Media Access Control (MAC) address, where the first 6 digits of the MAC address are vendor specific (001856 = EyeFi), and does not change between devices.

3.3.1 Security

The first note one can make is that these cards have used WPA encryption for many years which is considered a weaker form of wireless encryption compared to WPA2. A dictionary attack can easily be used against this kind of network, with the success

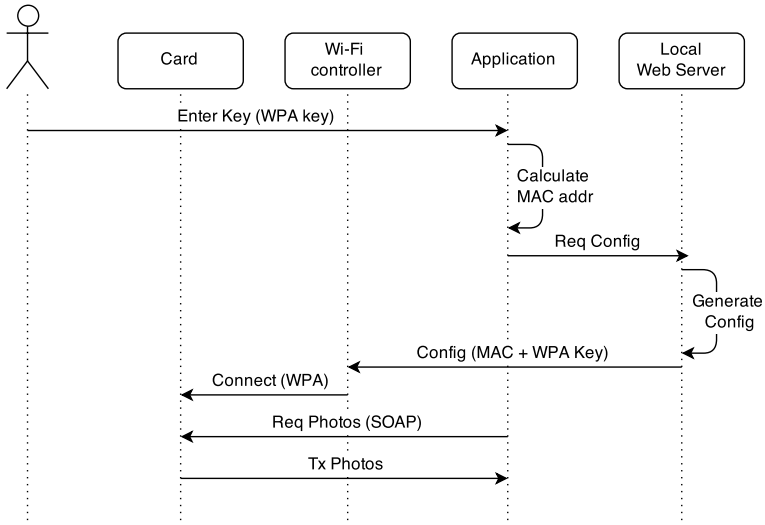

Figure 3.7: The EyeFi key setup procedure

depending on the complexity of the key. A typical network key for the Mobi SD card is in the following form:

A52QMKSHVZ

We see that the key consists of ten upper-case letters and numbers, in no particular order. The number of possible combinations for this type of key is $(26+10)^{10} = 36^{10}$, which makes it hard to brute-force. In reality the keys exclude the letters I and O, and the numbers 0 and 1, reducing the number of possible keys to $(24+8)^{10} = 32^{10}$, but still a too large number of keys to make brute-force practical.

We then examine the mobile application that is used to connect to the network and transfer images to the mobile device. Upon the first startup of the application, the user is asked to provide the key that is printed on a label on the container the SD card is shipped in. The mobile application will then automatically generate a provisioning profile which configures the Wi-Fi settings of the mobile device to connect to the users specific card. The mobile device will automatically open the browser and download this provisioning file from a URL in the following format:

http://localhost:59278/provisioning/wifi.mobileconfig?Mac=00-18-56-60-66-43

We see that the configuration is requested from the localhost address, meaning that the EyeFi Mobi application is running a web-server in the background, listening on port 59278. It is also easily seen that the URL contains the MAC address of the device the user is trying to connect to, meaning that the application already knows the MAC address of the Mobi SD card after only being provided a Wi-Fi key. Doing the same procedure again in flight-mode yields the same results, showing that the application is not requesting a central database to get information about the SD card. This also proves that there is a relationship between the key and the MAC address of the device.

There are two different solutions on how the device can know the MAC address from the network key:

- The mobile application can store a local database containing a mapping between the keys and the MAC address of the wireless card. This would result in either a bloated application containing all possible MAC addresses that would be used now and for the future, or an application that needed to be updated for all users each time a new shipment was ready.

- The mobile application contains a function to calculate the MAC-address from the network key. This would also mean that the key is generated from the MAC address during production, and that this is thus not a one-way function. This would be the more practical of the two solutions, but would mean that the security of the network is reliant on the function being kept a secret, or security by obscurity.

To find out what solution was used, a version of the mobile application written for the Android operating system in the Java programming language was acquired from Google's application market as an Android application package (apk) file. Using this package, we are able to unpackage it to a compiled Dalvik Executable (dex) file, and from there translate it to a Java Archive (jar) file. Using automated software to decompile the application, we are able to read most of the original Java source code.

After examining the application's code, the function that translates between the network key and the MAC address is visible in a class called MobiDecoderRing, showing that the key derivation function uses the MAC address that is easily visible in the broadcasted SSID as keying material. The key derivation method used in this product was then analyzed, and proved to be a simple monoalphabetic substitution cipher (hence the "Decoder Ring" class name), although with some added entropy. Simple attacks will be able to crack this form of "encryption" easily.

3.3.2 Privacy

Wi-Fi hosts will have the same problem as Bluetooth does, where the device is broadcasting a unique MAC address that will be fixed and traceable. While the transmission power is likely on par with the Bluetooth module, these cards will only be broadcasting its network when they receive power, i.e. when the camera is powered. When the camera is powered off and stored in a pocket or backpack, no traceable information can be picked up, and thus the privacy implications with these devices are minuscule.

3.3.3 Security Implications

As the security of these cards are reliant on a monoalphabetic substitution-cipher key generation algorithm, using the MAC address as keying material, what started out as a relatively secure system is now insecure. An adversary could get access to all the images stored on a camera remotely, potentially from a far distance using a high-gain antenna.

3.4 Analysis of the HomeEasy Protocol for Home Automation

The HomeEasy automation protocol is used by many device manufacturers such as Nexa, Byron, Proove and Anslut. The system relies on 433.92MHz radios to transmit information between them. The system is used for controlling power plug relays, dimmers, door bells, windows blinds and motion sensors. The system use On-Off Keying, an Amplitude Shift Keying technique, to transmit codes to the receivers [Wes12]. The devices are connected to the Internet through a central hub that upon request controls the 433MHz devices, or without Internet through local 433MHz transmitters (light switches) directly.

3.4.1 Security

The HomeEasy protocol works by assigning receivers to transmitters. This is done by pairing each individual receiver to the transmitter by pressing keys of both devices, and exchanging the ID from the transmitter to the receiver wirelessly. This ID is a 24 bit integer, distinct to each transmitter. When a receiver receives a command from one of the stored IDs, and its own unit code, it will act upon the command. If not, the command will be discarded.

The protocol also allows for one group command to be transmitted by each transmitter. Every receiver that is included in the group command will act upon the group command request from the stored transmitter ID. The packet format consists of 32 bits in total (in practice 64 as all bits are sent redundantly). The first 26 bits

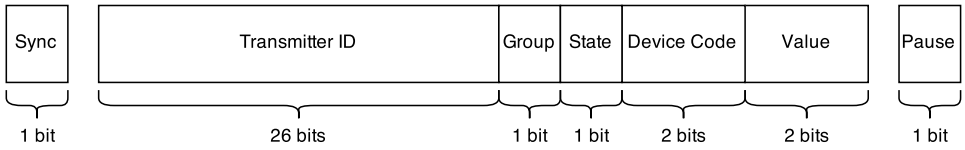

Figure 3.8: The HomeEasy packet format

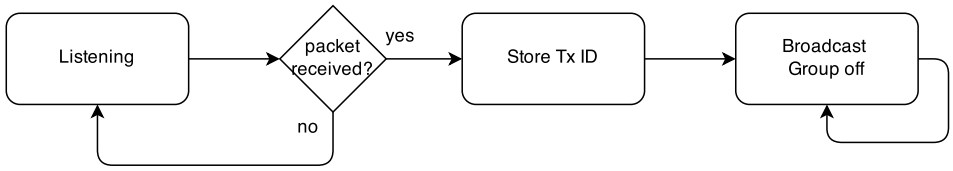

Figure 3.9: The HomeEasy protocol security exploit state diagram

are the transmitter ID. This is the identification of the transmitter that needs to be unique for every transmitter for different users not to interfere with each other. The next bit is the group flag, indicating whether this is a group command or not. Then there is the state, indicating whether the device should be switched on or off. The next two bits is the device code, indicating the receiver, so that individual receivers can be controlled. The last two bits indicate the value of the action. This can be a dimming value or the heights of window blinds. The packet structure is shown in Fig. 3.8.

Since there is only one group command, most receivers will be included in this command. This means that transmitting a group command, using the specified ID will likely trigger most of a users devices. To test what impact this has on the HomeEasy protocol, a microcontroller was used to evaluate this in practice.

The device is made from an Arduino microcontroller with both a 433MHz transmitter and a 433MHz receiver. Its operation is in two modes, listening and broadcasting. In listening mode, it will constantly listen to the receiver, decoding all packets that arrive in the correct format. If a correct packet is received, the microcontroller will store the ID of the transmitter. As every command will include the transmitters ID, any button pressed on a transmitter (light switch) will be sufficient to acquire the transmitter ID. This process is illustrated in Fig. 3.9, and the Software Defined Radio (SDR)-traces of the experiment is shown in Fig. 3.10.

After acquiring the transmitter ID, the device will impersonate the transmitter,

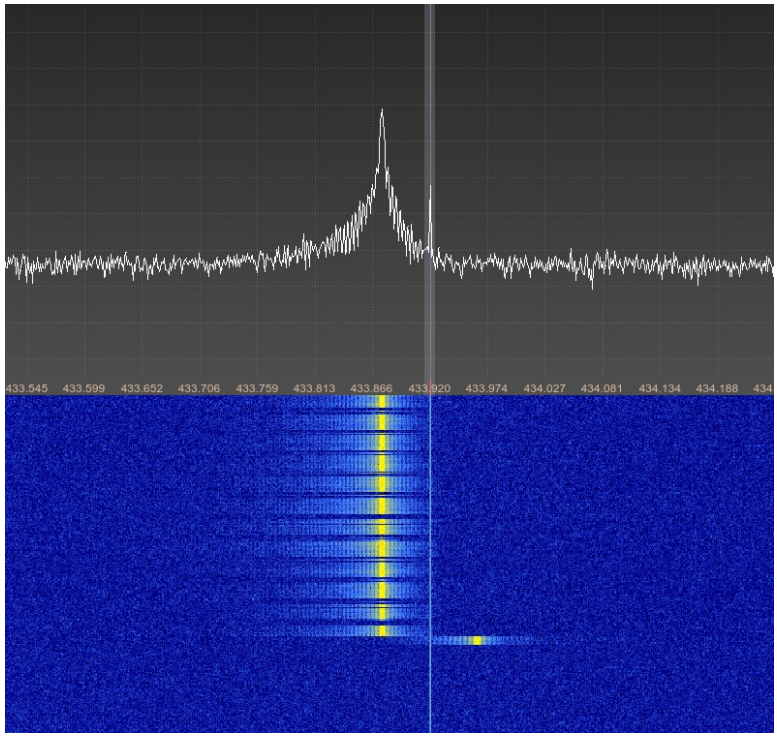

Figure 3.10: Using a Software Defined Radio to show how the device reads the transmitter and then floods the wireless network with "group off" packets

setting the transmission ID to the original transmitter in all packets sent from the device. It will then constantly send out a "group off" command, making it impossible for any of the original transmitters to give actions to their receivers, and thus rendering the whole system unusable. One could alternatively send alternating on/off packets to lights, doorbells and blinds either to the group command or by reading the receiver IDs. The bottom line is that we are now in full control of the system, and can control every individual part of it.

As the security of this system relies on the source ID that is transmitted in every packet in the clear, the security mechanisms can easily be exploited, as shown in this experiment.

3.4.2 Privacy

The privacy implication in this system is close to non-existent. The transmitter will broadcast a uniquely identifiable ID when they are used, but transmission happens relatively rarely compared to other wireless protocols, and since these devices are

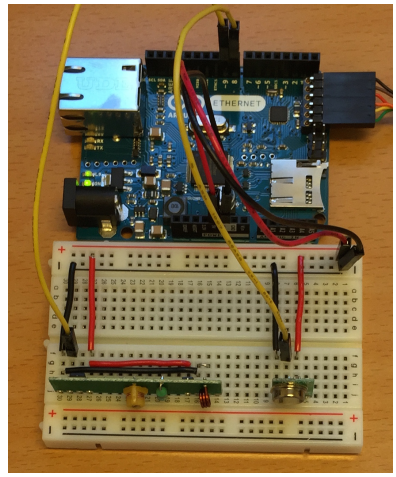

Figure 3.11: The microcontroller used to receive and transmit HomeEasy packets

usually used at home, the possible privacy issues are even less of a concern. One could imagine creating a detailed map of a user's common patterns, but there are other, more available methods of doing this.

3.4.3 Security Implications

The security of these devices have improved since they were first introduced to the market, using a manually addressable, short ID that could easily be brute-forced as only 256 variations were available, and was the same system-wide for sender and receiver. The security in place today is still not effective, as the ID is the only requirement for an action to be accepted by the receiver. Home automation devices based on the Bluetooth or ZigBee protocol will primarily only be vulnerable upon key exchange, and provides a much better security solution than what is implemented in the HomeEasy protocol.

Chapter 4
Challenges in the Internet of Things

The Internet of Things is facing security challenges that differ vastly from regular desktop computing, due to the unique constraints. In this chapter we show the main constraints, what challenges they cause, and challenges that will surface in the future. The challenges that are presented includes some of the typical challenges within IT security such as authentication, authorization, availability and confidentiality, and also challenges such as privacy, usability, DoS and physical security which are not **as** prominent challenges in the more classical computer systems. While other factors such as psychological, economical, environmental and political will impact the future of IoT, and possibly present other challenges, this is considered to be beyond the scope of the thesis.

4.1 Constraints

There are many limiting factors associated with security in the Internet of things, the three most prominent being:

- Processing capability
- Power requirements
- Bandwidth requirements

When faced with these constraints, one will quickly understand that we cannot simply use the same security features as are used in desktop computers without considering how it will impact our devices.

Processing capability is becoming less of an issue as time passes, as increasingly faster chips are developed every year. How much of an impact processing performance has on a device will be dependent on the type of device. If it is not important that the device is ready to receive data or act on events all the time, processing speed will

not be an issue in that regard. Such devices will typically be temperature sensors that transmit on set intervals. If the chip knows that it will need to transmit sensor values every X minutes, it will be able to encrypt the payload in between each measurement. But it the device needs to act on certain events such as motion and need to report this data in real time, it cannot be preoccupied with encryption over a long time.

Power requirements are a big problem with the current Internet of Things. The device's power usage will have an impact on processing speed, bandwidth, temperature and/or battery life. As many of the devices are operating with a battery as the only energy source, it is desirable to have the longest operating time possible, and thus low power usage. But whatever way we look at it, securing data will require increased power usage compared to no security, as some form of computation is required, and all computation will consume power.

But even though we include security mechanisms in the devices, there are many differences between what solutions we implement. There are clear differences in between different encryption algorithms, key generation algorithms, digital signature algorithms and hashing algorithms with regards to power efficiency [RPHJ11]. In addition, different chips will also have an impact on how these algorithms perform.

Bandwidth is often a scarce resource in the Internet of things. To conserve energy and keep heat waste low, one needs to power the radio for as short amount of time as possible. This means that one would like to use as high frequency as possible, and have as small payload as possible to transmit data quickly. But with higher frequency, the range of the wireless radio decreases, and we need to increase transmission power to transmit data over longer distances. At the same time we do not want to waste power by having the signal reach further than necessary. One thus needs to find the best intersection between the desired speed and range for the specific application depending on payload and available power.

Encrypted data will always cause some bandwidth overhead, but the impact on the actual packet size has not been prohibitive until now, as packet sizes have been relatively large in traditional network systems. With protocols specifically tailored for the Internet of Things, on the other hand, the percentage increase in packet size is becoming prohibitive as the packet size is compressed to the bare minimum.

Balancing these three factors is crucial for any device in the Internet of Things. Currently, when designing new devices, one needs to make trade-offs, and decide which to make for each specific case:

To preserve computational power one can offload processing to central servers, but this will require transmitting more data or more often, resulting in energy usage

by the wireless radio rather than the processor. If we instead process the data locally before transmission, we reduce energy cost in the radio, but increase processing and storage cost in the device.

To reduce power cost in the radio, one can use adaptive power control for the radio, but this will require slightly more processing, and can increase retransmissions as a result of increased packet loss if the distance or interference is highly variable.

Another solution often used in IoT is the concept of a central hub, handling communication between two different networks, with different constraints. This concept is implemented in wireless sensor networks, and home automation systems amongst others. The hub is often not constrained by power, but is not placed where there is a need for the service, and thus constrained by the environment. These hubs are usually a link between network connections with long range requiring large amounts of power, like cell-networks, parabola-connections or Wi-Fi-connections, and network connections with shorter range, but low power requirements such as ZigBee, Z-Wave or Bluetooth Low Energy. In wireless sensor networks, devices will report their data to the hub directly, often using a form of mesh networking, and the hub will then forward either processed or unprocessed data to central storage.

4.2 Current Challenges

The current Internet of Things is considered quite simple compared to what is theoretically possible. Many devices will connect to a phone acting as a hub to a central server, connect to a stationary home hub, or connect directly to a central server. Some devices will use a mesh network connected to a hub to communicate with a central server.

During this section, the different current and future challenges in IoT are presented. These are all shown in Tab. 4.1 together with the result of the challenge, what their respective constraints are, and the related solutions that exists.

4.2.1 Authorization

Authorization is the act of granting access to different parts of the system only to devices that should have access. Authorizing a device within IoT have some challenges that does not exist in desktop computing, where the concept of a user with different user-names and passwords will be entered into the service a user wishes to use. In IoT, the user is not actively using the device through an advanced user interface. As devices are physical, they can be lost or sold to other users, and thus the principle of “a single device equals a single user” will not be viable. The concept within IoT can better be explained as a user using both a service and a device, with these three

Challenge	**Result**	**Related Constraints**	**Solution**
Authentication	False data can be treated as correct	Processing, Bandwidth, Power	Authenticated encryption
Authorization	Adversary access to important data and functions	Processing, Bandwidth, Power	Authorization
Availability	Delayed updates, management	Power, Bandwidth	Work with existing restrictions
Lacking multi-layer security	No added security if one fails	Processing, Bandwidth, Power	Use encryption on multiple levels
Key distribution	Key can be snapped up during transit	Processing, Bandwidth, Power	Use case specific. Different solutions.
Prost production Management	Backdoors can be introduced to the system	Bandwidth	Authorization, Authentication, Network encryption
Privacy	Ability to track users		Privacy
DoS	Render device unusable, loss of data	Power, Bandwidth, Processing	Detection, Network design
Unintended uses	Device is insecure because the use case was not predicted		Always prioritize security
Usability Before security	Easy to use, but insecure		User friendly security. Always include security.
Local Storage	Large amounts of local data can get in wrong hands	Bandwidth	Use case specific. Offloading possible?
Local Processing	Local processing will impact ability to do other things	Power, Processing	Increased processing. Offloading possible?
Interoperability	No devices talks with each other, extra layer needed (added breakpoint)	Bandwidth, Processing	Adaption, corporate unity

Table 4.1: Table of challenges, what the result of the challenge is, what constraints they are related to, and the possible solutions.

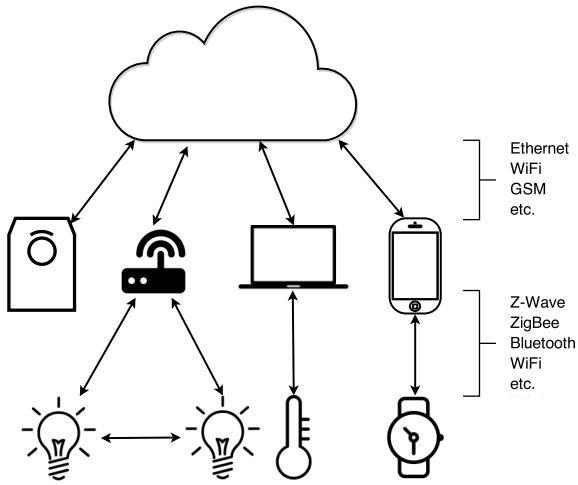

Figure 4.1: The current ways to communicate within IoT. The central server is the backbone of the whole system.

concepts being distinct things, rather than a device and user being the same from the point of view of the service. Thus, both users and devices need to register as distinct entities. For this to be possible in IoT, one needs to use authorization to distinctly define rights for both the device and user.

With improper authorization, a rouge device can easily masquerade an authorized device. Authorizing a device is a problem in the world of IoT, as is evident by for example the BMW incident. The solution implemented by BMW would authorize received requests by looking at the source IP of the request, and accept everything sent with the BMW owned IP-addresses. Using IP spoofing to mimic an authorized device, an adversary could easily get its requests accepted by the car's on-board computer system, as the authorization mechanism is close to non-existent.

4.2.2 Authentication

The IoT market is increasing, and measuring increasingly more of both our personal and public life and the environment around us. The increased gathering of data is improving our ability to make decisions from hard data, where there before was none. But the consequences of falsified or lost data can be severe, and can lead to decisions that results in a worse outcome than if there were no data to base the decision on.

Data received from an authorized device might not be the data that the transmitting device sent. If an adversary masquerading as an authorized device were to eavesdrop on the communication channel and alter data during transit, we would

not be able to detect the change, and thus accept the data as correct. This could in some circumstances have a huge impact on the system, depending on the type of information that is transmitted.

4.2.3 Availability

To preserve energy, IoT devices do not usually hold a persistent connection to a network, and will often enter a "sleeping" state for longer periods of time, depending on their use case. In terms of security, this can be an issue with regards to updating security parameters (e.g. cryptographic keys), and remote monitoring. When a potentially rogue node enters a trusted network, either all nodes or a central console needs to be alerted of the breach. With sleep-cycles as long as days, it will take a long time to alert the other nodes. Depending on the breach, the adversary can stop the alert from propagating to the other nodes when they are woken up through network or physical triggers, when they wake up by themselves, or stop the alert from reaching the central console.

4.2.4 Lack of Multi-layer security (confidentiality)

The concept of securing a device on multiple layers of the Open Systems Interconnection (OSI) stack will introduce added security layers if one of the other layers should be breached.

If a device is physically locked behind a door, a concept of security is introduced in the physical layer. This physical layer security will increase the barrier for an adversary to get physical control of the device, in many cases to such a degree that physical breach is unlikely. At the same time, physical control of a device can be regarded as a complete breach, as all stored data and keys are available either in encrypted or unencrypted form. Various ways of securing the physical layer such as epoxying the chips and adding a tamper-sensor are implemented in devices, but with varying degree of success in providing resistance against an adversary.

Introducing security at the higher link, network and transport layers will secure the communication between devices, or to a central hub or server. Securing these layers are more involved compared to securing the device behind a pad-lock as information can be gathered remotely, and every connected node on the network or Internet is a possible attacker. At the same time, this kind of breach does not necessarily result in as much information loss as a physical breach. There are many ways of securing devices on these layers such as Virtual Private Network (VPN), IPsec, SSL/TLS, etc.

Security on the application level will secure the device on a even higher level. Securing the application layer is on par with the lower layers of the stack (excluding

PHY) both in terms of potential data loss and feasibility of breach. Encrypting stored and transmitted data locally on the device or securing the web interface of the device from Cross Site Scripting (XSS) are examples of common application layer security implemented in devices.

Encrypting data on the application level before transmitting over a secured link will ensure that the data is kept confidential, even if an adversary is to break the network layer security. This is what was found when analyzing the Fitbit tracker in the analysis. As it was easy to read the data between the two network transmissions (Bluetooth and Ethernet), all the data would have been revealed, had it not been for the fact that the data was encrypted on the application level.

While the possibility of the device being physically breached is higher within IoT than in other networked computer systems, it is also less feasible to secure these at a physical level. Sensoring nodes are often placed in public spaces and within sight, for example in buildings or carried on a person, which makes them an easy target. Incorporating motion detectors for alerting is often too costly, or not an option due to environmental restrictions.

Relying on a security mechanism on another level is not a good idea. Ideally one should secure all levels of the OSI model. This means securing the device physically (everything from epoxying the chips to securing the device behind a lock), on a network level, and on the application level.

4.2.5 Key Distribution

Distributing keys over the network is often a precarious part of an IoT devices lifespan. By obtaining the encryption key during transit, one is often able to decrypt future communication to and from the device. The key distribution problem is more of an issue in IoT systems than in normal computer systems, as there are usually few ways for the user to interact with the device enough to input a key through other means than the network.

This is a known problem, and as is evident from the analysis in Chapt. 3, device manufacturers try to solve this problems in many different ways. The FitBit distributes the encryption key through the secured Bluetooth connection, while the EyeFi calculates a key from the device's MAC-address, and puts this key on the packaging of the physical device. Physically distributing the key in the way that the EyeFi does will not make it possible for an adversary to obtain the key through the network, but will rather need to exploit some other mechanism or get a hold of the key physically. At the same time, this will make it hard to change the key if it should be obtained by an adversary. Distributing keys over the network will allow for more flexibility as the keys can be changed and does not need to be generated

during production and physically hidden during shipment. But this means that the network used to distribute the keys will need to be secured during transfer. This is currently the weakest point of many wireless technologies such as Wi-Fi and ZigBee [Wri09] [SR13].

4.2.6 Post-Production Management

Mistakes are often made when designing new products, and they will always happen. Thus, it is usually necessary to implement a way to update the firmware and/or software of a device after it has been delivered to the customer, to fix any potential vulnerabilities that is found after production has begun.

Insecure implementations of management protocols makes it easy to create backdoors into products. When implementing such a feature, it is crucial that it is implemented in a secure way. Firmware or software updates can compromise devices completely, or a nefarious entity can introduce discreet back-doors that cannot easily be detected. As user-interfaces and physical connections are scarce, the updates are often handled remotely over the network, making this an effective remote exploit if not handled correctly.

One of the most widely used configuration and management protocols used today is the Technical Report 069 (TR-069). While many propose this standard as having a bright future, and perfect for M2M communication [For14b], it has shown some weaknesses. Authorization is handled with a single username/password combination, providing no form of authentication. As is often the case, using SSL is only a recommendation, and in testing, only 19% of implementations was found to actually use SSL, and many of them did not actually check the certificate validity, allowing man in the middle attacks [Tal15]. Maybe the most important aspect of TR-069 is that it is a resource-intensive (chatty) protocol using Extensible Markup Language (XML) over Hypertext Transfer Protocol (HTTP) (Simple Object Access Protocol (SOAP)), and thus not suited for resource-restricted devices.

The Open Mobile Alliance - Device Management (OMA-DM) is an alternative protocol, originally targeted at mobile terminals. It does not require SSL either, but does recommend it. Authenticateon is handled in the HTTP header using HMAC to authenticate requests using MD5 as the hashing algorithm, and is thus highly vulnerable to replay attacks if not implemented properly [Sol13]. Also using XML over HTTP (Synchronization Markup Language (SyncML)) and requiring SSL for confidentiality, it is also not well suited for resource-constrained devices in the Internet of Things.

With that in mind, the Open Mobile Alliance decided to create a Device Management protocol specifically for M2M communication, called Lightweight M2M

(LWM2M). The protocol is based on an Object model with defined object templates instead of the resource intensive XML-over-HTTP model used by TR-069 and OMA-DM. LWM2M is based on the principles of Constrained Application Protocol (CoAP), but unlike the CoAP specification, requires that all communication between servers and clients are authenticated, encrypted and integrity protected with DTLS when using the UDP channel [Mob15a]. The protocol is also specified to be used over an SMS channel where "NoSec"-mode is available for triggering, and should only be used for this, but a faulty implementation could potentially use the SMS channel in NoSec mode for all communication [Mob15b]. Requiring DTLS is a good solution for this type of protocol, but DTLS is, at the same time, relatively resource-intensive for the most resource-constrained devices.

4.2.7 Privacy

As was shown in the analyses, there is a potential for privacy concerns in most devices. To further exemplify this, a scanner was set up to filter out Bluetooth LE advertisement frames on a laptop. Over the course of 3 days, and without any strategical placement, over 200 unique addresses were found, shown in Appendix B. It is possible that some of these are the same device using randomization, but judging from experience, this is likely not the case for the majority of traced devices.

To what extent privacy is a challenge will be a matter of individual perception. As long as we are able to uniquely identify a device, we are usually able to identify a user. The topic of privacy is therefore often a matter of perception, as what constitutes a privacy violation can be vastly different between users. For example, the power company knows when and how much power is being used in an area or even individual houses. This data can then easily be used to make assumptions as to when someone wakes up, leaves for work, goes on vacation, etc. An example of this is shown in Fig. 4.2, and the same can be said for any Internet Service Provider (ISP) and mobile network operator. We all know that this data is being collected at these companies as unique identifiers are needed to operate, but do not usually see this as a breach of privacy unless sold to third parties. On the other hand, creators of mobile applications, mobile operating systems, health monitoring devices, and many others are often viewed as violating users privacy.

All devices in the Internet of Things do, and will for the foreseeable future have a unique identifier, as they will be connected to a network in some way or the other. One will therefore always have the possibility of monitoring a specific device uniquely, and usually be able to correlate the use of the device with a specific user. Privacy is a concern within all aspects of computing, but is especially relevant within IoT as devices are often used to measure private data, and many are worn on the body at all times, giving more precise tracking of a user. Depending on the wireless technology,

Figure 4.2: Heat map of power usage versus time. The power company can deduce a lot about its users by using simple analysis. Original heatmap without analysis by: Powershop Australia Pty Ltd

the RSSI value or timing can be used to track the suers distance from the receiver, and using triangulation can give a more exact tracking.

4.2.8 Denial of Service

There are many possible attacks against computer systems, many of which are applicable to the field of Internet-connected devices. As the different devices usually have reduced processing, power and bandwidth capacity, all these attacks are even more effective on these systems than on larger computer systems [CMYP09].

A Denial of Service attack is one of the typical attacks that have a large impact on IoT systems. Since devices are restricted in processing capacity, they do not have the resources to effectively defend against a denial of service attack. And since devices often have little power available, this attack is hugely effective, and can completely drain the battery of a device quickly.

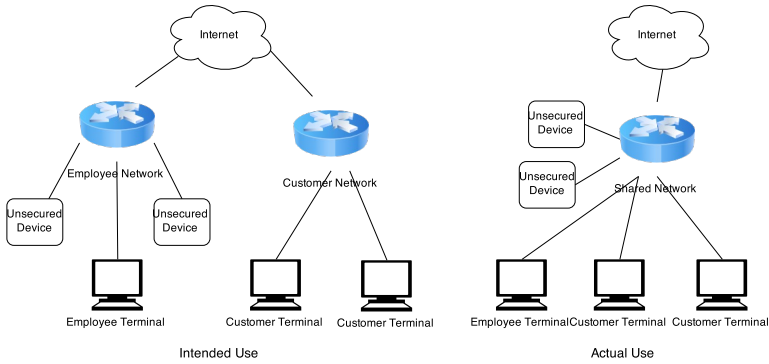

Figure 4.3: Illustrating the Sonos case of an unintended use-case where the developers assumed the device would be used on a secure network

Jamming

A jamming attack is an attack that makes it impossible for devices to communicate over the network by blocking all network communication. This can most easily be described as creating deliberate interference of the signal. An example of this is shown in Sect. 3.4.1 where a development board is used to jam the wireless protocol, rendering all attempts of communication between the devices useless. As IoT devices usually are communicating using wireless technologies, they are especially prone to interference. Since the devices will not be able to transmit or receive their stored data or commands, they will have to store the data that is not sent, and continue trying until they give up. This could increase processing, power and storage usage, and will usually render all devices communicating over this network useless for the duration of the jamming attack.

4.2.9 Unintended uses

While standard desktop computers have been used in a somewhat controlled, standard way, devices within the Internet of Things can be used in ways that was not originally intended by the manufacturer. To make products as user friendly as possible, manufacturers often only include security suited for their specific use case, as setting up strong encryption often adds to the startup-cost for the user.

Audio and video systems often excludes all forms of security in their products, and rely on an existing secured network. An example that one often finds is bars and other public venues that use audio visual systems originally intended for private use. It is starting to become common for bars and restaurants to provide a wireless network for customers, as an added service. When users suddenly have access to the venue's network, the security that all the audio visual systems rely on is suddenly

gone. Networked speakers such as the Sonos system [Son] only requires the user to have an application installed on their mobile phone for them to gain control of not only the venues music, but also all their linked accounts such as Spotify, Google Music, Tidal, Deezer, etc. This problem might be more common than one might think, as this has been the case several places, where we were quickly able to control the music at popular venues after discovering that the audio system was produced by Sonos.

The lack of security in more or less all Bluetooth-enabled audio systems is worth mentioning here, but in practice it seems most people understand that the complete lack of security creates an obvious problem, as these systems are rarely encountered in pubic venues.

In the paper "Internet Census 2012" released by an anonymous researcher (as the methods used are of dubious legality), a lot of devices that were never intended for the public Internet were discovered, quoted as "half a million printers, or a Million Webcams", and gives the following tip to future designers of IoT systems:

"As a rule of thumb, if you believe that "nobody would connect that to the Internet, really nobody", there are at least 1000 people who did." [5]

4.2.10 Usability Before Security

In the case of the EyeFi wireless card, the security mechanisms implemented were relatively secure (WPA), even though in this case the key derivation function was based on known data, and used a mono-alphabetic substitution cipher. In addition to the implemented security, the developers included a feature to connect to the wireless network programmatically, so that the user would not have to enter their password to connect to the wireless network. This exposed how their key is derived from the card's MAC-address, and will be of great value for anyone trying to break their keys.

Similar examples like this are common in IoT, and manufacturers will even exclude security overall, on the foundation that it is a hindrance for the consumer and/or the developers.

4.3 Future Challenges

One should be cautious when trying to predict the future of IoT, as the future can take many different directions. We can safely assume that processing capacity will be increased compared to cost or size of devices as this is a trend that has been ongoing for many years, and does not seem to be changing anytime soon. There are many possible uses for this computational increase, that will impact the future

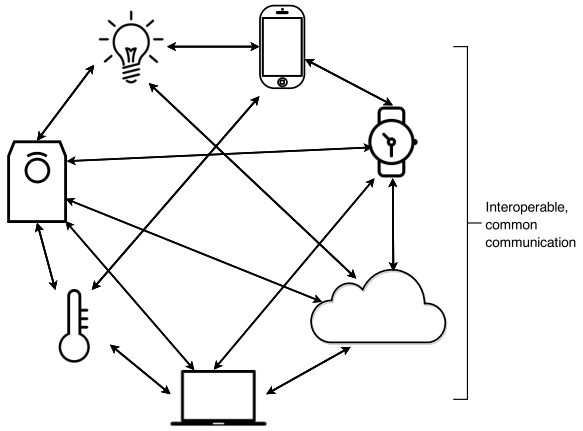

Figure 4.4: The future of IoT as envisioned by IBM, amongst others. What role the cloud will have in the future is one of the main questions.

of IoT. Reduced size can push IoT forward in the medical industry, reduced cost can increase the number of devices in the Internet of Things, the increased process capacity can be used for security mechanisms, or local pre-processing before reaching the central server. One of the possible visions is to reduce the importance of a central server, relying on all the different devices taking directly and doing local processing on data [BI14]. Another is to make the central server even more of a focus point of the Internet of Things, by using all available data to make predictions for the user. Different variations on these two futures are probably more realistic. From a business perspective, one is often interested in acquiring data from its user, so a central database will still be a part of the future of IoT.

Whatever the future brings, we can safely assume that the number of devices will increase, and thus interference and network collision will be an increasing problem.

4.3.1 Privacy

As discussed in the current challenges, Sect. 4.2.7, we will always have to include a unique identifier for devices at some point. And as we can only assume that the number of devices will increase in the future, the number of devices that can be identified will increase. If we envision a future where all devices communicate with each other, and one can have a common user identification for all devices connected to a specific user, privacy breaches will become even more severe.

To keep from transmitting data unnecessarily, event-based triggering of transmission is often used in IoT. When using this technique, meta-data can become a

large privacy concern. Even though the payload is encrypted, the fact that there is a transmission is enough to impact privacy. If there is no consolidated cloud for all the different services, the destination of a trigger based transmission will reveal even more information than one might think.

With the future of IoT comes more sensors, and more collected data. Defining what data should be reported back over the network will be crucial in a privacy setting. If, for example, a device is based on voice commands, most developers would consider central processing of voice audio the best solution, as this processing usually requires a lot of resources. If this communication is not triggered by a specific command or button, most would see this as a huge privacy problem, and most likely be devastating for the product. Sufficiently defining what data should be uploaded, and good mechanisms to initiate transmission, will be even more important as people become more aware of privacy.

4.3.2 Data Storage

With increasing number of devices in a network, and the proposed future of increased device-to-device communication, the need for on-device processing of data will be necessary, and thus, more data will need to be stored locally. In addition, if we are not to have direct communication with a fault-tolerant backend, the storage-network itself will need to be fault-tolerant. Even more so than the backend, as the possibility of loosing more devices within a shorter time-span is more likely. This will require multiple nodes to know data about another node. Storing more or less everything in a distributed fashion between all nodes in a Wireless Sensor Network (WSN) could result in an adversary gathering a complete overview of a network over a long period of time.

4.3.3 Interoperability & Consortiums

Security implementations are an important part of interoperability between products, and important in the unification of standards into one common IoT standard.

In standard desktop computing, machines usually support many different encryption schemes. In the TLS protocol, a client and server will negotiate a common cryptography scheme by “best” common denominator available. The client will send the server a list of its available encryption schemes, and the server will compare that list to its own, choosing the strongest amongst the two. This negotiation is not valuable in the Internet of Things, as devices usually only incorporate one cipher, often dictated by the hardware available. Therefore, a common Internet of Things standard will need to decide upon one, or a few common ways of securing data between devices.

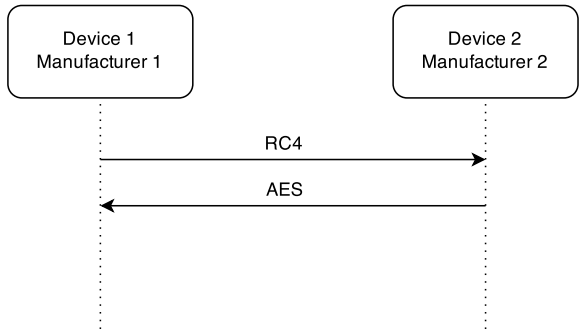

Figure 4.5: Even if two products communicate using the same protocol, they have to agree on encryption scheme

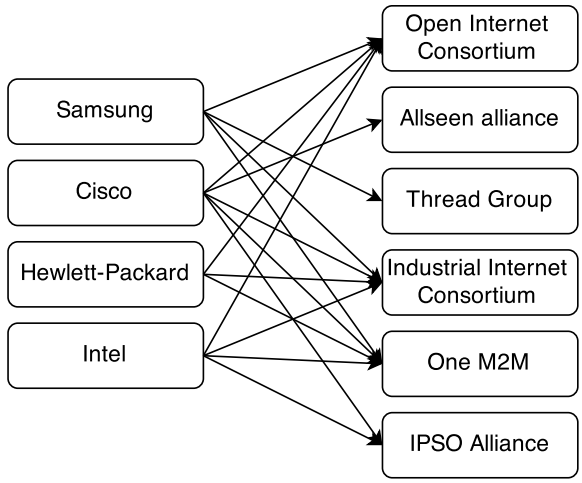

Figure 4.6: Some of the largest companies and their association to the different IoT consortiums

The problem with the consortiums trying to unify the Internet of things today, is that no manufacturer wants to pick a side, as shown in Fig. 4.6. When all manufacturers are joining all or none of the groups, you do not gain anything compared to not having any groups in the first place. At the the moment there is no clear advantage for a smaller company to join any of the existing consortiums. A consortium does not seem to be the solution to this problem, and it is likely the big players in the market are the only ones who will be able to dictate the future of common security mechanisms.

4.3.4 Post-Production Management

It has become more and more usual to handle updates remotely and automatically. If this trend continues it is vital to have simple and secure protocols available to all developers, especially as the amount of devices increases. If we are to play down the role of the central server, the update has to be transferred from other devices in the Internet of Things, and thus, an update that has been tampered with can potentially spread like a virus through the network. With 70 % of all devices not encrypting network communication, and 60 % using no form of encryption when downloading software updates today [HP14], this is clearly a problem today that can become severe in the future.

Chapter 5
Solutions to challenges in the Internet of Things

This chapter will present current and future solutions to the challenges within IoT, and current and future recommendations for securing IoT systems. As the solutions are dependent on the available resources, some topics will include different solutions based on the constraints.

5.1 Authorization

Many simpler back-end systems in the Internet of Things will typically use a unique client key in the header of a request to authorize a device to communicate with the back-end. This will require an adversary to get a hold of the key to access the device, and if an encrypted connection is required to communicate, will be a viable solution. If this solution is to be utilized, the key is required to be transferred to the device, either after the encrypted connection is set up, or during production. Products that can handle a standard HTTP TLS connection will usually use this to communicate with the back-end and transfer the key. Though if the product is capable of communicating in this way, we are closer to a basic networked computer, and might as well use regular account login to the back-end using a mobile application, and receiving a device-specific access token related to the user account.

Authorization should ideally be on device-level, and related to a user as a separate value, as we should be able to exclude lost devices without excluding the user account, and grant different permissions to different devices that the same user owns. As the unique value specifically defines a device, it does not need to be changed during the life of a product, and can be added during production. But as it needs to be related to the user in some way, a system for association still needs to be implemented, and will usually be a part of the key distribution process.

5.2 Authentication

Using authenticated encryption as explained in Sect. 2.3, any changes made by an adversary in transit would be detected as the MAC would not match. For an adversary to be able to manipulate data, they would need to have the correct key used to generate the MAC. Popular authentication algorithms include MD5, SHA-1, SHA-2 and SHA-3/Keccak. An ongoing competition called Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) is currently running to provide new authenticated encryption algorithms [cr.15].

5.3 Reduce Bandwidth Overhead

To reduce the bandwidth needed, one wants to transmit data only when needed and keep the packet small, while still keeping the need for retransmissions to the minimum. Several protocols are designed specifically with this goal in mind, but their use is often limited to a specific use case, and no **one** general solution exists.

5.3.1 Mesh Networks

Within mesh networks, we want to route data quickly to the receiver node(s) while keeping traffic to a minimum. Several different methods for transmission in a mesh network are suggested in the research community, but some proposed solutions will only work for specific types of mesh networks (who needs to know what?).

Securing the networked devices within a mesh network can be handled with varying degree of overhead. To secure all connections within a mash network, one should ideally secure the direct link between each device, but again this will be dependent on what mesh technology is used.

Intrusion detection is an integral part of every mesh network. The ability to exclude individual nodes in a network should be possible within a reasonable amount of time. In a mesh network, intrusion through a border gateway between the local mesh and the external Internet should be detected on the gateway, while intrusion into the local mesh should ideally be detected by the mesh nodes themselves. This is not always possible, as an adversary can get hold of the keys to the network, and masquerade as a new trusted node connected to the network. In these cases, excluding nodes in the network should be possible when the breach is discovered through an alerting or monitoring system.

Monitoring the state of the nodes with regards to their integrity will generate extra packets in the network. But as devices usually transmit additional data such as battery state and network state, this does not increase the load on the network significantly. If the devices are severely constrained in both power and bandwidth,

one will often use a heartbeat to detect whether devices are still alive and/or present in the network. In these cases, one would need to use the central monitoring service to detect abnormalities in the network. If all devices are controlled by one entity (usually the case in industrial or research settings), inclusion of new devices can be manually controlled. This is not a scalable solution for the future of the Internet of things as the number of devices can quickly reach an unmanageable amount.

By utilizing an announcement layer in the mesh network, one is able to transmit information between the nodes in a mesh network in a power-constrained environment efficiently. The packets will piggyback on the required packets already transmitted between the nodes. A further power-saving feature is to use point to point connections instead of relying on a broadcast solution. This will require less radio time, but might include more local processing to ensure data correctness [DMT+11].

5.3.2 Optimized Protocols

Using the web stack (TCP/IP HTTP REST APIs) is still very common in IoT, either between the whole system, or as a part of it (hub communicating with a back-end server). Using this common mode of communication provides more "features", and easier interoperability than the specialized protocols, but at the same time they require more resources, and has a huge overhead.

Several protocols have been proposed to solve the bandwidth overhead problem in IoT systems. In recent years we have seen the emergence of protocols such as Bluetooth LE (Smart), IPv6LoWPAN, MQTT, CoAP, DDS, AMQP, STOMP, and many more. These protocols strive to keep packet sizes to the bare minimum, while still providing reliable communication. This is achieved by removing any header and payload that is not strictly needed for communication in IoT. When the focus is strictly on reducing the packet size, security mechanisms are not always a priority. Therefore, one should evaluate whether the the theoretical reduction in bandwidth translates to measurable decrease in packet size when security is added to the protocol. MQ Telemetry Transport (MQTT) is a protocol that uses the subscriber publisher method of transmitting data. All data is sent from a device to a message broker point-to-point, where the broker broadcasts this message to all devices that subscribes to the specified topic. By utilizing this method, in theory, the only required elements of the packet is the topic, message type and message (QoS level, DuplicationFlag and Retain fields are included for added reliability). With the strict focus on reducing packets, there is absolutely no focus on security in the MQTT protocol. But even with this apparently glaring flaw, it is still a viable solution to the future of some IoT systems. On the webpage of the MQTT organization it is acknowledged that no security is implemented in the standard, and that "traditional" SSL might be used to secure communication in MQTT, although this adds huge

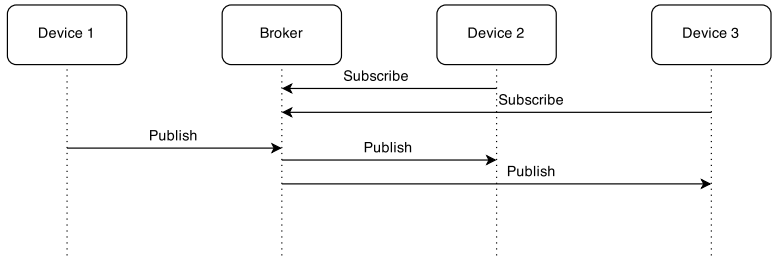

Figure 5.1: Illustration of the publisher / subscriber method used by MQTT to reduce bandwidth overhead

bandwidth overhead [MQT]. It is also mentioned that you could encrypt data that is transmitted over the protocol without encrypting the communication channel, but this is not in line with multi-layered security that is proposed as an important part of IoT security going forward.

Not focusing on the security aspects of their protocol is a choice that the MQTT organization has made deliberately. The thought is that the transmission protocol and security protocol should be kept separate, and that security should be implemented in parallel, not in the same product. The upside to this approach is that security can be provided by experts within the field, that overhead of the security protocol can be reduced in a different pace than the transmission protocol, and that common security mechanisms (such as SSL) can be used if desired. A clear drawback of this approach is that the startup-cost of securing MQTT is increased considerably, and developers might not consider security as important because of the added burden.

A newer protocol that considers both packet overhead and security is CoAP. It uses User Datagram Protocol (UDP) on the transport layer to decrease the overhead created by Transmission Control Protocol (TCP). Even though it is possible to run CoAP without any form of security, it was designed with the intention of using DTLS to secure the connection [ZSB14]. This would lower the barrier for developers to use secure connections by facilitating for the use of DTLS within the distributed software. In reality, this has shown not to be the case. In a plugtest report from 2013, it is acknowledged that there were too few implementations that support DTLS to get any significant data from tests [EO13]. Even though the standard facilitates security from the beginning, it is the actual implementations that in the end will have to include secure communication methods. The implementations that exists have shown that more than 1/3 of the packet will be taken up by DTLS, thereby reducing the available space in each packet significantly [Juc12] [HB12].

It is worth noting in the context of optimized protocols that the IPv6 standard

originally mandated IPsec, but was redacted in 2011, as "some devices may run on extremely constrained hardware (e.g., sensors) where the full IPsec Architecture is not justified" [EJN11], showing that the Internet of Things was a part of the considerations when creating the IPv6 specification. The new HTTP 2.0 protocol does not mandate HTTPS in the specification, but large web-browser distributors like Google (Chrome) and Mozilla (Firefox) has decided to require HTTP 2.0 to use SSL/TLS for all connections [Ste14], thereby utilizing the performance-gain in HTTP 2.0 for security instead of faster browsing. These examples show the power which the implementors have over these protocols.

Proprietary protocols such as Bluetooth have shown to be used in a more secure way than the open-source protocols, as the group can mandate how the protocol is implemented in a tighter way. Others wishing to use or implement the core specification will usually need to acquire a license, thereby restricting how the specification is implemented. In the case of Bluetooth, security is an optional feature, but is implemented in chips or libraries developers use, as the manufacturer often will want to implement the whole specification.

The newer Bluetooth Low Energy (Smart) protocol has seen wide adoption because of its relatively high bandwidth compared to power requirements. This protocol includes a security standard, but was quickly shown to have weaknesses. As much so that there are software released to automatically crack the BTLE encryption by sniffing the pairing process between two devices [Rya13]. After we analyzed the pairing process of some popular BTLE devices (Fitbit, Pebble [Peb]) using an Ubertooth One [Oss], it quickly became apparent that several vendors deviate some from the standard to create an added barrier from the released automated cracking tools. Released in December 2014, version 4.2 of the LE protocol fixes many of the potential vulnerabilities, especially in the pairing process [Gro14a]. In general, the benefits of Bluetooth LE are high, and the security concerns are not completely devastating, at least compared to many of the other solutions deployed in the Internet of Things.

5.4 Application Layer Encryption

There is a clear correlation between the main processing power and how data encryption is handled in devices. When the main processor has low processing capabilities, the device will likely either use hardware-accelerated encryption provided by the microcontroller, or implement no encryption at all. This is often the case with low-power devices that are supposed to run disconnected from the electrical grid for a long time. An example of this is the Fitbit health monitor analyzed earlier which use an ARM Cortex-M3 (STMicroelectronics STM32L 151) with accelerated 128 bit AES.

Many devices that are not constrained by processing capacity or power requirements will utilize any encryption technique that fits the device or service in question as they are not required to use any built-in encryption to conserve resources. These devices are often smarter devices and might even utilize a full fledged operating system, where developers are able to decide what security should be implemented without being (strongly) dictated by the chip specification.

5.5 Public Key Infrastructure

Increased use of Public Key Infrastructure will allow for better control of devices within a network. If a device is compromised, one can invalidate the certificate of the device from the network in an automated or manual fashion easily. The feasibility of introducing PKI into the Internet of Things is a heavily discussed topic, and while not widely in use today, it is considered a viable solution for the Internet of Things.

The main issue with implementing PKC in the Internet of Things is that the constrained devices would use too much resources to validate the key of another device, compared to using a pre-shared key [RALS11]. Using ECC in PKI will lower the processing requirements and thus the power requirements while providing greater cryptographic strength compared to RSA [Age09]. With the cost reduction of the main cryptographic operations of ECC (scalar point multiplication) in the later years, the validation process is becoming negligible compared to the security gains by using PKC. To gain additional control of a PKI, one can use online validation using the wireless network and central CA, as opposed to preloading the CA's certificate on all devices.

5.6 Post-Production Management

By utilizing authorization, one can make sure that the update is coming form the product developer, and not an adversary. Including authentication will allow the device to reject any altered firmwares with potential back doors.

Many older products will typically include a USB port through which new firmware can be added to the device (TVs, speakers, set-top boxes, etc.), requiring explicit human interaction to initiate the update. In this case, the update is downloaded using a desktop computer with all cryptographic functions required to make sure the update is the correct one.

Newer products tend to initiate updates over the wireless network. In this case the update is either initiated by the user, or initiated automatically from the developers. User initiated updates will typically include using a mobile phone to initiate the download, or pressing a button on a user interface. Systems using the mobile phone

as an initiator will often use the phone to authenticate, authorize and download the new update before transferring the update to the device over the local network. Even though it is positive that there is some form of authorization and authentication, it is useless if the device is used in a shared and/or open network setting.

Devices that initiate updates directly on the device should authorize and authenticate the update on the device itself. Devices like TVs might have an account on the manufacturers service, and allow TLS connections to the update server. This is of course restricted to the more capable devices.

Using the LWM2M protocol mentioned in Sect. 4.2.6 is a good candidate, but requires relatively resource-intensive and preferably use UDP and CoAP at the moment. If the device can support SSL/TLS, many off the shelf protocols can be used. If the device is severely constrained, connecting the device to a computer using a dongle, or connecting physical storage medium to the device will be secure alternatives.

5.7 Privacy

Privacy is a difficult topic within IoT. On one hand, we want to be able to recognize all devices uniquely, but at the same time, the user of the device often does not wish to be recognized.

Wireless networks such as Wi-Fi or Bluetooth use unique identifiers (addresses) for each network interface, which is broadcast in every transmission to or from the device. These unique identifiers was shown in the analysis of the EyeFi card, where the network interface broadcasts a MAC address with each packet, in the Fitbit where the device address was unique could track a persons movement, and in the HomeEasy system where the ID of the transmitter is broadcasted with each signal.

When one wants to send data specifically to one receiver, this identifier is used as an address for the receiver to be able to identify the sender and vice versa. In broadcasts directed at anyone who wants to listen, however, these addresses are not really needed, and excluding or randomizing these can provide privacy for the user. This feature is already incorporated into the Bluetooth LE standard when broadcasting advertisement frames (although not commonly implemented), and are used by some vendors for Wi-Fi probes (e.g. Apple's iPhone).

Where this feature does not prohibit the use of the product, it should be implemented. But in many circumstances, the uniqueness of the device ID is the whole point of the device. An example of this is Bluetooth "bag/key/remote-trackers". The possibility of randomizing the address to avoid privacy concerns is not a possibility, as uniquely tracking the device is the selling feature of a Bluetooth tracker.

5.8 Focus on Security Across All Products and Services

The reality of security considerations is that people tend to make the wrong assumptions, and does not see all the potential damage that can be made. If you where to ask a person what is more important to secure; their front door, light, TV, stereo or Wi-Fi, people are in general are afraid of someone getting in through their front door.

As a developer of new IoT products one will then focus on securing what oneself perceive as the most important. Looking at existing solutions for securing the front door of homes such as the August Smart Lock [Aug] and the Kwikset Kevo [Kwi], there is a clear focus on how secure the products are on their webpages. Of course locks are inherently a security-device, so it is not surprising that their focus is on security. But in general, a door lock will not provide much security when a home has glaring back doors such as literal back doors, or glass windows. This is why safes has been used throughout centuries to secure valuable items. On the other hand, security focus in IoT devices in the home is almost non-existent, even though they often contain access tokens or credentials to user's accounts. (case in point is the Sonos system mentioned in Sect.4.2.9)

Even though this is an excessively assertive example, it shows why security considerations are hard. Even though a developer does not consider the device "important" enough to secure, this might not be the case in reality. And even though one development team has decided that the system really needs no security mechanisms, future additions to the product might create new requirements that one cannot foresee in the first version of the product. Security should thus always be considered a necessity, even though it does not seem necessary from current requirements.

5.9 Data Storage

When designing an IoT device for the future, one should consider the possibility of physically losing a device. Securing the devices 100 % will never be realistic. Thus the amount of data stored locally on the device should be considered in relation to the importance of the data remaining hidden from an adversary.

Data stored on each device should ideally be the bare minimum that is needed. Unless the data is strictly required for local analysis, or is not yet shared elsewhere, there is no need to have it stored on the device. As the Internet of Things is storing personal data from consumers, security is, and will be, a clear focus amongst consumers. Reducing the amount of local data should be prioritized, compared to Somewhat reducing the energy efficiency of the device. When local computing is

performed on a device, raw data should not be kept after the data has been analyzed.

Encryption on locally stored data can be used, i.e. securing the key in an encryption chip, but ultimately, the key will need to be stored on the device, and thus only adds some resistance for adversaries that needs to extract the key to decrypt the data.

5.10 Increased Computational Speed

We can only assume that the speed of processors and microcontrollers will increase in the coming years, at least in relation to power consumption. This will make it possible to implement stronger cryptographic algorithms, without increasing power consumption.

The special constraints existing in the Internet of Things means that computational power will be significantly lower then that of desktop computers for the foreseeable future. But as long as strong cryptographic algorithms are used, this computational gap will be minuscule compared to the computational power difference between bruteforcing an encryption algorithm, and encrypting data using the encryption algorithm.

5.11 Interoperability

As was presented in Sect.4.3.3, the standards groups have until now caused more problems than actual solutions to the interoperability problem. The closest we have come to an interoperable standard is services like IFTTT (IF This Then That) and Twitter, which is supported by most devices on the market, although with varying degree of usability. There are some proprietary corporate solutions such as Apple's HomeKit and HealthKit that tries to get vendors to support a common standard for control and storage of data. Their HomeKit[1] solution has yet to be launched, and large players in health monitoring solutions such as Fitbit, has publicly announced [Fit14] that they will not support the HealthKit standard. This problem is sadly a question of economics and competition more than technology as a set of common security protocols is definitely a possibility, and has already been made in the different consortiums. Thus, the true solution to this problem lies beyond the scope this thesis, although it will impact the future of security in IoT systems in a big way.

[1] One of the key selling points of HomeKit is mandatory end-to-end encryption, hopefully forcing more manufacturers to prioritize security

Chapter 6
Designing Devices for the Internet of Things - a Guideline for Developers

This chapter serves as a guideline for developers designing products for the Internet of Things. It will chronologically go through the special considerations that has to be taken into account, and the important questions that needs to be answered to maintain adequate security when designing IoT devices.

There are many considerations that one has to think of when creating secure devices for the Internet of Things, and no *one* solution will work for every type of system. When considering how to design a new system, the first focus should be on the three main constraints of IoT devices; processing capability, power requirements and bandwidth requirements. If none of these are a constraint to the system, the system can be designed with the mindset of regular desktop computing (typically HTTPS over TCP/IP). But if you are not constrained by any of the three, then you are probably not creating a system within the Internet of Things (excluding backend systems).

6.1 Power - The Main Differentiator

The main differentiator both today, and for the future, is in the power requirements of the system, since processing capabilities and networking often will be dictated by the power available. The power requirements can be split into three groups: systems that have unlimited access to power, systems that have limited access to power, and mixed systems.

Systems that have unlimited access to power is typically home appliances such as washing machines, stoves, AV-systems, cars, and all other systems that are constantly connected to a large energy source. Systems with limited access to power is typically smart watches, health monitors, trackers, and other devices connected to a limited energy source such as a relatively small battery. Mixed systems are systems that will have some parts of the system connected to a reliable, large energy source, and some parts connected to a limited power source. This is typically home automation

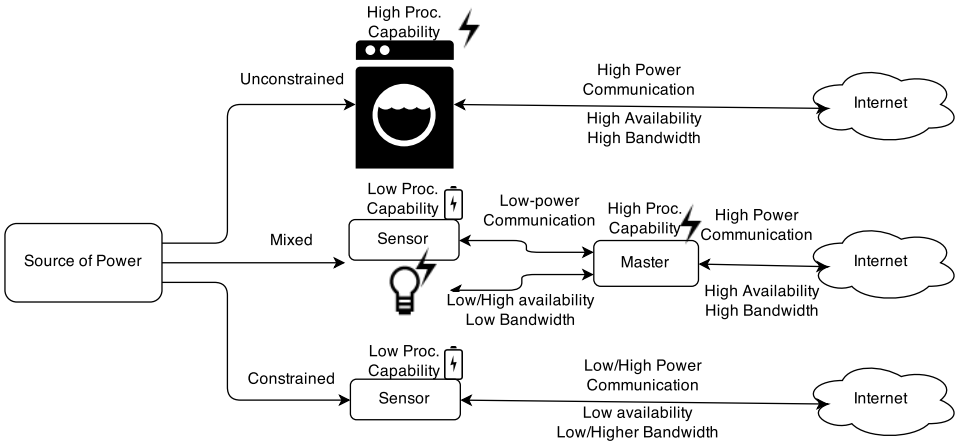

Figure 6.1: Illustrating the different categories dictated by the available power-source

systems, sensor networks, etc. and is done to utilize the features of the systems with a large power source, while also gaining the flexibility of the systems with a limited power source.

The security mechanisms that can be implemented in these different systems are somewhat different. An **unconstrained** system will often have many likenesses to a desktop computer. These devices often use Wi-Fi or Ethernet to communicate over an IP network, as there are little restrictions on power or placement. Wi-Fi will provide long range, easy integration with other devices, and will be able to sustain a constant connection if needed. Even though these devices are not restricted by energy consumption or bandwidth, processing capabilities will usually be kept to a minimum to keep cost, heat and size small. Thus, the pure computational power available can be as low as with devices with a limited energy supply.

Devices within a **mixed** system will usually use a low-power communication channel to the more capable device. Using BTLE, ZigBee, Z-Wave etc. will usually include their own network security platform in the wireless controller. The more capable devices will then have a high-power connection to the Internet such as Wi-Fi, Ethernet, cell (LTE/HSDPA/UMTS/EDGE) or WiMax. The availability of the capable device(s) will usually be high, but connection to the more constrained devices will usually be dictated by the constrained device, and thus, forwarding information to all devices will not be immediate.

The **constrained** devices will often have high power radios, connecting directly

to the Internet using a mobile network or Wi-Fi-radio. These devices will connect to the Internet very rarely, usually only to transmit some form of data. An example of this is home locks or the Dash button previously mentioned, that will connect to the Internet upon a triggered event only. Their availability will thus be extremely low.

Looking at the devices that has been previously analyzed, they can all be separated into the three groups. In the group of unlimited systems we place the automobiles, as they have a lot of power, bandwidth and power available to implement security-mechanisms close to what is used in desktop computing. Examples such as the Sonos system will also fit inside this group, not being restricted by neither bandwidth nor power. The group of mixed systems will include the Fitbit connecting to a capable computer or cellphone via BTLE, and the HomeEasy devices connecting to the central hub. The Fitbit is constrained in power and bandwidth availability, while the HomeEasy devices seldom are restricted by power, but usually by processing power and bandwidth, except from the central hub. The last group of constrained devices will include the EyeFi card as it is using a high-power radio, little power available, and low bandwidth availability.

6.2 Hardware Considerations

Therefore, all security mechanisms should be considered at an early stage in the development-process, and taken into consideration when deciding on chips and designing the Printed Circuit Board (PCB). Including hardware-accelerated security in the prototyping stage has been made easy by dedicated PCBs with cryptographic Integrated Circuits (ICs) released for all the major prototyping boards [Dat14]. Open source initiatives are also making hardware cryptography more available for developers [ea]. Including dedicated ICs for cryptography can be costly, especially compared to increasing the processing capability of the microcontroller and implementing all cryptographic functions in software. A more common approach is to have hardware-accelerated cryptography implemented by the manufacturer of the microcontroller. In some situations, for instance if you are restricted to a specific architecture, dedicated cryptographic ICs will be a good, if not the only, option.

What cryptographic abilities one includes in a device is strongly dictated by the category the device falls under. In the best case of available resources one should include the ability to encrypt data with a (relatively) strong encryption, ability to authenticate data, and a signature algorithm. By including encryption, data can be encrypted on the application layer before transmission, using the authentication algorithm, data can be authenticated to avoid data injection, and using the signature algorithm, certificates can be used for PKC. If resources are constrained, chances are that a PKI will not be suitable for the situation, and the signature algorithm can be omitted. In any system type, we will need to distribute key(s) to the device in a

secure manner. Different solutions to this problem exists, but the goal is to transfer the key securely while still being user friendly. Even though we are talking about an initial transfer, sending the key unencrypted over the network will render the system insecure, as you cannot know if an adversary has acquired the key in transit.

6.3 Key Distribution Best Practices

A fairly common solution for low-powered devices that does not have any way for the user to interface with it, is to have a key defined during production. This solution can be a viable option, but not recommended, as it makes changing the key almost impossible if the key is compromised[1]. One could imagine resetting the key during a firmware update that requires physical interaction from the user, and communication through other means (e.g. a desktop computer), but if such a process exists, this process should be used for key distribution in the first place.

If the device have some form of user interface such as one or more buttons, one or more Light Emitting Diodes (LEDs), an Liquid Crystal Display (LCD) display etc., this can be used to input the key into the device by the user. LCD displays are common for smart watches, home appliances, fitness devices, and has been used to verify a pairing (common in Bluetooth devices like the Fitbit), or to display a unique key. LCD displays with increased resolution and contrast can be used to display a unique image (like the Apple watch), or a Quick Response (QR) code that can be optically scanned by a more capable device such as smart-phones or computers using the built in camera in these devices.

Initiating pairing on constrained devices should include some form of user interaction, as a device should not be accepting pairing requests without a user's consent. A simple button press is enough to put a device into pairing mode, but if not even a button is available on the device, consider using one of the sensors, and a clear user interaction that could not be misunderstood as normal values. Using an accelerometer and requiring the user to flip the device upside down is an example of such an interaction using existing sensors.

Verifying pairing does not require a very long identifier, so other methods can be used to achieve the same without the use of LCD displays. Using RGB LEDs, either blinking, solid, or off, will provide 9 different states per LED[2]. With four LEDs, we have a total of $9^4 - 1 = 6560$ different combinations (including different blink speeds or colors will of course increase the number of possible combinations). Using this method, we are able to verify a pairing-process, but this is not enough

[1] Some manufacturers of hardware encryption chips will not allow for keys to be changed, only initially set, which will influence this choice somewhat

[2] Red, Green, Blue, White, Blink R, Blink G, Blink B, Blink W, OFF

combinations to define a unique encryption key. The same effect can be implemented using a single, single-color LED blinking in a number sequence[3], but this method will quickly become useless as the length of the number increases. In this instance, security is prohibiting use of the device to the point where it will become unusable for many users. One should not focus on security before usability or usability before security, but rather create a nice balance between the two as we need both usability and security.

A method that is used more and more in IoT is transmission via sound from a more powerful configuring device such as a cellphone or computer, to a lower-power device. The configuring device will encode a unique key as sound played through its speakers, and an integrated microphone on the receiving device will pick up the sound which is then decoded into the key. This is more common on cheaper devices with extremely limited user interfaces such as the Amazon Dash Button [Ama15].

The method of including a key into products by setting up a direct, unencrypted connection out-of band is a common method used in constrained devices. The out-of-band channel is often a short range communication channel such as NFC, RFID, Wi-Fi or even light or sound. To set devices into unencrypted setup mode should require physical user interaction. In many cases this can be done by pressing a button on the device, but other solutions such as turning the device upside down has also been implemented (e.g. the Supermechanical Twine sensor[4]) using an internal accelerometer.

In the end, we are restricted to what we are able to measure on the device. The common channels for out-of-band transmission are electromagnetic waves, electromagnetic wired, light and sound. Some more eccentric solutions include acceleration and moisture, and we can even use existing unique keys from the human body such as fingerprints, breathprints [SKZ13], face-recognition, voice-recognition, heat-patterns and many more [JHP00].

Whatever method is chosen, it is important to acknowledge that the encryption key for symmetrical ciphers, and the private keys in PKC must be distributed in as secure manner as possible. This will always add an extra step to the setup-process for any device, but it is nevertheless an important step in any IoT system.

6.4 Post-Production Control

In early stages of the design process, it needs to be considered if, and if so, how the firmware of the device will be updated after it has left the production line. Being

[3] $Blink \rightarrow Blink \rightarrow Pause \rightarrow Blink = 21$

[4] http://supermechanical.com/twine/

able to update the firmware post-production will make it possible to fix potential flaws even after the user has started using the device. While this potentially will fix any flaws detected during the life cycle of a product, it can at the same time open up a new security flaw in the system. If an adversary is able to rewrite the firmware of a device remotely, the whole system can potentially be compromised.

Therefore, security mechanisms should be put in place that prohibits firmware not designed by the manufacturer to be loaded onto the device. Using authentication and certificates is a viable solution for this [Lor11] if the device can handle it. In the cases when this is not possible, we are no longer talking about Over The Air (OTA) updates, and a computer is needed anyway. As long as the computer is not compromised, and has a trusted connection to the manufacturer's server, the firmware update can be handled this way.

6.5 On-Device Storage Best Practices

With the increased amount of data gathered by sensors, the need to store and process data is increasing as well. How this should be handled is deeply dependent on the amount of local processing that needs, or can, be done on the device. In many situations there will be a trade off between energy efficiency and local storage of data Some examples include:

- How often should data be sent to the server from this device?
- Is data requested so rarely that it should rather be uploaded to central storage, and downloaded when needed?
- Are we OK with losing data, or reducing the redundancy of this data?

Using the Fitbit analyzed in Sect. 3.1 as an example, some decisions had to be made on the part of the developers. As this is a device that is designed to be worn by a user, and thus likely to not have access to the network most of the time, data will have to be stored locally. There will always be a limit to how much data can be stored on a device, and how much data should be included in long term logs. The developers of the Fitbit tracker decided that minute-precision data should be stored for 7 days, and daily summaries should be stored for 30 days. When the device finally gains access to a network, data is sent immediately.

Not only will this make it possible to offload processing to the backend server (the device's ability to analyze a user's sleep-pattern is conducted on the backend using minute-precision data) and reduce the need for local storage capacity, it will also reduce the amount of possibly sensitive data stored on the device. If one where

to gain physical control of a device, the historical data stored locally is not enough to gain the full picture of a users life. In the case of the Fitbit, it does not make sense to regard the data of the day as sensitive, as this data is displayed on the LCD display of the device, and thus will be readable by anyone in physical control of the device.

If we look specifically at mesh-networked devices where the network of sensors in total gathers huge amount of data, it does not make sense for every sensor in the network to request a backend service individually with every new measurement. In this scenario, one would like to distribute both storage and processing. Distributing processing, means that we need to distribute raw data. One or more rouge nodes strategically placed in the network would be able to access all data in the network. Thus, authentication, authorization and integrity need to be implemented in between the nodes as well as on the connection to the backend when distributing data. Incorporating an announcement layer to find routes and distribute information about the nodes in the network would not need to have the same properties, and will thus make safe distribution within the network physically possible.

6.6 Privacy Best Practices

One of the most devastating outcomes for an IoT device is ending up in the media as publicly advertising private data, or even something simple as gathering data unnecessarily. A study from 2014 [For14a] found that 69 % were concerned about data breaches, 63 % think that privacy and trust are legitimate concerns, 48 % will hold the manufacturer responsible for any vulnerabilities, and 62 % would feel "completely violated and extremely angry to the point where I would take action." if anonymous data is collected and shared with a third party.

When designing an IoT device, there are some key questions one needs to think about regarding privacy:

- Are more types of data collected than necessary?
- Are data collected more often than necessary?
- Are more data transferred from the device than necessary?
- Are data transferred more often from the device than necessary?
- Are there existing ways to achieve the same with less privacy invasion?

An often encountered example of this is location data. As the first question asks, is this data really necessary? Many IoT devices will report location information back

to the server without needing this information for anything other than promotional or localizing reasons. Even if location is needed, many devices and/or services will be as well off with using geofencing on the device as opposed to constantly reporting the location back to the central server. If precise location data is needed, how often does this data actually need to be logged, and how often does data need to be reported back to the server? These questions can, and should should be used for any type of device with any type of sensor.

Chapter 7
Discussion

In this chapter we discuss the extent of the thesis, and other topics not discussed in the main part.

7.1 Other "Outside" Challenges Will Impact Security

While the challenges, solutions and recommendations presented in this thesis try to encompass all important aspects of security in IoT, it is acknowledged that not all variables of product design are included such as economical, social and philosophical variables. They have intentionally been left out of the thesis as these topics are regarded as outside the scope of the thesis. It has, for example, become more common for companies to set a cost on potential security problems and comparing them to the cost of incorporating security into products [Gua12]. In this comparison, the cost of securing a system is often considered "too costly". By presenting the unique challenges and questions for developers, the focus on security and potential damages will hopefully become more clear and common amongst manufacturers, and they will hopefully stop doing cost-analysis on not securing devices. Some of the psychological questions are tackled by the privacy-focus in the thesis, presenting the different view both developers and users have on security.

7.2 Even Simple Solutions Will Help

While the solutions included in this thesis are comprehensive, and describe how security should be handled, sometimes even simpler solutions exists. The state IoT is in now, with 70 % unencrypted communication [HP14], and application level encryption usually missing, we are currently at such a low level of information security that at least doing **something** will improve security. In the bar-situation where the audio system and music streaming-account was accessible without security, even a simple password would stop random guests from accessing the system. It would not stop someone with some knowledge of networking, but taking control would be more

involved than simply by opening an application on a phone. As discovered with the Fitbit, Pebble and other Bluetooth Low Energy devices, changing the pairing process slightly makes automated software such as CrackLE unusable [Rya13]. This will also not stop someone with involved knowledge of the pairing process, but the security by obscurity makes it impossible without any knowledge of Bluetooth. Cell-networks have not publicly been as big target as for example WiFi-networks, possibly because of hardware cost and potential harm that can result, and is therefore a barrier for many to attack. While these networks are encrypted, data should still be encrypted on the application level in the same ways as should be done in for example Bluetooth networks.

Chapter 8
Conclusion

In this chapter we summarize and restate the main conclusions of the thesis.

8.1 Summary

Security in the current Internet of Things is not as good at it ought to be. This thesis shows some glaring flaws in existing products, which is often created because of oversight from the developers, as the constraints existing in IoT requires a more thorough thought-process than is normal in desktop computing. Due to limited power, bandwidth and processing power, everything needs to get stripped down to the bare minimum, while still maintaining good security properties.

Security is an oversight in many projects. Using examples from previous research, and conducting unique analysis on existing products, it is shown that many developers more or less ignores everything related to security (BMW, HomeEasy, Sonos), or creates their own cryptographic algorithms with clear flaws (Eye-Fi, OSGP smart-grid). To ensure that the future of IoT is secure, this thesis aims to make developers think about the limitations that exists, and provide solutions to the problems that will occur when designing a device for the Internet of Things.

Securing the Internet of Things is important to consumers. Through previous research it is shown exactly how devastating not focusing on the security of IoT devices can be, with the majority of consumers (62 %) "feeling completely violated and extremely angry to the point where I would take action.". Close to half (48 %) of all consumers would hold the manufacturer responsible if a flaw was to be found in the system, showing the obvious economical risks taken by not securing a device properly.

Some of the topics of challenges presented are common in information security, but poses new challenges because of the unique constraints. Securing an IT system requires confidentiality, integrity, and authorization. Where this usually is handled

by libraries like OpenSSL and using TLS int desktop computers, deciding on an encryption, authentication and signature algorithm is not as easy as calling a different method. The limited power, bandwidth and processing capabilities will require a thorough thought process to decide how to both efficiently, and effectively secure a device.

The other challenges are more specific to the Internet of Things. In regular desktop computing, an advanced user interface is usually available, and physical loss of a device during use is relatively uncommon. IoT devices will on the other hand usually have a really limited user interfaces, and will often be placed in exposed areas and used in situations with high physical stress.

Security should be a consideration through the whole project. Long before the first prototype PCB-design is sent to the factory, key decisions on security should have been decided. These include how keys should be distributed to each device, if hardware-acceleration should be used, how updates can be handled, if PKI is a viable solution for the device, what type of cryptographic algorithms should be used, etc.

8.2 Future work

The Internet of Things is a relatively new concept in terms of optimized protocols and security, and thus there is a lot of work for the future. The most pressing issue is simplifying the use of security in IoT for developers without thorough knowledge of IT security. Designing and implementing security in protocols that is simple for developers to use is a must for the future of IoT.

Speed and cryptographic strength is especially important in the Internet of Things. As devices in the Internet of Things are constrained devices, efficient implementations of cryptographic algorithms is especially important to keep the cryptographic strength at an acceptable level.

The Internet of Things is an ever-changing area that will continue to change in the future. While the recommendations in this thesis are made with assumptions for the future of IoT, and encompassing many different solutions, one will have to re-examine the recommendations with large changes in the market. But as many cryptographic properties will always be existing and important, most of the recommendations will be the same for all foreseeable future.

References

[1] Allgemeiner Deutscher Automobil-Club (ADAC). Sicherheitslücken bei bmw connected drive. January 2015. Available at http://www.adac.de/sicherheitsluecken.

[AFK11] P. Karlton A. Freier and P. Kocher. The secure sockets layer (ssl) protocol version 3.0. *RFC6101*, 2011.

[Age09] National Security Agency. The case for elliptic curve cryptography. January 2009. Available at https://www.nsa.gov/business/programs/elliptic_curve.shtml.

[Ama15] Amazon. Dash button. August 2015. Available at https://www.amazon.com/oc/dash-button.

[5] Carna Botnet (anonymous). Internet census 2012. October 2012. Available at http://internetcensus2012.bitbucket.org/paper.html.

[Ash09] Kevin Ashton. That 'internet of things' thing. June 2009. Available at http://www.rfidjournal.com/articles/pdf?4986.

[Aug] August. Smart lock. Available at http://august.com.

[BC14] Daniela Miao Michael Specter Britt Cyr, Webb Horn. Security analysis of wearable fitness devices (fitbit). September 2014. Available at https://courses.csail.mit.edu/6.857/2014/files/17-cyrbritt-webbhorn-specter-dmiao-hacking-fitbit.pdf.

[BI14] Paul Brody and Veena Pureswaran (IBM). Device democracy. September 2014. Available at http://www-935.ibm.com/services/us/gbs/thoughtleadership/internetofthings/.

[CMYP09] Xiangqian Chen, Kia Makki, Kang Yen, and Niki Pissinou. Sensor network security: a survey. *Communications Surveys & Tutorials, IEEE*, 11(2):52–73, 2009.

[Com13] Summit Data Communications. Ble overview. February 2013. Available at http://www.summitdata.com/blog/ble-overview/.

[Com15] US Federal Trade Commission. Internet of things - privacy & security in a connected world. January 2015. Available at http://www.ftc.gov/system/files/documents/reports/

federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf.

[cr.15] cr.yp.to. Caesar: Competition for authenticated encryption: Security, applicability, and robustness. May 2015. Available at http://competitions.cr.yp.to/caesar.html.

[DA99] T. Dierks and C. Allen. The tls protocol version 1.0. *RFC2246*, 1999.

[Dat14] Josh Datko. Crypto for makers - projects for the beaglebone, pi and avrs. July 2014. Available at https://jbdatko.files.wordpress.com/2014/07/cryptomakers_nopause.pdf.

[Die14] Tim Dierks. Security standards and name changes in the browser wars. May 2014. Available at http://tim.dierks.org/2014/05/security-standards-and-name-changes-in.html.

[DJB13] Tanja Lange Daniel J. Bernstein. Security dangers of the nist curves. May 2013. Available at http://www.hyperelliptic.org/tanja/vortraege/20130531.pdf.

[DMT+11] Adam Dunkels, Luca Mottola, Nicolas Tsiftes, Fredrik Österlind, Joakim Eriksson, and Niclas Finne. The announcement layer: Beacon coordination for the sensornet stack. In *Wireless sensor networks*, pages 211–226. Springer, 2011.

[ea] Jacob Appelbaum et al. Cryptech. Available at https://cryptech.is.

[EJN11] J. Loughney E. Jankiewicz and T. Narten. Ipv6 node requirements. *RFC6434*, 2011.

[EO13] IPSO ETSI and OMA. Coap#3 and oma lwm2m plugtests. 2013. Available at https://portal.etsi.org/CTI/Downloads/TestReports/CoAP3_LWM2M_Plugtest_TestReport_v1.0.pdf.

[Fit14] FitBit. Integrate with ios 8 health app. October 2014. Available at https://community.fitbit.com/t5/Feature-Requests/Integrate-with-iOS-8-Health-App/idi-p/319432.

[For14a] Fortinet. Internet of things: Connected home - survey results. June 2014. Available at http://www.fortinet.com/press_releases/2014/internet-of-things.html.

[For14b] Broadband Forum. Tr-069 gears up for ten more years as device management evolves to services management. June 2014. Available at http://www.broadband-forum.org/news/download/pressreleeases/2014/BBF_Q2meeting_TR-069_FINAL_25June2014.pdf.

[GOP12] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[Gro14a] Bluetooth Special Interest Group. Bluetooth core specifications. December 2014. Available at https://www.bluetooth.org/en-us/specification/adopted-specifications.

[Gro14b] Bluetooth Special Interest Group. Bluetooth le security developer documentation. December 2014. Available at https://developer.bluetooth.org/TechnologyOverview/Pages/LE-Security.aspx.

[Gua12] Portal Guard. Quantifying the costs of hacking. May 2012. Available at http://www.portalguard.com/costs_of_hacking.html.

[HB12] K Hartke and O Bergmann. Datagram transport layer security in constrained environments, 2012. Available at http://www.ietf.org/proceedings/83/slides/slides-83-lwig-2.pdf.

[HP14] Hewlett-Packard. Internet of things research study. July 2014. Available at http://h30499.www3.hp.com/hpeb/attachments/hpeb/application-security-fortify-on-demand/189/1/HP_IoT_Research_Study.pdf.

[JHP00] Anil Jain, Lin Hong, and Sharath Pankanti. Biometric identification. *Communications of the ACM*, 43(2):90–98, 2000.

[JN15] Philipp Jovanovic and Samuel Neves. Dumb crypto in smart grids: Practical cryptanalysis of the open smart grid protocol. Cryptology ePrint Archive, Report 2015/428, 2015. http://eprint.iacr.org/.

[JS10] Du Jiang and Chao ShiWei. A study of information security for m2m of iot. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 3, pages V3–576. IEEE, 2010.

[Juc12] Stefan Jucker. *Securing the Constrained Application Protocol*. PhD thesis, 2012.

[KTT+12] Vishal Kumkar, Akhil Tiwari, Pawan Tiwari, Ashish Gupta, and Seema Shrawne. Vulnerabilities of wireless security protocols (wep and wpa2). *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(2):34–38, 2012.

[Kwi] Kwikset. Kevo. Available at http://www.kwikset.com/kevo.

[Lab07] RSA Laboritories. Are elliptic curve cryptosystems patented? September 2007. Available at http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/are-elliptic-curve-cryptosystems-patented.htm.

[Lor11] K Loren. Shade." implementing secure remote firmware updates". *ESC*, 202, 2011.

[Mob15a] Open Mobile Alliance Mobile. Lightweight machine to machine technical specification: Security. February 2015. Available at http://dev_devtoolkit.openmobilealliance.org/IoT/LWM2M10/doc/TS/index.html#!Documents/security.htm.

[Mob15b] Open Mobile Alliance Mobile. Lightweight machine to machine technical specification: Sms channel security. February 2015. Available at http://dev_devtoolkit.openmobilealliance.org/IoT/LWM2M10/doc/TS/index.html#!Documents/smschannelsecurity.htm.

[MQT] MQTT.org. Frequently asked questions, does mqtt support security? Available at http://mqtt.org/faq.

[Oss] Michael Ossmann. Ubertooth one. Available at http://ubertooth.sourceforge.net/hardware/one.

[Peb] Pebble. Smartwatch. Available at https://getpebble.com.

[RALS11] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering*, 37(2):147–159, 2011.

[RPHJ11] Helena Rifa-Pous and Jordi Herrera-Joancomartí. Computational and energy costs of cryptographic algorithms on handheld devices. *Future internet*, 3(1):31–48, 2011.

[Rya13] Mike Ryan. Crackle. February 2013. Available at http://lacklustre.net/projects/crackle/.

[SKZ13] Pablo Martinez-Lozano Sinues, Malcolm Kohler, and Renato Zenobi. Human breath analysis may support the existence of individual metabolic phenotypes. *PloS one*, 8(4):e59909, 2013.

[Sol13] BlackBerry M2M Solutions. White paper: A comparison of protocols for device management and software updates. 2013. Available at http://global.blackberry.com/content/dam/blackBerry/pdf/business/english/telemetry/BlackBerry-M2M-Solutions_White%20Paper_Protocols-for-Device-Management-Software-Updates_A4_June-2013.pdf.

[Son] Sonos. Audio system. Available at http://www.sonos.com/.

[SR13] Bjorn Stelte and Gabi Dreo Rodosek. Thwarting attacks on zigbee-removal of the killerbee stinger. In *2013 9th International Conference on Network and Service Management (CNSM)*, pages 219–226. IEEE, 2013.

[Ste14] Daniel Stenberg. Http2 explained (book). page 14, 2014. Available at http://daniel.haxx.se/http2/.

[Tal15] Shahar Tal. Defcon 22: I hunt tr-069 admins: Pwning isps like a boss. January 2015. Available at https://www.defcon.org/images/defcon-22/dc-22-presentations/Tal/DEFCON-22-Shahar-TaI-I-hunt-TR-069-admins-UPDATED.pdf.

[Vac04] John R Vacca. *Public key infrastructure: building trusted applications and Web services*. CRC Press, 2004.

[vR14] Anne van Rossum. Fitbit data dump. August 2014. Available at http://pastebin.com/9STG0yus.

[vR15] Karl von Randow. Charles web debugging proxy. February 2015. Available at http://www.charlesproxy.com/.

[Wes12] Joakim Wesslen. Home automation – rf protocols. April 2012. Available at http://tech.jolowe.se/home-automation-rf-protocols.

[Wri09] Joshua Wright. Killerbee: Practical zigbee exploitation framework. October 2009. Available at http://www.willhackforsushi.com/presentations/toorcon11-wright.pdf.

[ZSB14] K. Hartke Z. Shelby and C. Bormann. The constrained application protocol. *RFC7252*, 2014.

Appendix A

Hardware, Software and Devices

This appendix lists the hardware and software used for analysis, and a description of the devices referenced in the thesis.

A.1 Hardware Used in Analysis

- Arduino Ethernet
- RF Link 433 MHZ Transmitter
- RF Link 433 MHZ Receiver
- Nexa LYCT-705 Transmitter
- RTL-SDR (RTL2832U)
- Ubertooth One
- Nordic nRF51822 Dev. Dongle
- Generic Bluetooth 4.0 dongle

A.2 Software Used in Analysis

- Wireshark
- Charles Proxy
- GQRX
- BlueZ
- CrackLE

A.3 Devices Referenced in Thesis

The devices listed here are either part of the Analysis, or mentioned elsewhere in the thesis as examples of security concepts.

Device	**Description**	**Link**
EyeFi Mobi	Wi-Fi enabled SD card	http://www.eyefi.com/products/mobi
BMW Connected Drive	In-dash system for automobiles produces by BMW	http://www.bmw.com/com/en/insights/technology/connecteddrive/2013
Fitbit One	Health Tracking Device	https://www.fitbit.com/one
HomeEasy	Protocol specifically made for home automation	http://www.homeeasy.eu
Sonos	Internet-connected speaker system for streaming audio	http://www.sonos.com
Amazon Dash Button	Wi-Fi-button used to order specific products	https://www.amazon.com/oc/dash-button
Supermechanical Twine	A multi-sensor device	http://supermechanical.com/twine
Pebble	Smartwatch	https://getpebble.com

Appendix B

BTLE Advertisement Frames

LE packets received during experiment, showing only the Lower Address Part of the BD_ADDR for anonymity. The datetime variable shows first encounter with the associated address.

number	address	rssi	datetime
1	5c:f9:38:xx:xx:xx	-83	2015-01-28 15:30
2	ff:e8:16:xx:xx:xx	-48	2015-01-28 15:31
3	54:4a:16:xx:xx:xx	-89	2015-01-28 15:39
4	c7:6e:93:xx:xx:xx	-62	2015-01-28 15:53
5	78:f0:bf:xx:xx:xx	-92	2015-01-28 16:05
6	6d:9e:d1:xx:xx:xx	-91	2015-01-28 16:05
7	79:e1:e3:xx:xx:xx	-87	2015-01-28 16:40
8	62:53:b2:xx:xx:xx	-54	2015-01-28 18:01
9	04:88:e2:xx:xx:xx	-93	2015-01-28 19:18
10	79:1b:82:xx:xx:xx	-93	2015-01-28 19:36
11	ee:90:8b:xx:xx:xx	-91	2015-01-28 19:51
12	72:88:e4:xx:xx:xx	-92	2015-01-29 08:17
13	43:50:1e:xx:xx:xx	-91	2015-01-29 08:21
14	36:84:62:xx:xx:xx	-97	2015-01-29 08:59
15	f3:c8:f0:xx:xx:xx	-90	2015-01-29 09:11
16	00:24:e4:xx:xx:xx	-92	2015-01-29 09:26
17	5e:bd:3a:xx:xx:xx	-85	2015-01-29 10:01
18	55:44:5b:xx:xx:xx	-90	2015-01-29 10:03
19	64:8c:6a:xx:xx:xx	-94	2015-01-29 10:06
20	34:be:00:xx:xx:xx	-93	2015-01-29 10:08

21	04:88:e2:xx:xx:xx	-90	2015-01-29 10:17
22	7e:ab:d6:xx:xx:xx	-90	2015-01-29 10:30
23	d3:44:75:xx:xx:xx	-88	2015-01-29 10:44
24	44:ed:01:xx:xx:xx	-92	2015-01-29 11:02
25	7d:b4:3b:xx:xx:xx	-92	2015-01-29 11:06
26	18:19:fa:xx:xx:xx	-96	2015-01-29 11:20
27	53:f2:57:xx:xx:xx	-91	2015-01-29 11:29
28	66:98:50:xx:xx:xx	-51	2015-01-29 11:30
29	40:f9:3f:xx:xx:xx	-46	2015-01-29 11:54
30	4f:89:23:xx:xx:xx	-91	2015-01-29 12:11
31	58:bf:40:xx:xx:xx	-88	2015-01-29 12:14
32	42:b0:ff:xx:xx:xx	-90	2015-01-29 12:27
33	69:52:7a:xx:xx:xx	-91	2015-01-29 12:27
34	4b:39:2f:xx:xx:xx	-91	2015-01-29 12:30
35	6e:fe:53:xx:xx:xx	-86	2015-01-29 12:43
36	4f:ec:9d:xx:xx:xx	-85	2015-01-29 12:57
37	5b:19:39:xx:xx:xx	-91	2015-01-29 13:11
38	47:6c:98:xx:xx:xx	-87	2015-01-29 13:52
39	22:b3:77:xx:xx:xx	-94	2015-01-29 13:59
40	47:bd:1e:xx:xx:xx	-91	2015-01-29 14:00
41	64:eb:b1:xx:xx:xx	-83	2015-01-29 14:00
42	10:c6:fc:xx:xx:xx	-86	2015-01-29 14:01
43	43:fb:68:xx:xx:xx	-91	2015-01-29 14:09
44	56:7c:d8:xx:xx:xx	-39	2015-01-29 14:12
45	3a:f6:74:xx:xx:xx	-57	2015-01-29 14:12
46	49:c3:a6:xx:xx:xx	-80	2015-01-29 14:13
47	59:34:4a:xx:xx:xx	-81	2015-01-29 14:37
48	68:e9:a6:xx:xx:xx	-80	2015-01-29 14:43
49	6e:8e:69:xx:xx:xx	-86	2015-01-29 14:54
50	62:9d:af:xx:xx:xx	-83	2015-01-29 14:59
51	44:13:bf:xx:xx:xx	-86	2015-01-29 15:13
52	64:63:58:xx:xx:xx	-94	2015-01-29 15:25
53	69:85:ca:xx:xx:xx	-89	2015-01-29 15:27
54	73:0b:9a:xx:xx:xx	-94	2015-01-29 15:33
55	49:f2:06:xx:xx:xx	-94	2015-01-29 15:34
56	54:2c:b4:xx:xx:xx	-90	2015-01-29 15:39
57	5f:df:58:xx:xx:xx	-89	2015-01-29 16:07
58	43:41:75:xx:xx:xx	-83	2015-01-29 16:24
59	d3:36:56:xx:xx:xx	-94	2015-01-29 17:02
60	c9:36:89:xx:xx:xx	-91	2015-01-29 17:03

61	69:df:82:xx:xx:xx	-89	2015-01-29 17:04
62	c8:1a:5f:xx:xx:xx	-86	2015-01-29 18:28
63	6e:31:00:xx:xx:xx	-89	2015-01-30 07:53
64	71:eb:25:xx:xx:xx	-91	2015-01-30 08:49
65	50:48:73:xx:xx:xx	-93	2015-01-30 09:13
66	9c:a1:34:xx:xx:xx	-89	2015-01-30 09:13
67	6a:cf:d4:xx:xx:xx	-89	2015-01-30 10:08
68	64:6d:b5:xx:xx:xx	-92	2015-01-30 10:10
69	61:c9:99:xx:xx:xx	-91	2015-01-30 10:14
70	56:b3:f3:xx:xx:xx	-96	2015-01-30 10:15
71	68:2b:06:xx:xx:xx	-87	2015-01-30 10:15
72	70:e8:11:xx:xx:xx	-94	2015-01-30 10:29
73	48:e6:eb:xx:xx:xx	-87	2015-01-30 10:37
74	55:bb:64:xx:xx:xx	-53	2015-01-30 10:41
75	61:26:21:xx:xx:xx	-66	2015-01-30 10:41
76	5c:e2:13:xx:xx:xx	-90	2015-01-30 10:58
77	5e:84:cd:xx:xx:xx	-86	2015-01-30 11:03
78	73:ae:e6:xx:xx:xx	-94	2015-01-30 11:07
79	42:23:d3:xx:xx:xx	-95	2015-01-30 11:10
80	55:f4:41:xx:xx:xx	-92	2015-01-30 11:12
81	53:b4:06:xx:xx:xx	-94	2015-01-30 11:14
82	64:b1:48:xx:xx:xx	-96	2015-01-30 11:17
83	4a:91:34:xx:xx:xx	-92	2015-01-30 11:29
84	5c:1c:e2:xx:xx:xx	-90	2015-01-30 11:30
85	42:29:b1:xx:xx:xx	-94	2015-01-30 11:34
86	65:af:2d:xx:xx:xx	-91	2015-01-30 11:38
87	40:ed:8b:xx:xx:xx	-90	2015-01-30 11:39
88	4e:b9:aa:xx:xx:xx	-47	2015-01-30 11:53
89	70:97:b7:xx:xx:xx	-86	2015-01-30 11:58
90	62:6a:7a:xx:xx:xx	-91	2015-01-30 12:11
91	60:8f:3c:xx:xx:xx	-94	2015-01-30 12:17
92	55:9e:14:xx:xx:xx	-91	2015-01-30 12:29
93	63:7b:04:xx:xx:xx	-92	2015-01-30 12:30
94	49:d5:e3:xx:xx:xx	-93	2015-01-30 13:21
95	7b:1c:6a:xx:xx:xx	-90	2015-01-30 14:02
96	60:67:13:xx:xx:xx	-86	2015-01-30 14:23
97	64:2d:be:xx:xx:xx	-93	2015-01-30 15:27
98	00:22:d0:xx:xx:xx	-78	2015-01-30 15:51
99	46:7c:d6:xx:xx:xx	-93	2015-01-30 16:33
100	7a:b0:a1:xx:xx:xx	-92	2015-01-30 16:39

101	55:0b:04:xx:xx:xx	-84	2015-01-30 16:56
102	65:eb:00:xx:xx:xx	-91	2015-01-30 17:42
103	4e:fc:32:xx:xx:xx	-92	2015-01-30 20:50
104	d6:3c:34:xx:xx:xx	-87	2015-01-30 20:50
105	41:2f:fc:xx:xx:xx	-87	2015-01-31 11:56
106	45:95:4f:xx:xx:xx	-87	2015-01-31 13:40
107	59:23:69:xx:xx:xx	-90	2015-01-31 15:25
108	4d:c4:e1:xx:xx:xx	-94	2015-01-31 18:05
109	7a:fd:7a:xx:xx:xx	-95	2015-02-01 10:54
110	5a:ff:46:xx:xx:xx	-85	2015-02-01 20:30
111	54:bf:6b:xx:xx:xx	-94	2015-02-01 22:07
112	55:29:8f:xx:xx:xx	-90	2015-02-02 07:44
113	5b:c1:0c:xx:xx:xx	-87	2015-02-02 08:00
114	4f:13:8d:xx:xx:xx	-93	2015-02-02 08:04
115	4f:07:20:xx:xx:xx	-93	2015-02-02 08:10
116	4a:c2:22:xx:xx:xx	-88	2015-02-02 08:19
117	78:a4:19:xx:xx:xx	-94	2015-02-02 08:45
118	6c:06:2b:xx:xx:xx	-93	2015-02-02 09:01
119	56:87:5b:xx:xx:xx	-94	2015-02-02 09:19
120	46:f1:72:xx:xx:xx	-95	2015-02-02 09:34
121	7c:32:79:xx:xx:xx	-92	2015-02-02 09:34
122	43:46:be:xx:xx:xx	-94	2015-02-02 09:41
123	51:8a:d0:xx:xx:xx	-93	2015-02-02 09:47
124	4c:1a:c2:xx:xx:xx	-78	2015-02-02 09:56
125	59:ab:e5:xx:xx:xx	-91	2015-02-02 10:07
126	69:be:b2:xx:xx:xx	-94	2015-02-02 10:11
127	42:9f:1c:xx:xx:xx	-93	2015-02-02 10:13
128	2b:76:b2:xx:xx:xx	-90	2015-02-02 10:16
129	46:de:51:xx:xx:xx	-94	2015-02-02 10:21
130	79:4b:91:xx:xx:xx	-91	2015-02-02 10:29
131	4e:10:4d:xx:xx:xx	-93	2015-02-02 10:31
132	03:72:f9:xx:xx:xx	-86	2015-02-02 10:32
133	53:62:e7:xx:xx:xx	-92	2015-02-02 10:33
134	56:35:dd:xx:xx:xx	-83	2015-02-02 10:40
135	4e:df:48:xx:xx:xx	-87	2015-02-02 10:40
136	68:18:e1:xx:xx:xx	-89	2015-02-02 10:46
137	6c:0e:e9:xx:xx:xx	-88	2015-02-02 10:52
138	2a:0d:06:xx:xx:xx	-85	2015-02-02 11:34
139	7d:84:36:xx:xx:xx	-91	2015-02-02 11:51
140	66:5b:c1:xx:xx:xx	-89	2015-02-02 11:51

141	6f:7e:0f:xx:xx:xx	-93	2015-02-02 11:51
142	4b:5f:3a:xx:xx:xx	-83	2015-02-02 12:03
143	66:aa:59:xx:xx:xx	-94	2015-02-02 12:03
144	5d:b7:7d:xx:xx:xx	-91	2015-02-02 12:03
145	64:83:1f:xx:xx:xx	-96	2015-02-02 12:09
146	51:df:a1:xx:xx:xx	-92	2015-02-02 12:37
147	71:bd:f0:xx:xx:xx	-94	2015-02-02 12:42
148	7f:13:d7:xx:xx:xx	-89	2015-02-02 13:03
149	5f:79:72:xx:xx:xx	-89	2015-02-02 13:06
150	55:cf:77:xx:xx:xx	-94	2015-02-02 13:15
151	77:7d:c5:xx:xx:xx	-94	2015-02-02 13:24
152	50:60:a5:xx:xx:xx	-94	2015-02-02 13:47
153	49:18:dc:xx:xx:xx	-88	2015-02-02 14:08
154	a8:88:08:xx:xx:xx	-87	2015-02-02 14:39
155	4f:aa:19:xx:xx:xx	-87	2015-02-02 14:39
156	7d:d9:93:xx:xx:xx	-79	2015-02-02 14:54
157	7e:e5:0c:xx:xx:xx	-93	2015-02-02 15:02
158	4d:99:c8:xx:xx:xx	-94	2015-02-02 15:02
159	6c:ed:b9:xx:xx:xx	-88	2015-02-02 15:05
160	5a:61:95:xx:xx:xx	-92	2015-02-02 15:14
161	0c:8c:dc:xx:xx:xx	-88	2015-02-02 15:34
162	f4:6a:bc:xx:xx:xx	-92	2015-02-02 15:41
163	6c:55:69:xx:xx:xx	-91	2015-02-02 15:43
164	04:88:e2:xx:xx:xx	-92	2015-02-02 15:53
165	00:22:d0:xx:xx:xx	-92	2015-02-02 15:56
166	d0:ff:50:xx:xx:xx	-93	2015-02-02 16:18
167	65:53:02:xx:xx:xx	-95	2015-02-02 16:22
168	55:64:3e:xx:xx:xx	-94	2015-02-02 16:27
169	65:4e:b7:xx:xx:xx	-80	2015-02-02 16:32
170	44:5c:51:xx:xx:xx	-94	2015-02-02 16:58
171	04:88:e2:xx:xx:xx	-85	2015-02-02 17:46
172	77:97:2e:xx:xx:xx	-94	2015-02-02 18:27
173	7c:5f:1d:xx:xx:xx	-92	2015-02-02 19:16
174	78:40:17:xx:xx:xx	-80	2015-02-02 20:42
175	65:6a:70:xx:xx:xx	-92	2015-02-02 21:54
176	df:e5:40:xx:xx:xx	-92	2015-02-03 07:53
177	72:61:3b:xx:xx:xx	-94	2015-02-03 08:01
178	5b:1e:e3:xx:xx:xx	-90	2015-02-03 08:19
179	62:a8:42:xx:xx:xx	-94	2015-02-03 09:15
180	49:f2:bc:xx:xx:xx	-80	2015-02-03 09:18

181	7b:a4:ec:xx:xx:xx	-86	2015-02-03 09:42
182	7d:fa:40:xx:xx:xx	-87	2015-02-03 09:42
183	6b:fb:be:xx:xx:xx	-91	2015-02-03 10:17
184	ff:e8:13:xx:xx:xx	-91	2015-02-03 10:25
185	7d:ad:8c:xx:xx:xx	-89	2015-02-03 10:58
186	71:e3:01:xx:xx:xx	-93	2015-02-03 11:01
187	69:f8:e6:xx:xx:xx	-92	2015-02-03 11:46
188	43:9e:80:xx:xx:xx	-95	2015-02-03 12:11
189	6e:95:eb:xx:xx:xx	-94	2015-02-03 12:16
190	79:94:b2:xx:xx:xx	-94	2015-02-03 12:22
191	76:df:c3:xx:xx:xx	-88	2015-02-03 12:22
192	4a:08:0b:xx:xx:xx	-95	2015-02-03 12:46
193	71:7d:a3:xx:xx:xx	-92	2015-02-03 12:55
194	5d:06:da:xx:xx:xx	-94	2015-02-03 13:10
195	62:fe:81:xx:xx:xx	-94	2015-02-03 13:26
196	41:2b:d7:xx:xx:xx	-91	2015-02-03 14:03
197	5c:ac:6c:xx:xx:xx	-93	2015-02-03 14:05
198	73:93:ad:xx:xx:xx	-93	2015-02-03 14:09
199	4a:d8:02:xx:xx:xx	-91	2015-02-03 14:50
200	1c:2f:79:xx:xx:xx	-95	2015-02-03 15:11
201	6c:e5:51:xx:xx:xx	-92	2015-02-03 15:21
202	4c:5e:38:xx:xx:xx	-92	2015-02-03 16:06
203	69:f8:88:xx:xx:xx	-94	2015-02-03 16:16
204	4c:c5:60:xx:xx:xx	-93	2015-02-03 16:19
205	4f:e1:81:xx:xx:xx	-91	2015-02-03 16:32
206	6a:39:0c:xx:xx:xx	-89	2015-02-03 17:03
207	71:42:d6:xx:xx:xx	-92	2015-02-03 17:21
208	75:a1:3f:xx:xx:xx	-85	2015-02-03 18:00
209	72:3c:00:xx:xx:xx	-90	2015-02-03 18:29
210	73:81:f4:xx:xx:xx	-91	2015-02-03 18:49
211	53:6a:97:xx:xx:xx	-91	2015-02-03 19:00
212	6a:7b:ff:xx:xx:xx	-92	2015-02-03 19:11
213	6e:ef:99:xx:xx:xx	-92	2015-02-03 19:38
214	59:5b:66:xx:xx:xx	-88	2015-02-03 22:06
215	76:9e:c7:xx:xx:xx	-87	2015-02-04 08:16
216	4b:41:3b:xx:xx:xx	-92	2015-02-04 08:18
217	6a:45:f3:xx:xx:xx	-90	2015-02-04 08:26
218	45:13:8e:xx:xx:xx	-94	2015-02-04 08:30
219	79:9e:0f:xx:xx:xx	-82	2015-02-04 08:37
220	d0:ff:50:xx:xx:xx	-91	2015-02-04 09:07

221	6f:a3:fb:xx:xx:xx	-88	2015-02-04 09:13
222	40:b0:5a:xx:xx:xx	-93	2015-02-04 09:27
223	c0:c8:9c:xx:xx:xx	-92	2015-02-04 09:32
224	c2:cc:cf:xx:xx:xx	-92	2015-02-04 09:44
225	51:6c:de:xx:xx:xx	-88	2015-02-04 09:45
226	d0:1a:6a:xx:xx:xx	-84	2015-02-04 09:58
227	50:bd:5a:xx:xx:xx	-95	2015-02-04 10:00
228	58:78:d6:xx:xx:xx	-93	2015-02-04 10:44
229	58:51:63:xx:xx:xx	-94	2015-02-04 10:53
230	4e:d7:09:xx:xx:xx	-93	2015-02-04 11:17
231	6a:54:97:xx:xx:xx	-95	2015-02-04 11:28
232	58:7f:ac:xx:xx:xx	-93	2015-02-04 11:52
233	4d:a4:53:xx:xx:xx	-91	2015-02-04 12:14
234	e3:ae:81:xx:xx:xx	-92	2015-02-04 12:16
235	49:19:60:xx:xx:xx	-92	2015-02-04 12:17
236	60:70:f1:xx:xx:xx	-95	2015-02-04 12:19
237	47:90:dc:xx:xx:xx	-92	2015-02-04 12:25
238	59:43:d1:xx:xx:xx	-90	2015-02-04 12:58
239	51:0e:dc:xx:xx:xx	-93	2015-02-04 13:31
240	78:55:e4:xx:xx:xx	-92	2015-02-04 13:42
241	53:5f:92:xx:xx:xx	-90	2015-02-04 13:47
242	46:42:fe:xx:xx:xx	-92	2015-02-04 13:57
243	7b:91:35:xx:xx:xx	-89	2015-02-04 13:59
244	65:f9:82:xx:xx:xx	-93	2015-02-04 14:03
245	4f:1a:bd:xx:xx:xx	-87	2015-02-04 14:03
246	71:94:cc:xx:xx:xx	-92	2015-02-04 14:03
247	5a:20:c1:xx:xx:xx	-94	2015-02-04 14:34
248	7a:9c:ab:xx:xx:xx	-92	2015-02-04 16:04
249	3b:cc:f6:xx:xx:xx	-93	2015-02-04 16:10
250	64:0b:1c:xx:xx:xx	-92	2015-02-04 16:10
251	74:96:41:xx:xx:xx	-88	2015-02-04 16:23
252	66:d3:5d:xx:xx:xx	-73	2015-02-04 16:56
253	5f:db:6a:xx:xx:xx	-52	2015-02-04 17:16
254	42:a2:a3:xx:xx:xx	-43	2015-02-04 18:04
255	66:8c:53:xx:xx:xx	-91	2015-02-04 18:58
256	59:38:4a:xx:xx:xx	-91	2015-02-04 19:17
257	4e:60:73:xx:xx:xx	-39	2015-02-04 19:23
258	4b:d0:d1:xx:xx:xx	-42	2015-02-04 19:42
259	60:1a:2d:xx:xx:xx	-90	2015-02-04 20:04
260	5b:54:1f:xx:xx:xx	-84	2015-02-04 20:34

261	6e:50:dd:xx:xx:xx	-86	2015-02-05 08:38
262	77:72:30:xx:xx:xx	-89	2015-02-05 08:55
263	7e:ea:1a:xx:xx:xx	-91	2015-02-05 09:29
264	15:0b:24:xx:xx:xx	-89	2015-02-05 10:08
265	76:68:10:xx:xx:xx	-52	2015-02-05 10:44
266	69:cf:d6:xx:xx:xx	-90	2015-02-05 11:20
267	65:a2:6e:xx:xx:xx	-89	2015-02-05 11:33
268	57:47:ed:xx:xx:xx	-82	2015-02-05 11:41
269	4a:16:f8:xx:xx:xx	-91	2015-02-05 11:59
270	7f:b6:6a:xx:xx:xx	-40	2015-02-05 12:17
271	50:78:96:xx:xx:xx	-46	2015-02-05 12:24
272	47:ab:46:xx:xx:xx	-32	2015-02-05 12:37
273	57:5e:4c:xx:xx:xx	-89	2015-02-05 12:38
274	4e:8c:7e:xx:xx:xx	-92	2015-02-05 12:40
275	6e:62:84:xx:xx:xx	-89	2015-02-05 13:02
276	42:b3:8d:xx:xx:xx	-89	2015-02-05 13:07
277	52:f4:8f:xx:xx:xx	-44	2015-02-05 13:10
278	13:70:9a:xx:xx:xx	-83	2015-02-05 13:10
279	74:19:aa:xx:xx:xx	-86	2015-02-05 13:26
280	61:d5:23:xx:xx:xx	-40	2015-02-05 13:55
281	42:53:da:xx:xx:xx	-92	2015-02-05 14:05
282	42:7f:6c:xx:xx:xx	-89	2015-02-05 14:05
283	4f:5c:e3:xx:xx:xx	-90	2015-02-05 15:26
284	7a:d4:dd:xx:xx:xx	-89	2015-02-05 15:34