OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

FACULTY OF COMPUTER SCIENCE

Communication and Networked Systems

Communication and Networked Systems

Bachelor Thesis

Secure IoT Device Commissioning

Jana Eisoldt

Supervisor: Prof. Dr. rer. nat. Mesut Güneş
Assisting Supervisor: M.Sc. Marian Buschsieweke

Institute for Intelligent Cooperating Systems, Otto-von-Guericke-University Magdeburg

18. Februar 2020

Abstract

Abstract

Internet of Things (IoT) solutions are being applied in many fields such as smart home, healthcare and manufacturing. These networks often handle sensitive data which require a strong level of security. However, the commissioning of new devices could often already be easily exploited by an adversary. A successful attack would not only affect a single device but a whole network. Besides that, the commissioning requires a solution which is not only secure, but also complies with the limitations of IoT devices. Since the devices often lack interfaces like a display, the network credentials need to be transmitted over a wireless interface. A desirable solution is one which allows to use existing hardware to make it applicable on a wide range of devices.

This thesis introduces a new commissioning protocol, which aims at being lightweight, secure and without a need for additional hardware. The majority of IoT devices is supplied with a Light Emitting Diode (LED), which can serve for creating an auxiliary channel to communicate unsecured data. The developed protocol utilizes the LED to transmit a self-generated key. This key is captured by the user's smartphone camera and used for encrypting the network credentials. In that way, the information can be shared with the target device without making them accessible to an adversary.

Experiments on a prototype show that the commissioning protocol has a low memory footprint and can therefore be used in highly constrained environments. Apart from that, it is evaluated that the transfer of the encryption key via light satisfies the demand of a fast setup. Furthermore it is analysed how an attacker may try to get hold of the network credentials or prevent the commissioning. The evaluation shows that the protocol succeeds in providing a secure communication between the two previously unknown parties by making use of existing hardware.

Contents

List of Figures

List of Tables

Acronyms

AES Advanced Encryption Standard. 3, 13, 15, 22

ASK Amplitude-Shift Keying. 13

CBC-MAC Cipher Block Chaining Message Authentication Code. 3, 4

CRC Cyclic Redundancy Check. 11, 14

DDoS Distributed Denial of Service. 1

DoS Denial of Service. 20, 22, 25, 28

FCS Frame Check Sequence. 9, 14, 15

IEEE Institute of Electrical and Electronics Engineers. 22

IoT Internet of Things. iii, 1, 2, 7–9, 11, 12, 14, 19, 20, 22, 24, 27, 28

LED Light Emitting Diode. iii, 2, 3, 9, 11–15, 17, 27, 28

LiFi Light Fidelity. 2, 3, 7, 9, 11, 13–15, 17, 22, 24, 27, 28

LIRA Light channel for device Initialization and Radio channel for Authentication. 7

LTO Link Time Optimisation. 19

MAC Message Authentication Code. 3–5, 11–13, 24

MITM Man in the Middle. 12

NIST National Institute of Standards and Technology. 3, 22

PRNG Pseudo Random Number Generator. 12, 15, 22

PUF Physical Unclonable Function. 12

RAM Random Access Memory. 19, 20, 27

RSSI Received Signal Strength Indicator. 12, 13

SFD Start Frame Delimiter. 11

SRAM Static Random Access Memory. 12

TRNG True Random Number Generator. 12

WiFi Wireless Fidelity. 7, 8

WSN Wireless Sensor Network. 13

Glossary

Counter with CBC-MAC (CCM) Mode of operation for symmetric encryption which includes a MAC value to prove authenticity. 2–4, 11–15, 27

Differential Manchester Code Modification of Manchester Code which defines a signal with the presence or absence of a transition in a bit interval. 13

Hamming Distance Number of different symbols at the same positions of two equal-length strings. 14

Manchester Code Line code which includes the transmission of a clock pulse with each data bit. 11, 13, 14, 27

On-Off Keying An amplitude-shift keying modulation technique which is characterized by either the presence or absence of a signal. 13

Out-of-Band Communication which is not taking place on the main channel. 2, 3, 7, 27, 28

RIOT-OS Open-source operating system for the IoT. 14, 15, 19

SRAM PUF A function which produces unpredictable and unique results based on the state of the SRAM. 12

CHAPTER 1

Introduction

The Internet of Things (IoT) is a network of physical objects which sense, collect and process data of the environment and are able to interact with it [1]. The IoT has enabled new technologies in the last years in fields like smart home, healthcare or manufacturing. It is expected that the number of connected devices could rise to 75.44 billion by 2025, which would be five times higher than ten years before [2]. The use of a network made up of smart objects for example allow a user to remotely control the temperature, light or washing machine in the form of smart homes.

IoT devices are often characterized by high memory restrictions, low computational and low battery power. These are limitations which are normally not faced in traditional computing. For this reason new standards need to be developed to make a similar behaviour to the traditional internet possible. One aspect that is often neglected in the development of new applications is security, although sensitive data is permanently collected. Attacks like Mirai in 2016 [3] have shown that flaws in IoT security can be exploited to cause a Distributed Denial of Service (DDoS) attack on a large scale. In that case, 400 000 IoT devices were infected to form a huge botnet. An attacker could also use security flaws for gaining physical access to a house or influence devices to cause a flooding or fire.

This thesis aims at providing a new commissioning protocol for IoT devices to enhance the security already in the initialization process of a new device. It furthermore shows the feasibility of the specification with an implementation of a prototype on a bluepill board [4].

1.1 Motivation

Device commissioning is the process of transferring data to an entity so that it can participate in a secure network [5]. A user who is setting up a smart home network may wish to integrate temperature sensors in a network to make a remote control and energy efficient heating possible. Since the sensors will not be equipped with an interface like a display, the question arises of how to share the network credentials without making them accessible to an adversary as well. If intruders are able to get hold of the network credentials, they would not only be able to influence the heating but also control other devices or access private data. Sending this data unencrypted to the target device via a wireless interface is not desirable, because this channel could easily be eavesdropped. The solution might be a pre-shared key, but this could be exploited or influenced by an adversary already during the manufacturing process or during delivery to the customer.

A solution for this issue should have a focus on high security and be able to be used with a wide range of devices. IoT devices are in most cases equipped with a simple Light Emitting Diode (LED). As shown by Duque et al. [6], even low power LEDs can be used for communication over light. Since this signal has only a small range it is possible to share sensitive, unsecured data with another device over this channel.

Additionally, constrained devices are often characterized by low processing power, high memory restrictions and a need for applications with low power consumption [7]. For these reasons, a new commissioning protocol should consider the resource restrictions of the target devices.

Nevertheless, the user experience should also not be neglected. It can be assumed that the user who wants to commission temperature sensors in a network may wish to integrate further devices. For that, it is preferred to use the same bootstrapping method for all devices. Furthermore, a user is interested in a fast and convenient commissioning.

With consideration of the above mentioned restrictions, a new commissioning protocol is introduced in this thesis. It requires the user to guide the bootstrapping of a new device by receiving keying material on a smartphone via Light Fidelity (LiFi) communication. This keying material forms a trust anchor for secure communication with the target device.

1.2 Thesis Structure

This thesis is structured as follows: In the next section, the theoretical background of the protocol is outlined. The following chapter presents related work on other commissioning strategies in the IoT. In Chapter 3 the protocol is described in its details. Furthermore, the implemented prototype is presented. Thereafter a performance and a security analysis are conducted in Chapter 4 to prove the feasibility of the protocol. The last chapter provides a summary of this thesis and investigates future work.

1.3 Theoretical Background

In this section, the theoretical background of the commissioning protocol with a focus on the main security measures is depicted. At first, the security goals of the protocol are defined. Besides that, the technology LiFi is studied for its usage as an Out-of-Band channel. On that follows a description of the encryption standard Counter with CBC-MAC (CCM).

1.3.1 Security Goals

To provide a high level of security, commonly different aspects have to be considered. Especially the security goals of integrity, confidentiality, availability and authentication are important to allow a secure commissioning process. They are defined as the following [8]:

- **Confidentiality**: Only trusted entities can access messages sent via a network.
- **Integrity**: The content of a message cannot be changed without notice.
- **Authentication**: Messages can only be send by trusted entities, which means that the sender can be identified.
- **Availability**: Entities in the network are always able to access data and functionalities of the network.

1.3.2 LiFi as an Out of Band Channel

Roman et al. [9] describes that an Out-of-Band channel is often characterized by limited capabilities such as a small signal range compared to the main channel. Furthermore it often requires human interaction as well. These properties make a secure communication of unencrypted data possible, since the risk of eavesdropping or manipulation is low. Out-of-Band channels can for instance operate over light, over sound or visually.

One possible choice is LiFi, which was introduced by Prof. Harald Haas in a Global Ted Talk in 2011 [10]. The transmitter flickers an optical light source such as an LED to transfer data, which can be received with a photo detector [11].

As pointed out by Roman et al. [9] the limited range of a LiFi signal makes it a feasible choice for Out-of-Band communication. The range of the signal is restricted by walls, for that reason an adversary would need to be physically close. If human interaction is part of the communication as well, the security can be improved even further since the user can choose an adequate environment and the time of activation. If a distracting signal is present it would probably not go without notice. These properties make LiFi suitable for exchanging sensitive data between previously unknown parties.

1.3.3 Counter with CBC-MAC (CCM)

The following descriptions are done according to the recommendation on CCM by the National Institute of Standards and Technology (NIST) [12]. CCM is an block cipher mode which combines Cipher Block Chaining Message Authentication Code (CBC-MAC) with the Counter mode of encryption. It allows authenticated encryption, therefore it provides authentication and integrity by adding a Message Authentication Code (MAC) value and confidentiality by encrypting the message. The algorithm allows to include additional data in the MAC computation which is not being encrypted. It uses symmetric block encryption such as Advanced Encryption Standard (AES).

The computation consists of the processes generation-encryption and decryption-verification. In generation-encryption the MAC value is calculated first by using the CBC-MAC algorithm. The result is encrypted in the next step, together with the plaintext message. In decryption-verification the communication partner calculates the ciphertext and the according MAC value. The result of the MAC calculation is compared to the received value.

Currently only block sizes of 128 bit are supported in CCM. The algorithm requires a payload P, a cryptographic key K and a nonce N as input. It is important to use a nonce only once in conjunction with a given key. Reusing a nonce would allow an adversary to gain knowledge of the plaintext messages and therefore break its confidentiality. Besides that, other parameters need to be set. These are the bit length of the authentication field T_{len}, or t as the same value in octets. Furthermore q as the maximum size of the payload's binary length is required. The value for t can be any even value between 4 and 16 octets. A long MAC length will allow higher security, but on the other hand also lead to longer messages. The value of q can be in the range from 2 to 8 octets. The length of the nonce N is calculated as $15 - q$. Besides that, the length of the message in bit is stored as P_{len}.

The calculations use the XOR operation, in the following represented as $\oplus$. The function $\text{MSB}_s(X)$ returns the s most significant, or leftmost, bits of X. The $||$ operator is used for concatenation. The workflow of generation-encryption is presented in the following.

At first, the authentication field is calculated. This is done by applying CBC-MAC, which uses the key stream blocks $B_0, \ldots, B_r$ as input. These building blocks are made up of 16 octets. The first block consists of flags, the nonce N and the octet length of the payload P. In case additional data shall be included, the length of it is encoded in a predefined manner and concatenated with the corresponding data. The last block is zero padded if it does not consume the full size. The initialization vector is set to 0. Following on that is the message data, which has to be zero padded as well in case the last block is not entirely filled. The MAC value is calculated using the encryption key K and the key stream blocks $B_0, \ldots, B_r$. It results in the output T of length T_{len}:

$$\begin{aligned} X_1 &:= E\left(K, B_0\right) \\ X_{i+1} &:= E\left(K, X_i \oplus B_i\right), \text{ for } i = 1, \ldots, r \\ T &:= \text{MSB}_{T_{\text{len}}}(X_r) \end{aligned}$$

The algorithm starts by encrypting the first block B_0 with the given key. Next, the result is XOR-ed with the following block B_1 and encrypted. This is continued for all blocks, with the last output representing the MAC value. The first T_{len} bits of the result will be concatenated to the plaintext for encryption.

In the encryption phase the same key K as for the MAC calculation is used. For that reason it has to be made sure that the building blocks differ from the initialization vector B_0 in the CBC-MAC calculation. The ciphertext is the result of the calculation presented below, where P is the plaintext message with bit length P_{len} and T the result of the above described MAC calculation.

$$\begin{aligned} S_i &:= E\left(K, A_i\right), \text{ for } i = 1, \ldots, r \\ S &:= S_1 || \ldots || S_i \\ C &:= \left(P \oplus \text{MSB}_{P_{\text{len}}}\left(S\right)\right) || \left(T \oplus \text{MSB}_{T_{\text{len}}}\left(S_0\right)\right) \end{aligned}$$

The building blocks A_i contain flags, nonce N and counter i, which is incremented accordingly. The result of this is XOR-ed with the message plaintext P. However, the first building block S_0 is not used for the encryption of plaintext. Instead it is XOR-ed with the previously calculated MAC value and the result appended.

The decryption-verification algorithm behaves similar to the previously described computation. For the calculation the encrypted message C, nonce N, additional authenticated data A and the key K are required.

At first C is decrypted and in the second step the MAC value T is calculated. The decrypted value is calculated by using the same keystream as in encryption and XOR-ing it with the received ciphertext C. Following on that the MAC value is computed in the same manner as in the generation-encryption phase. If T and the computed value are equal, the decrypted message is returned. If not, the algorithm will return `INVALID`.

CHAPTER 2

Related Work

The topic of secure commissioning of IoT devices is not completely new. Several different approaches exist on how to securely commission IoT devices. The most common approach is to use an Out-of-Band channel in combination with a radio channel. However, the usage of an auxiliary channel often comes with additional hardware requirements like a photo sensor or microphone.

Kovačević et al. [13] developed the Light channel for device Initialization and Radio channel for Authentication (LIRA) protocol and a modified version LIRA+. The approach has the disadvantage that IoT devices need to be capable to capture signals over the light. The protocol makes it possible to commission several devices at the same time by placing them on a flashing screen. This screen transmits a secret key to every single device. One out of these devices will act as the group coordinator to verify the received keys, which is realised with a master key. This master key serves as a parameter for generating the other keys. For that reason the group coordinator is able to verify if the devices received the correct keys. While the initial transmission of the keys is done by using LiFi, the communication between the devices takes place on a public radio channel.

Another approach is to use an audio channel for communication, which is used by Soriente et al. [14]. The commissioning takes place in two steps. In the first phase the IoT device transmits cryptographic material via an audio channel. The receiving device uses this key to encrypt other information and sends it back via the audio channel. The verification is done by the user, who needs to compare audio sequences produced by both devices. This can be seen as a drawback, since it cannot be assumed that the user is always capable to commit this task correctly.

Kuo et al. [15] do not rely on the use of a Out-of-Band channel. Instead, the target device is placed in a Faraday Cage, together with a keying device. The Faraday Cage is meant to shield the communication between the devices. However, a disadvantage is the need for additional hardware.

Commercial solutions for the issue of secure commissioning exist as well. The company Electric Imp introduced the BlinkUp application [16]. BlinkUp has the purpose of connecting a new device to the Wireless Fidelity (WiFi) network and to add it to the user's account. This is done by transmitting data from a mobile phone to the device with a flashing screen.

A similar way of commissioning was developed as BlinkComm by Perkovi et al. [17] which uses a differential coding scheme to make the transmission faster. This needs a photodiode on the IoT device to receive the signal from the smartphone. In BlinkComm no logic was used for error detection and correction to keep power and memory usage as low as possible.

Amazons Dash Button [18] is another commercial solution which uses ultrasonic sound to allow a secure connection to the local WiFi network. Similarly to the BlinkUp application the user needs to provide the WiFi credentials to a smartphone application. The smartphone then delivers the credentials to the Dash Button via ultrasonic sound.

CHAPTER 3

Thesis Contribution

The goal of this thesis is the development of a commissioning protocol and furthermore the implementation of a prototype. The protocol aims at providing a secure commissioning standard which does fulfil the requirements of a constrained environment as described in Chapter 1.1. For that reason, the communication is initially done via LiFi to transfer an encryption key, which is then used to securely send the network credentials via wireless communication back to the target device. This process includes user interaction as well.

In this chapter, the commissioning protocol is described in detail, with an explanation of the design choices afterwards.

3.1 Specification of the Commissioning Protocol

The commissioning process of an IoT device following the specification of this thesis requires user interaction and a smartphone with the corresponding application installed. It is required that the smartphone receives signals over light. For that reason it needs to have an integrated camera, which is available on customary devices. During the commissioning process the smartphone's camera needs to be placed in front of the target device's LED to capture the encryption key. In case of failure, for instance because the encryption key is not correctly received, the application should inform the user to start over again.

The commissioning is launched by turning the target device on. The protocol flow is shown in Figure 3.1. The IoT device first gathers an entropy value for generating a cryptographically secure key. A new key is generated each time the power button is pressed.

After successful key generation the message is created. It consists of a 1 B header, the payload and a 9 bit trailer as depicted in Figure 3.2. The header includes information about the version number and the length of the message in byte. The version number is important for choosing the right encryption parameters in case the protocol is modified in the future. The first bytes of the payload comprise a unique identifier of the target device for addressing it. The size of it depends on the protocol standard used by the wireless interface. The rest of the payload consists of the commissioning key. The trailer contains a Frame

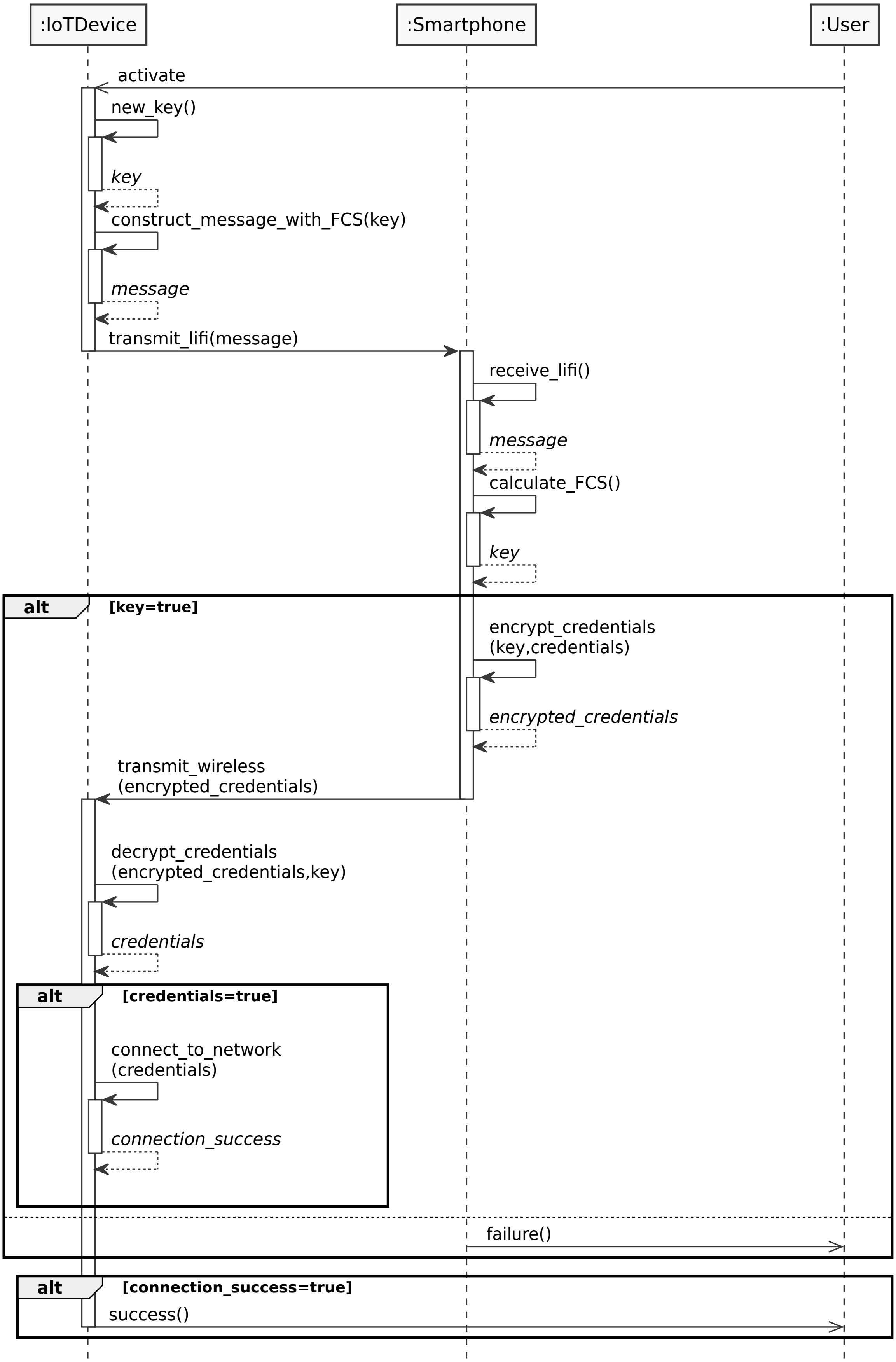

Figure 3.1: Sequence Diagram of the Protocol

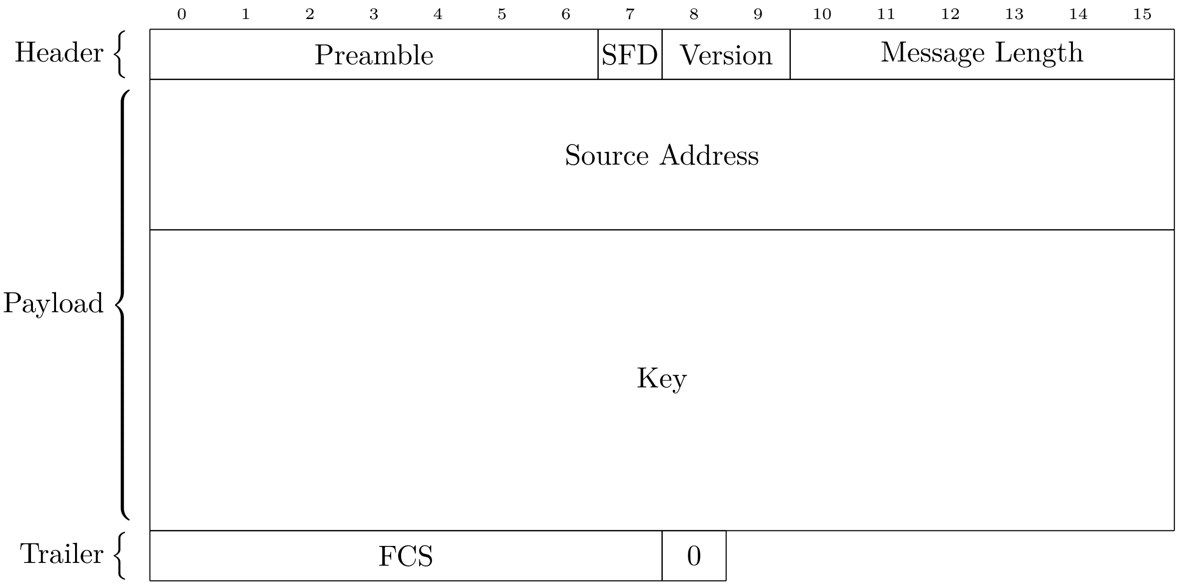

Figure 3.2: Package Structure of the Message from IoT Device to Smartphone

Check Sequence (FCS) to allow error detection on the smartphone, calculated with a 8 bit Cyclic Redundancy Check (CRC). The generator polynomial is $x^8 + x^7 + x^6 + x^3 + x^2 + x + 1$ with an initial seed of `0xff`. The choice of parameters is discussed in Chapter 3.1.4.

The transmission of the encryption key data is done by using LiFi. The data is sent via the status LED of the IoT device. The signal is captured with the smartphone's camera. For a correct transmission the smartphone should be placed as close to the target device as possible. The communication starts with a preamble build up of the 7 bit pattern `1111 111` and a Start Frame Delimiter (SFD), represented as `0`. This first byte is applied for clock synchronization. Afterwards, the message is transmitted using Manchester Code. The trailer finishes on a single `0` to turn the LED off. After that, the IoT pauses for a one bit period to signal the end of transmission. Since it is not unlikely that a user misses the transmission of the first bits, the IoT device retransmits the message one time.

After a successful reception of the LiFi message the smartphone runs the same CRC algorithm over the message as the IoT device previously. If this calculation results in a remainder different to 0 an error has occurred during transmission. In this case the smartphone prompts the user to restart the commissioning.

Alternatively if no error is detected, the smartphone uses the received commissioning key for encrypting the network credentials in CCM mode. Additionally to the password, the MAC value includes the message's header to obtain integrity for it as well. The header is made up of the following data:

- `uint8_t version`: version number of the protocol, which has to be equal to the version number received from the IoT device.
- `size_t pswd_len`: specifies the length of the plaintext password.
- `uint8_t nonce[nonce_len]`: the nonce used for encryption of the length as specified in the used protocol version.

- `uint8_t encrypted_msg[pswd_len+mac_len]`: the ciphertext which consists of the encrypted password and the MAC value.

This response is sent to the IoT device via its wireless communication interface. The standard used for this wireless communication depends on the interface provided by the target device. The IoT device uses the received information to decrypt the password with CCM. That will prove the authenticity of the message as well. Only if the decryption was successful and the authenticity was validated, the IoT device tries to connect to the local network by using the received credentials. If a connection is established, the target device turns the status LED on to notify the user about a successful commissioning. Otherwise the commissioning process needs to be started again.

3.1.1 Generation of a Cryptographically Secure Key

The first step of the protocol is the collection of an entropy value for generating the encryption key. A strong random number is crucial, otherwise an adversary could guess the generated value [19] and use it for decrypting the network credentials later on. Furthermore it is important to generate a new key every time the device is switched on. If this is not done a Man in the Middle (MITM) attack could take place while the IoT device is delivered to the customer. The adversary would simply start the device himself, collect the key and use it for decryption when the commissioning takes place. The detailed process of key generation is highly dependent on the target device's features. The implementation of generating a cryptographically secure key is not in the scope of this bachelor thesis, but the following aspects have to be considered.

As described by Schneider et al. [19], random numbers can be generated with a deterministic algorithm, called Pseudo Random Number Generator (PRNG), which requires a seed value. A PRNG used in a security related context has to be cryptographically strong, which means that an adversary must not be able to predict a future random number after accessing one result. An alternative is a True Random Number Generator (TRNG), which completely relies on physical phenomena such as thermal noise or radioactive decay [20]. That leads to unpredictable results if an adversary cannot extract the value as well or influence it. However, as analysed by Schneider et al. [19] a TRNG can have several drawbacks such as data availability and a limited amount of entropy. As a solution, a single value taken from a physical process can be used as a seed value for a PRNG. In case the entropy source does not deliver a high magnitude, care should be taken to either apply an amplifier or combine several entropy sources. Besides that, the entropy source should be shielded, so that it is not possible for an attacker to change the result.

For the scenario of highly constrained devices, an entropy source which does not need dedicated hardware should be used. As described by Holcomb et al. [21] an SRAM PUF is a promising entropy source. A Physical Unclonable Function (PUF) uses physical characteristics of a device as input to gain a unique result. The approach is based on the fact that on start up the single cells can be different states which produces an unpredictable, unique physical fingerprint. In that case, the Static Random Access Memory (SRAM) is used as the source.

Another entropy source might be the Received Signal Strength Indicator (RSSI) level of a transceiver as introduced by Latif et al. [22] for Wireless Sensor Network (WSN). The RSSI level is an indicator showing the power of a received signal.

3.1.2 Transmission via LiFi Using Manchester Encoding

The transmission of the encryption key has to be done in a manner that an adversary cannot eavesdrop or even manipulate the data. For that reason it is a feasible solution to use LiFi as communication channel, since it complies with these requirements and does not need additional hardware. According to Tanenbaum [23] it is self-clocking, which is realized with a transition in the middle of each bit period. This allows to use a single LED. A binary `0` is therefore transmitted with the pattern `01` while `1` uses the pattern `10`. This halves the bandwidth and for that reason doubles the transmission time. Manchester Code can be implemented using On-Off Keying, which is a form of Amplitude-Shift Keying (ASK) and uses the presence and absence of a signal for transmission. In the implementation developed in this bachelor thesis turning the LED on stands for binary `1`, turning it off for `0`. The chosen bitrate is limited by the smartphone's frame rate. Most smartphones are nowadays capable of 120 fps or higher. Since Manchester Code needs twice of the bandwidth the chosen bitrate is $60\,\mathrm{bit\,s^{-1}}$.

Manchester Code is preferred over Differential Manchester Code, because the state of the LED shall be zero after transmission of the trailer to turn it off. In Differential Manchester Code this includes further computation, because a bit value is defined by either the presence or absence of a transition at the beginning of an interval [23]. Therefore the state of the LED depends in this case on the previous bit values, whereas in Manchester Code a trailer ending on binary `0` will always turn the LED off.

3.1.3 Encryption with CCM

As described in Section 1.3.3, CCM allows both encryption and authentication on symmetric keys. Symmetric cryptography has the advantage that it needs less computational power than asymmetric cryptography [24], which is an important factor in a constrained environment. Furthermore CCM provides a convenient way of calculating the MAC value with additional data included [12].

CCM requires to specify parameters, which are chosen in this thesis in the following way:

- `L=2`, which is the size of the length field in byte. This limits the length of plaintext. It is the smallest value possible and allows $2^{8L} = 2^{8\times2} = 65536$ B plaintext. This value most probable satisfies any demands.
- `M=8`, which is the size of the MAC field in byte. This value keeps the message length as short as possible while providing a strong level security.
- `nonce_length=15-L=13`, specifies the length of the nonce in byte.
- `AES` as cipher algorithm for encryption.

Besides that the header data is included in the MAC calculation but not encrypted. This maintains the integrity for this part as well.

Using the same key for both authentication and encryption is considered to have a lower security margin than using separate keys. However, it has been proven that CCM provides enough restrictions so that security can still be guaranteed in this case [25]. Since no security flaws are known at present, CCM is an encryption mode satisfying the requirements of the protocol.

3.1.4 Error Detection and Correction

The commissioning protocol includes an FCS to detect errors during the transmission of the encryption key. The choice is based on the guidelines by Koopman et al. [26, 27].

The generator polynomial `0x1cf` was selected regarding the message length, the CRC length and the minimum Hamming Distance. The minimum Hamming Distance represents the number of errors which can be detected [23]. A CRC with a length of 8 bit is preferred over longer values, since this leads to increased transmission time. The size of messages of the prototype is 184 bit excluding the FCS. According to the guideline the maximum possible Hamming Distance for these parameters is 3 with the polynomial $x^8+x^7+x^6+x^3+x^2+x+1$. Besides that, the seed unequal to zero was chosen to allow zero-codewords, which does not affect the performance of the algorithm otherwise. The algorithm detects up to 3 single-bit errors and furthermore allows to identify burst errors with a length of up to 8 bit.

Another error detection is achieved by the usage of Manchester Code, since each data bit requires a transition in the middle of its interval. For this reason a bit error only remains undetected if both values are flipped.

In the case an error occurs the smartphone informs the user to initiate the commissioning again. The error detection helps to keep the power consumption on the target device as low as possible, since only messages using the correct encryption key are sent by the smartphone.

3.1.5 User Interaction

The user needs to initiate the commissioning process and has to hold the smartphone's camera in a manner that it can receive the signal from the LED. As stated by Kumar et al. [28], it can be assumed that a user is more interested in a fast commissioning process than in a high level of security. The user is focused on the device's application and therefore looks for a convenient and fast solution. As a result, the protocol in this bachelor thesis was designed in a way that the user interaction is kept as small as possible.

3.2 Implementation

The implementation in this thesis was done on a STM32F103C8 bluepill board [4] using RIOT-OS. RIOT-OS is an operating system for the deployment in IoT [29]. The board features a status LED which is used for the LiFi communication. The implementation includes the construction and transmission of the initial message from the IoT device. Furthermore, the processing of the encrypted message is handled. The development of a smartphone application and the generation of a cryptographically secure key are out of scope of this thesis. Due to the last aspect the prototype uses a PRNG to attain a key of 128 bit.

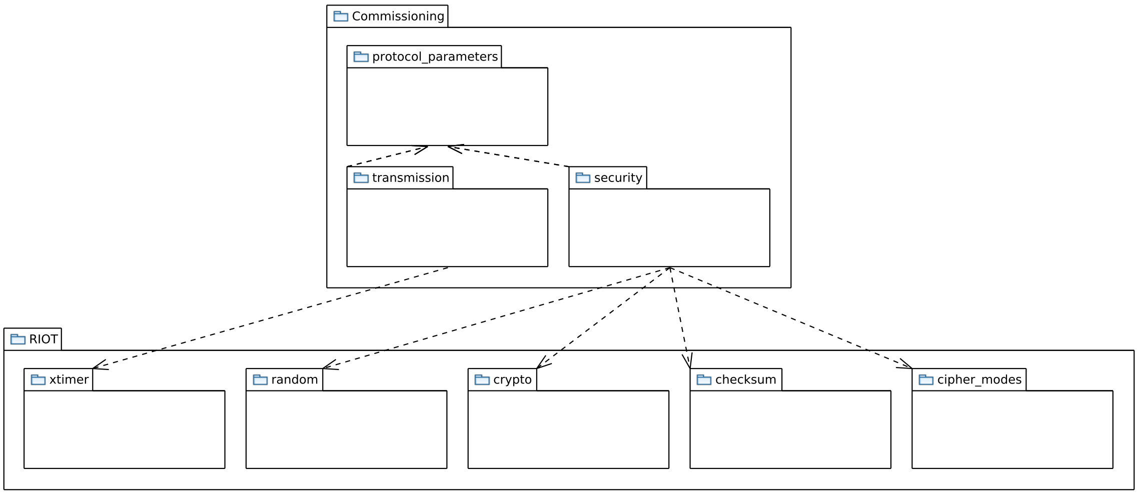

Figure 3.3: Package Diagram of the Prototype Implementation

The message's payload contains the encryption key and an unique identifier of 48 bit length. Furthermore, the FCS is calculated and added to the trailer. As described in Section 3.1, the start of transmission may be missed by the receiver. For that reason the message is sent out twice via LiFi.

For the prototype, the tasks of the smartphone are performed on the bluepill board as well. This covers the encryption process and the creation of the response message.

The board validates the response message and extracts the credentials. If the received version number conforms with the implemented version on the device, the password is decrypted. RIOT-OS allows to implement CCM using AES on block and key sizes of 128 bit. After the decryption, the LED is turned on to notify the user about a successful connection to the network.

As shown in Figure 3.3, the used functions are grouped in the two modules `transmission` and `cryptography`. The module `transmission` handles the LiFi communication by providing the function `transmit` with its subroutines `transmit_zero` and `transmit_one`. After successful connection to the network, the function `transmit_success` is called. The second module `cryptography` includes the generation of a cryptographic key and the decryption of messages. The file `protocol_parameters.h` is used for setting the parameters depending on the used version.

CHAPTER 4

Thesis Outcome

This chapter comprises the analysis of both performance and security. The performance analysis includes the conduction of experiments on the prototype implementation, together with their evaluation. The security analysis shows different attacks during the commissioning process.

4.1 Performance Analysis

As described in Section 1.1, constrained devices are characterised by high memory restrictions and low computational power. Besides that the user wishes for a fast process, which makes time an important factor as well. In the following, the conducted experiments for evaluating the performance are described and evaluated.

4.1.1 Experiments

Two experiments were conducted to test the performance of the prototype. The transmission of the LiFi signal was tested in the first experiment. The second experiment is used to investigate the memory footprint.

Transmission of LiFi Signal

The conducted experiment serves to measure the transmission time of the LiFi signal. Furthermore it allows to verify the correctness of the transmitted data. The experiment was conducted as described in the following.

A 24 MHz Salae logic analyser was connected for capturing the voltage on the LED pin. A visualisation of the complete result is shown in Figure 4.1, with the corresponding bit values beneath. The message consists in total of 193 bit, composed of a header with 8 bit, 176 bit payload and a 9 bit trailer. It was assumed that the source address is formed of 48 bit and a 128 bit cryptographic key is used.

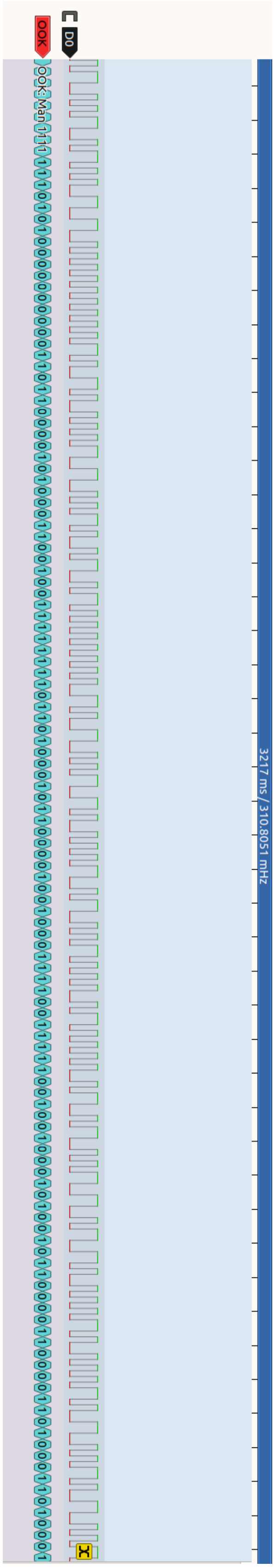

Figure 4.1: Total LiFi Signal Produced on the IoT Device

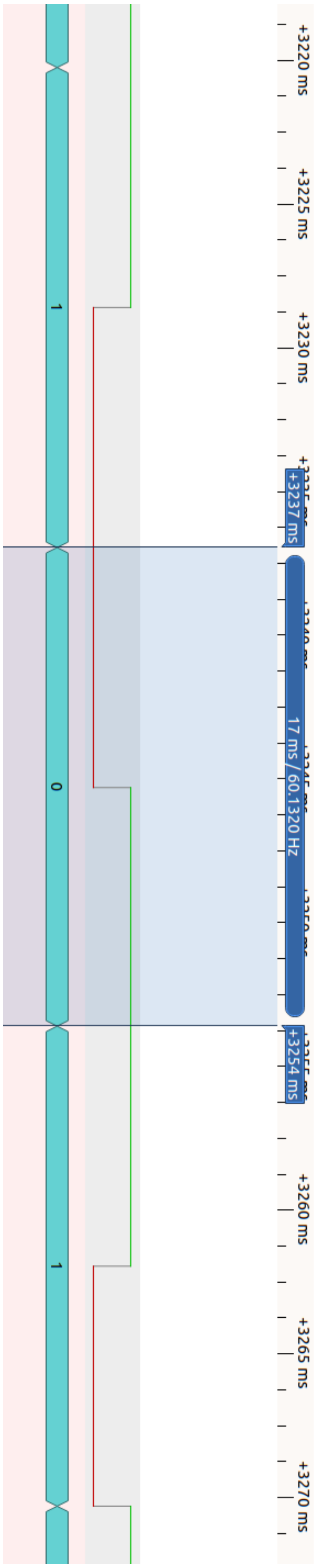

Figure 4.2: Single Bit of the LiFi Signal Produced on the IoT Device

Memory Segment	Prototype with Additional Modules	Additional Modules	Prototype Without Additional Modules
text	16544	8080	8464
data	112	112	0
bss	2644	2644	0

Table 4.1: Memory Usage Results for Prototype and Additional Modules

Memory Storage Medium	Total Available	Memory Usage of Prototype	Proportion
RAM	20 KiB	0 KiB	0 %
Flash	124 KiB	8.27 KiB	6.67 %

Table 4.2: Memory Consumption of the Prototype in Comparison to Available Memory

For correct identification of the values with the software *pulseview* [30] the clock synchronization pattern needed to be changed to `0000 0001`. As shown in both graphics, all bits can be identified without any error.

The measurement of the transmission time assumes a message of 193 bit as described above. The transmission time of with a bitrate of 60 bit s^{-1} can be calculated the following:

$$\frac{193\text{ bit}}{60 \times 10^{-3}\frac{\text{bit}}{\text{s}}} = 3216.67\text{ ms}$$

For a single bit, the transmission time is calculated as:

$$\frac{1\text{ bit}}{60 \times 10^{-3}\frac{\text{bit}}{\text{s}}} = 16.67\text{ ms}$$

The experiment in Figure 4.1 shows that the complete transmission takes 3217 ms. A single bit is therefore transmitted in about 17 ms as presented in Figure 4.2.

Memory Consumption

The implementation is written for a bluepill board which has a flash size of 128 KiB and a Random Access Memory (RAM) of 20 KiB [4]. According to RFC 7228 [7] it can be classified in class 2 of constrained devices.

Since it can be assumed that the IoT device has already loaded RIOT-OS and libraries for cryptography and timing, the result is compared to the memory consumption of these components. The libraries are most likely necessary for other applications as well. The used libraries are `crypto`, `checksum`, `cipher_modes` and `random` for cryptography as well as `xtimer` for timing. Table 4.1 shows the results of the experiment. The project was compiled with Link Time Optimisation (LTO), which is a inter-procedural optimisation that allows to tread the different units of a compilation as a single file [31]. The results were obtained by running the command `make info-buildsize`. The last column of this table shows the remaining memory consumption of the prototype without the above mentioned components. In Table 4.2, these results are compared to the available memory.

The RAM usage is the sum of the memory sections `bss` and `data`, flash memory is calculated by adding up `data` and `text`. As it can be seen in Table 4.2, the application uses no RAM. Besides that, it consumes 8.27 KiB of flash, which are 6.67 % on the used bluepill board.

4.1.2 Evaluation

The experiment described in Section 4.1.1 can be used for analysing the correctness of data transmission. Furthermore the time consumption can be seen. The results prove that all bits can be correctly identified. Furthermore, the duration of the transmission of 3217 ms is only slightly different to the theoretical value of 3216.67 ms. The deviation of the actual result may come from measurement errors.

In terms of user friendliness a commissioning time of about 3 s in the successful case is a satisfying value as well. For a deeper analysis of the commissioning time a user study including the smartphone application would need to be conducted.

Besides that the second experiment shows that the memory consumption is very low, for it only consumes 8.27 KiB of flash and nearly no RAM. According to RFC 7228 [7] constrained devices which are classified in class 0 possess less than 10 KiB RAM and 100 KiB flash. Even on these highly constrained devices the implementation would only use a reasonable amount of memory.

For this reason, it can be concluded that the protocol can be implemented even in highly restricted environment where a low memory consumption is needed.

4.2 Security Analysis

The proposed protocol aims at providing a secure way of commissioning an IoT device. In the following, the security threads are analysed and evaluated. An attacker may aim at gaining knowledge of the network access credentials itself. Besides that a Denial of Service (DoS) attack could prevent the commissioning of the smartphone. Both scenarios are evaluated with attack trees. This allows the analysis of different attackers, their techniques and with that helping to understand the security risks of a system [32]. Section 4.2.3 evaluates the previously analysed security threads.

4.2.1 Access Network Credentials

Figure 4.3 shows the different attacks an adversary may launch to access the network credentials during commissioning. In terms of security aspects, the attack would affect the message's confidentiality. The intruder could access information that is supposed to be accessible only to the smartphone and the target device. In the following, the different subtrees will be analysed by referring to the attack tree.

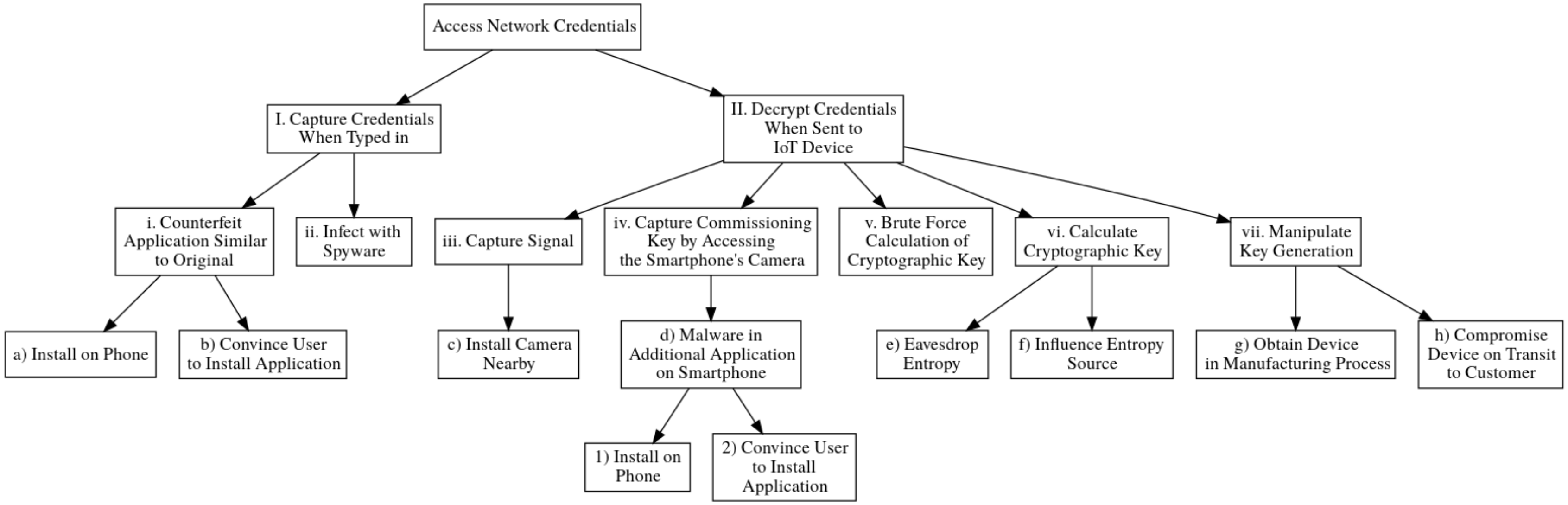
Access Network Credentials
I. Capture Credentials When Typed in
II. Decrypt Credentials When Sent to IoT Device
i. Counterfeit Application Similar to Original
ii. Infect with Spyware
iii. Capture Signal
iv. Capture Commissioning Key by Accessing the Smartphone's Camera
v. Brute Force Calculation of Cryptographic Key
vi. Calculate Cryptographic Key
vii. Manipulate Key Generation
a) Install on Phone
b) Convince User to Install Application
c) Install Camera Nearby
d) Malware in Additional Application on Smartphone
e) Eavesdrop Entropy
f) Influence Entropy Source
g) Obtain Device in Manufacturing Process
h) Compromise Device on Transit to Customer
1) Install on Phone
2) Convince User to Install Application

Figure 4.3: Attack Tree: Access the Network Credentials

The goal of accessing the network credentials can be reached by (I) capturing the network credentials when they are typed in or (II) decrypting the credentials when they are sent to the IoT device.

The analysis of sub goal I shows an attacker could (i) install a counterfeit application of the original commissioning application. If the user tries to use the application to commission a new device, the information would be available to the attacker as well. Another method to achive this goal is to (ii) infect the smartphone with spyware, which is a form of malicious software to collect data without the user's knowledge [33]. This spyware would extract the data and make it available to the attacker.

Sub goal II requires the intruder to get hold of the encryption key. One approach is to (v) break the key computationally. According to the specification of AES by the Institute of Electrical and Electronics Engineers (IEEE) [34], it is believed to be unfeasible to guess a cryptographic key consisting of 128 bit. This requires on average 2^{n-1} guesses with n being the length of the key, in this case that would lead to 2^{127} computations. According to the NIST [35], AES-128 has a security strength which is at present unfeasible to break by an attacker. This is all based on the assumption that a strong commissioning key is generated. If this is not the case an adversary could be able to compute it within a shorter time.

Another opportunity is to (iii) eavesdrop on the LiFi signal sent out by the target device. However, the commissioning can take place at any location chosen by the user. For that reason the attacker needs to (c) install a camera in a way which is not suspicious for the user and at a distance that is low enough to capture the signal. Since the range of the signal is limited by the surrounding walls, the adversary needs to place the camera close.

The LiFi signal can also be captured directly by (iv) accessing the smartphone's camera. This requires to (d) install malware on the smartphone with the permission to use the camera.

Another attack could be based on the assumption that if the entropy source for the key generation is (e) being eavesdropped on or (f) influenced, a formerly strong PRNG will use predictable seed values. With that manipulation it would be possible to (vi) calculate the key and use it for decryption.

Provided that the attacker is able to get hold of the target device, it is also possible to (vii) change the PRNG undetected so that it produces predictable results. The intruder could (g) influence the manufacturing process or (h) get hold of the device while it is shipped to the customer.

4.2.2 Denial of Service Attack to Prevent Commissioning

Another goal could be to run a DoS attack to prevent the commissioning as summarised in Figure 4.4. The subtrees I and III aim at jamming the communication between smartphone and IoT device. Another strategy is to stop one of the devices from working (III, IV). A successful DoS attack would break the security goal of availability.

An adversary may (I) jam the communication between the devices in the first phase of the protocol when the commissioning key is shared via LiFi. One approach to reach this goal is to (a) install another source of light. This needs to be strong and close enough

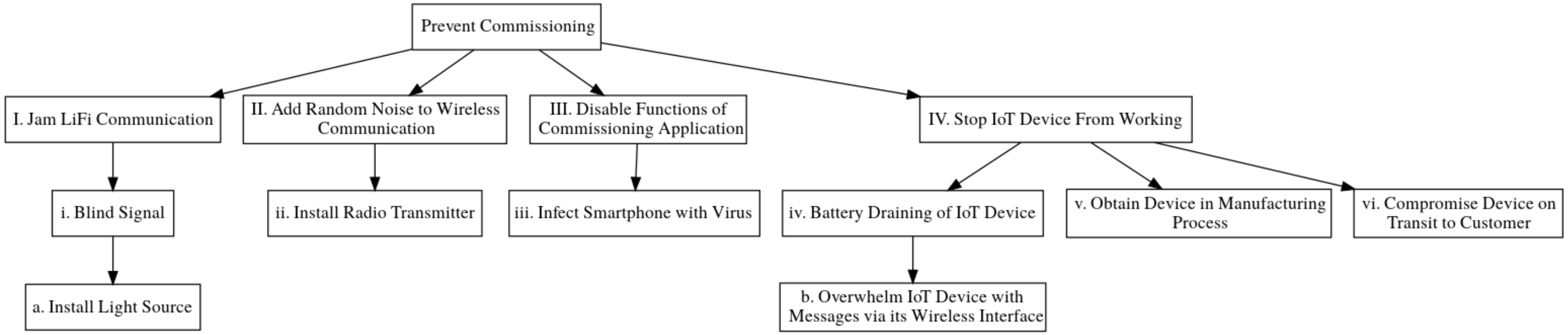
Prevent Commissioning
I. Jam LiFi Communication
II. Add Random Noise to Wireless Communication
III. Disable Functions of Commissioning Application
IV. Stop IoT Device From Working
i. Blind Signal
ii. Install Radio Transmitter
iii. Infect Smartphone with Virus
iv. Battery Draining of IoT Device
v. Obtain Device in Manufacturing Process
vi. Compromise Device on Transit to Customer
a. Install Light Source
b. Overwhelm IoT Device with Messages via its Wireless Interface

Figure 4.4: Attack Tree: Prevent Commissioning

to (i) interfere with the signal sent out from the IoT device. The smartphone would only receive binary `1` since it would be unable to distinguish between a high and low signal. Once again, this requires the adversary to install a light source in the user's environment undetected. Furthermore the smartphone might already be able to filter additional light so that it has no impact on the received signal. Even if it is successful, the user may notice the undesired light if it is operating in the spectrum of visible light. In this case the signal could be shielded with the hands or additional objects. The only solution for an adversary may be the installation of a laser pointer, but this needs to be placed with high precision. This will be impossible in most of the cases since the user will not hold the smartphone in a predictable and fixed position. However, an infrared laser pointer would not be visible for the human eye, but could still influence the camera's perception.

The communication may be jammed in the second phase of the protocol as well. The adversary could (II) listen to the wireless channel over which the encrypted network credentials are send and alter it. In that case the IoT device will notice this because of the MAC value included in the encrypted message, but will be unable to correct it.

As per the attack analysed in the previous section it is possible to compromise the smartphone. In this scenario the adversary plans to (III) stop the commissioning application from working. This could be achieved with a (iii) virus that attaches itself to the application.

The last subtree aims at (IV) influencing the IoT device directly so that it cannot be commissioned. If the adversary is able to (v, vi) get hold of the IoT device before it is delivered to the customer, it could get compromised to stop working. Besides that, the intruder could try to drain the IoT device's battery and for that reason make it unable to work. For this an adversary could (b) overwhelm it with messages via its wireless interface.

4.2.3 Evaluation of Security Attacks

As analysed in Section 4.2.1, an attacker may aim at stealing the network credentials. The analysis of the security threads shows that an intruder who is able to monitor both communication channels might be able to access this information. Attacks such as capturing the LiFi signal through the smartphone's camera aim at infecting the smartphone with malware. Since the smartphone application is out of the scope of this thesis, the risk of this cannot be evaluated. The prevention of the corresponding scenarios relies on the development of the smartphone application and the user's own responsibility. Besides that, the usage of LiFi ensures that an receiver needs to be physically proximate. For that reason, it is difficult to install a camera for capturing the signal without the user's notice. Furthermore it has been shown that it is unfeasible to guess a strong cryptographic key of 128 bit. However, the manipulation of the key generation process cannot be maintained through the protocol.

Attacks which aim at preventing the commissioning are difficult to defend. The highest weakness lies within the wireless communication with the IoT device. This can be misused to either drain the device's battery or to add random noise to the message. Although the LiFi communication may also be disturbed, it is difficult for an attacker to install a source of light without notice of the user and in a position that it is able to interfere with the IoT device's signal. As for the attack on the network credentials, the risk of an attack which manipulates the smartphone is difficult to evaluate.

It is especially important that the message's confidentiality is preserved, since otherwise an adversary could gain access to a whole network. All in all it can be concluded that the protocol introduces a large number of countermeasures to make the stealing of the network credentials improbable. The prevention of a DoS attack is difficult to ensure through the commissioning protocol.

CHAPTER 5

Conclusion

In this chapter, the specification of the commissioning protocol is summarised. It is evaluated how the requirements and goals of this thesis, which were identified in Chapter 1, are met. As a result, future work on the protocol is identified in the last section.

5.1 Summary

This thesis introduces a protocol which allows the secure commissioning of IoT devices. The protocol was tested with a prototype implementation and evaluated in terms of performance and security. The commissioning protocol has been proven as an approach that satisfies the described requirements in the field of IoT with focus on security.

The protocol makes use of the available hardware on IoT devices to allow deployment in a wide field. The only requirement on the devices is a LED. Additionally the user has to install an smartphone application as counterpart. Differently to other approaches the protocol uses a self-generated cryptographic key to securely share the network credentials with the target device. This cryptographic key is transferred to the user's smartphone by using LiFi, which operates as an Out-of-Band channel and reduces therefore the risk of eavesdroppers. The transmission uses Manchester Code as a self-clocking line code to make the usage of a single LED possible. By using the received symmetric key, the network credentials can be securely shared with the IoT device over its wireless interface. The encryption is done with CCM, which provides not only integrity but also authenticity of the message. The received credentials allow the target device to connect itself to the local network.

A prototype was used to analyse the memory consumption on the target device. The application consumes only 8.27 KiB of flash memory and no RAM. For that reason, the application is a feasible solution for devices with high restrictions on memory.

The user experience plays an important role in many applications of the IoT, where a convenient process is often desired. A user who is planning to integrate devices such as temperature sensors in a smart home network would only need to install the commissioning application on his smartphone and hold its camera in front of the LED. Since most IoT devices already use an LED to notify the user about correct operation this can serve as the transmitter of the signal. The same method could be used to integrate other devices into the network as well, so that the user does not need to get familiar with other commissioning strategies. Besides that, the commissioning will only consume about 3.217 s if no retransmission has to be done. This result should satisfy the user's demands of a fast commissioning process. The low memory footprint makes it also possible to integrate the application on a wide range of devices.

The security attacks on the protocol have been evaluated, by analysing an adversary who aims at gaining knowledge of the local network credentials and a DoS to prevent the commissioning. In both cases, it was evaluated that the highest risk lies within the smartphone application which could be infected with malware. Other attacks require the adversary to be physical close. The communication between the devices utilises a high number of security measures that make attacks improbable.

5.2 Future Work

The prototype of the application does cover the commissioning process on the target device. In addition, a smartphone application as the counterpart has to be developed to make a complete realisation of the specification possible. This would also make a thorough analysis of the reliability possible.

Another aspect is the generation of a cryptographically strong key on the IoT device. A concept for this was presented in this thesis as well. As the next step this has to be included in the implementation. If all components are available, a user study should be conducted to analyse the convenience of the commissioning.

Besides that further variants of the protocol could be developed. As it was shown in Chapter 4.2.3, an adversary who is able to monitor both the LiFi signal and the response channel might be able to access the credentials. As a result, asymmetric encryption could be used instead. The feasibility of this largely depends on the application area, since this type of cryptography is in need of higher computational power [24].

Furthermore it is possible to use another Out-of-Band channel for securely sharing the encryption key. If an microphone is available instead of a LED ultrasonic sound could be used, which can be received by a smartphone as well. In the following it would need to be tested if this fulfils the requirements on security, reliability and user experience.

Bibliography

[1] Jayavardhana Gubbi et al. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future Generation Computer Systems* 29.7 (Sept. 2013), pp. 1645–1660.

[2] *IoT: number of connected devices worldwide 2012-2025 | Statista.* `https://web.archive.org/web/20191231143648/https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide`. [Online; accessed 15. Jan. 2020]. Jan. 2020.

[3] Constantinos Kolias et al. "DDoS in the IoT: Mirai and Other Botnets". In: *Computer* 50.7 (2017), pp. 80–84.

[4] *Blue Pill - STM32duino wiki.* `https://web.archive.org/web/20190428082446/http://wiki.stm32duino.com/index.php?title=Blue_Pill`. [Online; accessed 8. Jan. 2020]. Jan. 2020.

[5] Leila Fatmasari Rahman, Tanir Ozcelebi, and Johan Lukkien. "Understanding IoT systems: a life cycle approach". In: *Procedia computer science* 130 (2018), pp. 1057–1062.

[6] Alexis Duque et al. "Unleashing the power of LED-to-camera communications for IoT devices". In: *Proceedings of the 3rd Workshop on Visible Light Communication Systems - VLCS '16.* ACM Press, 2016.

[7] C. Bormann, M. Ersue, and A. Keranen. *Terminology for Constrained-Node Networks.* RFC 7228. RFC Editor, May 2014.

[8] Aboul Ella Hassanien and Mohamed Elhoseny, eds. *Cybersecurity and Secure Information Systems.* Springer International Publishing, 2019.

[9] Rodrigo Roman and Javier Lopez. "KeyLED - transmitting sensitive data over out-of-band channels in wireless sensor networks". In: *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems.* IEEE, Sept. 2008.

[10] Harald Haas et al. "Wireless data from every light bulb". In: *TED Global, Edinburgh* (2011).

[11] Harald Haas et al. "What is lifi?" In: *Journal of lightwave technology* 34.6 (2015), pp. 1533–1544.

[12] Morris Dworkin. *Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality.* Tech. rep. National Institute of Standards and Technology, 2004.

[13] Tonko Kovačević, Toni Perković, and Mario Čagalj. “Flashing displays: user-friendly solution for bootstrapping secure associations between multiple constrained wireless devices”. In: *Security and Communication Networks* 9.10 (2016), pp. 1050–1071.

[14] Claudio Soriente, Gene Tsudik, and Ersin Uzun. “HAPADEP: human-assisted pure audio device pairing”. In: *International Conference on Information Security*. Springer. 2008, pp. 385–400.

[15] Cynthia Kuo et al. “Message-in-a-bottle: user-friendly and secure key deployment for sensor nodes”. In: *Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM. 2007, pp. 233–246.

[16] Electricimp. *BlinkUp*. `https://developer.electricimp.com/manufacturing/factoryprocess`. [Online; accessed 22-November-2019].

[17] Toni Perković, Tonko Kovačević, and Mario Čagalj. “BlinkComm: Initialization of IoT Devices Using Visible Light Communication”. In: *Wireless Communications and Mobile Computing* 2018 (June 2018), pp. 1–16.

[18] Amazon. *Amazon Dash Button*. `https://web.archive.org/web/20200121130722/https://www.amazon.com/b?ie=UTF8&node=17729534011`. [Online; accessed 22-November-2019].

[19] Bruce Schneier, Tadayoshi Kohno, and Niels Ferguson. *Cryptography engineering: design principles and practical applications*. Wiley, 2013.

[20] D. Eastlake, J. Schiller, and S. Crocker. *Randomness Requirements for Security*. BCP 106. RFC Editor, June 2005.

[21] Daniel E Holcomb, Wayne P Burleson, and Kevin Fu. “Power-up SRAM state as an identifying fingerprint and source of true random numbers”. In: *IEEE Transactions on Computers* 58.9 (2008), pp. 1198–1210.

[22] Rabia Latif and Mukhtar Hussain. “Hardware-based random number generation in wireless sensor networks (WSNs)”. In: *International Conference on Information Security and Assurance*. Springer. 2009, pp. 732–740.

[23] Andrew S. Tanenbaum. *Computer Networks 3rd Edition Prentice Hall*. 1996.

[24] Swapna B. Sasi et al. “A general comparison of symmetric and asymmetric cryptosystems for WSNs and an overview of location based encryption technique for improving security”. In: *IOSR Journal of Engineering* 4.3 (2014), p. 1.

[25] Jakob Jonsson. “On the security of CTR+ CBC-MAC”. In: *International Workshop on Selected Areas in Cryptography*. Springer. 2002, pp. 76–93.

[26] P. Koopman and T. Chakravarty. “Cyclic redundancy code (CRC) polynomial selection for embedded networks”. In: *International Conference on Dependable Systems and Networks, 2004*. IEEE, 2004.

[27] Philip Koopman. *Best CRC Polynomials*. `https://users.ece.cmu.edu/~koopman/crc/index.html`. [Online; accessed 6. Jan. 2020]. Jan. 2020.

[28] Arun Kumar et al. “A comparative study of secure device pairing methods”. In: *Pervasive and Mobile Computing* 5.6 (Dec. 2009), pp. 734–749.

[29] Emmanuel Baccelli et al. “RIOT OS: Towards an OS for the Internet of Things”. In: *2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*. IEEE. 2013, pp. 79–80.

[30] PulseViewSigrok. *PulseView - sigrok*. `https://sigrok.org/w/index.php?title=PulseView&oldid=14849`. [Online; accessed 14. Feb. 2020]. Feb. 2020.

[31] *LinkTimeOptimization - GCC Wiki*. `https://gcc.gnu.org/wiki/LinkTimeOptimization`. [Online; accessed 18. Feb. 2020]. Feb. 2020.

[32] Bruce Schneier. “Attack trees”. In: *Dr. Dobb’s journal* 24.12 (1999), pp. 21–29.

[33] Federal Trade Commission et al. “Monitoring software on your PC: Spyware, adware, and other software”. In: *Federal Trade Commission* (2005).

[34] W.E. Burr. “Selecting the Advanced Encryption Standard”. In: *IEEE Security & Privacy* 1.2 (Mar. 2003), pp. 43–52.

[35] Elaine Barker. *Recommendation for Key Management Part 1: General*. Tech. rep. Jan. 2016. URL: `https://doi.org/10.6028/nist.sp.800-57pt1r4`.

Appendix

A.1 Parameters Used For The Prototype

```
#ifndef PROTOCOL_PARAMETERS_H
#define PROTOCOL_PARAMETERS_H

#define COMMISSIONING_PROTOCOL_VERSION 0

#define ADDRESS_LEN_BYTE 6                                  //unique identifier as address of IoT device
#define PSWD_LEN_FIELD 2                                    //parameter for ccm: space needed for encrypting the byte value of password length
#define PSWD_MAX_LEN_BYTE 16                                //assume that the credentials are no longer
#define KEY_LEN_BYTE 16                                     //128 bit key
#define MAC_LEN_BYTE 8                                      //parameter for ccm: length of MAC value
#define MAX_MSG_LEN 40                                      //max size of message from smartphone to iot device
#define LIFI_PACKET_LEN (KEY_LEN_BYTE + ADDRESS_LEN_BYTE + 2)   //length of package transmitted via lifi: header(1 Byte)+key(16 Byte)+FCS(1Byte)
#define NONCE_LEN_BYTE (15 - PSWD_LEN_FIELD)                //parameter for ccm: length of nonce
#define ADD_DATA_LEN_BYTE (NONCE_LEN_BYTE + 2)              //parameter for ccm: length of additional data to be encrypted
#define CIPHERTXT_LEN_BYTE (MAC_LEN_BYTE + PSWD_MAX_LEN_BYTE)   //length of ciphertext

#endif
```

protocol_parameters.h