Universität der Bundeswehr München

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Master Thesis
at Columbia University, Department of Computer Science

Secure Authentication of Remote IoT Devices Using Visible Light Communication: Transmitter Design and Implementation

Submitted by:	Hagen Odenthal 1141178
Supervising Professor:	Prof. Dr. Henning Schulzrinne Prof. Dr. Gabi Dreo Rodosek
Advisors:	Dipl.- Ing. Jan Janak Marcus Knüpfer M.Sc.
Date:	24. August 2018

Universität der Bundeswehr München
Fakultät für Informatik
Institut für Technische Informatik

Abstract

The number of Internet of Things (IoT) devices is increasing every year. This also involves households and the trend is expected to be continued. Therefore, a new way to authenticate IoT devices by using the visible light communication (VLC) channel is introduced. For authenticating devices which are out of the user's reach, this new approach is valuable and better suited than existing approaches. This will also be needed in the future, when every electrical device in a household is connected to the network. To show that this concept is working, a smartphone with a camera and a Raspberry Pi with a tri-color LED is used. Depending on the user's smartphone capabilities the authentication takes between 8 and 20 seconds, by using the 4-level frequency shift keying (FSK) modulation method. The smartphone is handheld pointed at the LED of the IoT device and receives the transmitted authentication data. It works reliably on distances up to 4 m. This approach will likely have impact on the state of the art authentication of IoT devices. The concept of this new approach is separated in two part. The transmitter part, which this thesis describes. Alexander Linßen's thesis describes the receiver part [14].

Zusammenfassung

Die Anzahl an Internet of Things (IoT) Geräten steigt jährlich. Dies gilt auch für Geräte in Haushälten und es wird er wartet, dass der Trend sich fortsetzt. Aus diesem Grund, wird eine neue Art der Authentifizierung, bei welchem der visuellen Lichtkanal benutzen wird, eingeführt. Dieser Ansatz ist nötig um Geräte zu authentifizieren, welche sich nicht in der Reichweite des Nutzers befinden. Dies wird auch in Zukunft benötigt, wenn alle elektronischen Geräte in einem Haushalt mit dem Netzwerk verbunden sind. Um zu zeigen, dass das Konzept funktioniert, wird ein Smartphone mit Kamera und einen Raspberry Pi mit einer Dreifarben LED benutzt. Je nach Smartphone dauert eine Übertragung zwischen 8 und 20 Sekunden, wenn die 4-Level Frequenz Modulation verwendet wird. Das Smartphone wird in der Hand gehalten und auf die LED des IoT Gerätes gerichtet. Dabei werden die Daten zur Authentifizierung übertragen. Dieses Verfahren funktioniert zuverlässig auf einer Distanz von bis zu 4 m. Dieser Ansatz könnte einen großen Einfluss auf die aktuelle Vorgehensweise solcher Geräte haben. Das Konzept dieses neuen Ansatzes is in zwei Teile separiert. In dieser Arbeit wird die sendende Hälfte des Projektes behandel. In Alexander Linßen's Arbeit wird die empfangende Hälfte beschrieben [14].

Acknowledgments

In this chapter, I as the author of this thesis and Alexander Linssen as the author of [14], would like to thank our supervisors Professor Dr. Henning Schulzrinne and Professor Dr. Gabi Dreo Rodosek for giving us the opportunity to work on our master's project in the Internet Real Time Lab and at this truly great university.

To the same level we would like to thank our advisor Jan Janak, at Columbia University, for his extraordinary support during the project and also, for being second reader of our theses.

We would also like to thank Marcus Knüpfer, our advisor at Universität der Bundeswehr München, for all his guidance.

Last but not least, we would like to thank our employer, the Bundeswehr (German Armed Forces), to enable us to write our master's theses in the United States as international students as well as improving our inter-cultural awareness and language skills.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AP access point

API application programming interface

CA certificate authority

CRC cyclic redundancy check

FEC forward error correction

FPS frames per second

FSK frequency shift keying

GPIO general purpose input/output

HDLC high level data link control

HSV hue-saturation-value

HTTP hypertext transfer protocol

IoT internet of things

LED light-emitting diode

MITM man in the middle

NFC near field communication

OOB out-of-band

P2P peer-to-peer

PBC push button configuration

PIN personal identification number

PWM pulse-width modulation

QR quick response

RGB red-green-blue

TLS transport layer security

VLC visible light communication

WPS Wi-Fi protected setup

1 Introduction

Many new Internet of Things (IoT) devices are being developed and produced. According to the forecast by Gartner [6], the number of IoT devices will drastically increase the next few years. Gartner says that by the year 2020 there will be over 20 billion connected "Things". Figure 1 shows the fore cast by Gartner.

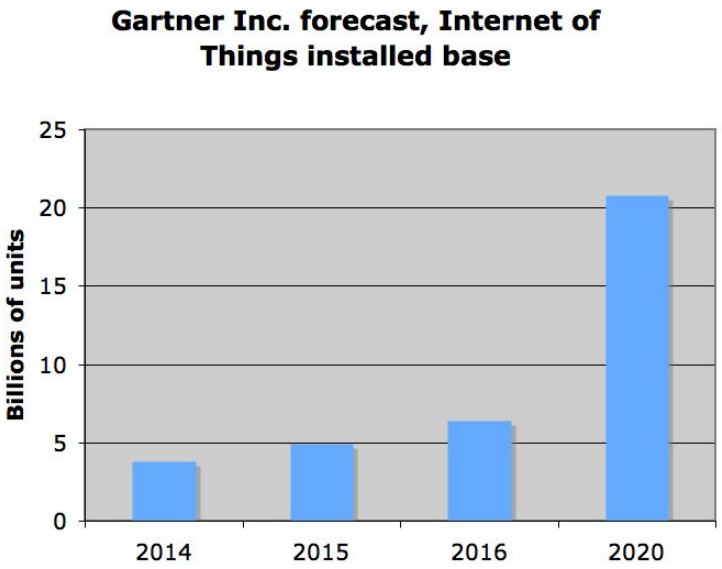


Figure 1: Forecast by Gartner 2020 [27]

One particular problem is secure IoT device authentication in the absence of a trusted third-party certificate authority. To authenticate a IoT device means, that the user ensures that the device he/she is communicating with, is this device indeed and not a different one. E.g., moving in an apartment where IoT devices are already installed. If the IoT devices installed in an apartment were used by the previous tenant, the new tenant may need to reset those devices to factory defaults. This includes potentially re-generating all security credentials to prevent the previous tenant from accessing those devices. This process requires a means to re-generate all security credentials used by the IoT device.

The general problem in IoT device authentication is that there is no trusted authority as on the web, see Figure 2a. The web uses certificate authorities (CA) which sign and issue certificates. The web browser comes with a list of trusted CA certificates. The list is either provided by the operating system, or is installed into the browser by the vendor. In either case, each CA certificate

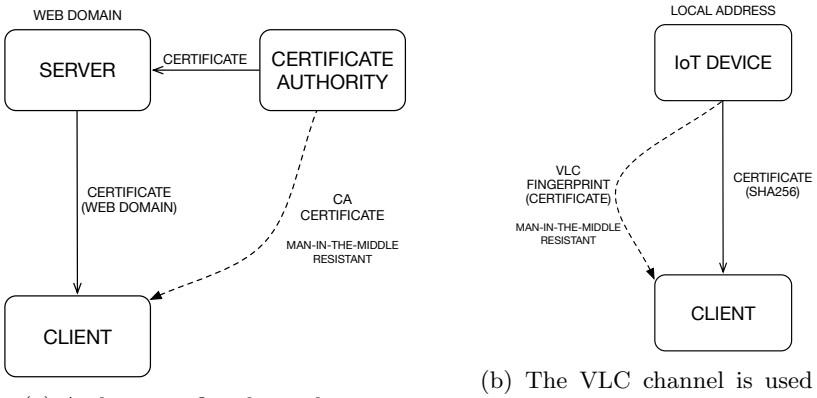
on the list needs to be carefully checked to avoid MITM attacks. The CAs' certificates are exchanged with the browsers or operation company to establish a man in the middle (MITM) resistant channel.

The authentication of IoT devices cannot use the existing web CA model, because IoT devices often do not have verifiable names. Thus, a different authentication mechanism is needed for IoT devices, one which does not rely on third-party CAs. There are many ways to authenticate an IoT device, including near field communication(NFC), quick response (QR) codes, or personal identification number (PIN)s. Most of the existing methods rely on physical proximity between the authenticated IoT device and the authenticator. Such methods are not suitable for IoT devices that are physically out of reach, or devices that require special tools to access, e.g., a ladder if the device is mounted under the ceiling.

In this and Alexander Linssens thesis [14], we introduce a new authentication approach using the visible light communication (VLC) channel. In particular, we use a red-green-blue (RGB) light-emitting diode (LED) and a smartphone with a camera to authenticate a remote IoT device's transport layer security (TLS) server certificate by transmitting its SHA-256 fingerprint via a visible light channel, this is illustrated in Figure 2b. The fingerprint of the certificate is sent because it is enough to authenticate the device and contains only 256 bit data.

This thesis is guided by three questions:

- How to design a usable transmitter using minimal hardware?
- How to port a transmitter for this approach to other devices?
- How fast and reliable is the VLC channel?



(a) A client verifies the authenticity of a web domain certificate by using a trusted CA

(b) The VLC channel is used as the MITM resistant channel because there is no CA for IoT devices

Figure 2: Difference between authentication approaches on the web and IoT.

The goal of this project to design, develop, and evaluate a concept of an authentication system for IoT devices based on visible light communication. The requirement for this concept, as well as background information are provided, in Section 3 and 4. The related work Section 5 gives an overview on prior research on VLC and on current state of the art authentication of IoT devices, motivating why another approach is needed. In the system architecture Section 6 the basics of the whole project are presented. The transmitter Section 7 shows the design and implementation, as well as the challenges. Experiments for color detection and transmission time are described in the evaluation Section 8. The results and analysis Section 9 discusses the result of the experiments. Section 11, summarizes the project, explains limitations and points to future work.

2 Problem Description

The vast majority of IoT devices are designed to be self installed by the user. We assume that this will not necessarily be the case in the future as the number of network-connected devices grows. Many devices require professional installation or calibration, some devices will be pre-installed by the building superintendent or a construction company. At the moment IoT devices are small devices like light bulbs or smart sockets, these devices are typically installed by the user. Once devices such as ovens or air conditioners are connected to the network, they will be set up by a company. Imagine an apartment where nearly every powered device is connected to the network, e.g., security cameras or water pumps. The landlord or a company installed these devices into a fixed infrastructure. Although, this is not the case for today, it is most likely to happen in the future. The problem we are facing is connecting these devices, some of them may be out of physical reach for the user, into its own secure network.

Without authentication, an attacker could perform a MITM attack by setting up a device that pretends to be the IoT device that the user wants to connect to. When the user unintentionally connects to the attacker's device, this device can secretly forward all the data to the IoT device and the other way around. So the user and the IoT device think there is a direct link, but the attacker is able to intercept and manipulate the traffic. For example, the MITM could hijack the video stream of a security camera in the apartment. MITM attacks could include watching the videostream without manipulation of the data or either manipulating the video stream or interrupting it. That means an attacker could spy on people in the apartment or could gain access to the apartment while streaming a pre recorded video sequence.

There are two ways to authenticate devices in general, with or without a trusted third-party. By using a trusted third-party the user trust that the authentication of those devices is done correctly and that they securely communicate the results. This can be a construction company, a building superintendent, a landlord, or somebody else. The other approach is not to rely on a third-party. The user authenticate the devices on its own. Such authentication mechanisms often rely on physical proximity. That, however, does not work well for many of the devices described earlier, because not all of them are in physical proximity to the user. Or it may be inconvenient for the user to reach them.

Devices which are physically not reachable like smoke detectors or cameras can be authenticated with the approach in this thesis.

This thesis shows a usable alternative to authenticate devices without the need to physically access them. Therefore, it shows the functionality using the VLC channel.

3 Requirements

The following requirements, for a secure authentication of remote IoT devices, are derived from the problems described in the problem description.

In the example of a tenant moving in an apartment with an existing IoT infrastructure the tenant needs to authenticate the devices once. Therefore, a device which the tenant already owns is required as the receiver, there should be no need of purchasing any special equipment.

IoT devices are developed for a particular use and therefore, they are cost-efficient. So, there should not be additional hardware which causes the IoT devices to be more expensive than before. Either use the hardware which the IoT devices already has or add a little hardware which would not bother the manufacturer. Also, the hardware used to authenticate the device should only take very little space in order to fit on any IoT device, regardless the form factor.

Since not everybody who uses IoT devices is an IT-expert, the authentication method should not require any professional knowledge or experience but should be easy to use by everyone.

Another requirement is that the concept is secure against MITM attacks and easily allows the user to detect a hijacking of the authentication process.

This concept needs to be able to authenticate an IoT device, even if it is out of the users reach. The receiver should be handheld and the time to authenticate a remote IoT device should be less than using other methods. Therefore, no high-bandwidth authentication channel is needed.

The authentication method should be dynamic and only visible on request. That means, it has no label printed on it or other indicators which are shown static. The authentication data is only shown during the authentication process. This method should base on an already existing one, which is considered secure and should only be started on request. It should be possible to avoid third-parties. Which means, if the tenant does not trust the landlord to provide the correct authentication data for the IoT devices, it is necessary to regenerate new authentication data, to ensure that only the tenant has knowledge of them.

Since there is a great variety in IoT devices, it is necessary that this concept is usable not only in specific case but in general.

Summarizing, the following requirements are important for the given use case:

- no special additional hardware for IoT devices
- easy to use by everyone
- secure against MITM attacks
- no special equipment for authentication
- receiver is handheld and remote to IoT device
- no need for a high-bandwidth authentication channel
- dynamic authentication data
- portable

4 Background

This section contains background information which is needed to understand the project. First, we describe VLC. Followed by information on data transmission, in particular modulation and forward error correction.

4.1 Visible Light Communication

This section focuses on VLC and gives an explanation how it works. VLC is a data communication method which uses only the visible light spectrum between wavelength of 375 nm and 780 nm as shown in figure 3.

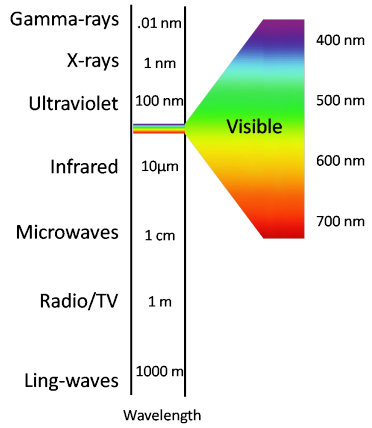


Figure 3: Visible light spectrum [3]

This frequency does not penetrate walls, consequently, it requires a line of sight to the object. In this project, this property offers an additional protection against attackers.

4.1.1 Color Models

This section provides information about the RGB and hue-saturation-value (HSV) color models. The RGB color model, which is based on the three primary colors red, green, and blue, is widely used in displays and computer graphics. Figure 4 shows a cube represents the color model.

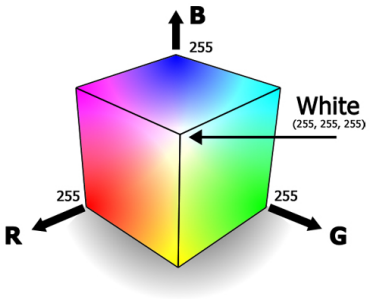


Figure 4: RGB cube [22]

One byte represents one primary color. This system is additive, so one color is represented with a triple of bytes, where one byte is the intensity of red, one of green and one of blue. As shown in Figure 4, if all values are set to their maximum the color is white. When all primary channel intensities are set to 0, the resulting color is black [10, 20]. Computer vision and image analysis uses the HSV model [10]. The reason is that HSV separates the chrominance and hue, from the luminance, i.e., the brightness of the color does not influence the hue value [20].

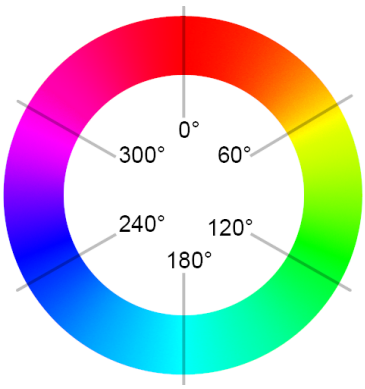


Figure 5: Hue circle [8]

As Figure 5 illustrates, the hue value can be represented by a circle. Therefore, the hue value has a range from 0 to 359. The saturation component defines the intensity of the color. When it is at a maximum, the color is deep and brilliant; when it decreases, it fades more and more. The value or brightness component describes how light or dark the color is. Any HSV color with the brightness 0 is black [20]. HSV is necessary for image processing and is used during the calibration. HSV cann be converted to RGB and vice versa [10].

4.2 Data Transmission

Data transmission is the process of sending data over a communication medium to a device. It enables one or more devices to communicate [4].

4.2.1 Modulation

Modulation is the process of varying a periodic waveform, the carrier signal, with additional information, which is the message [19] .

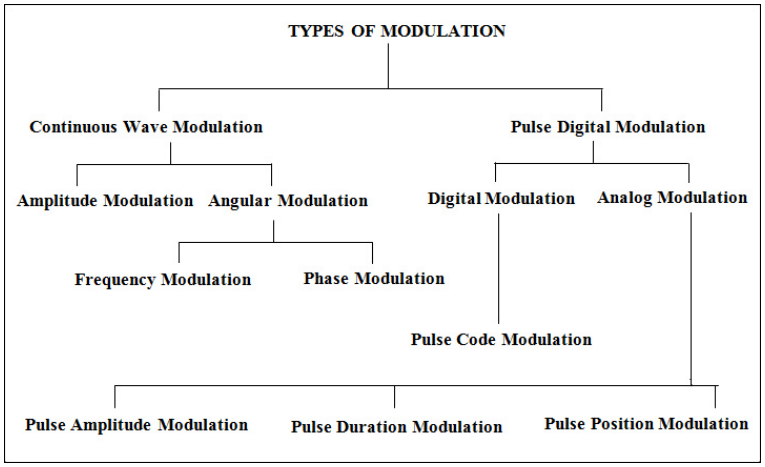
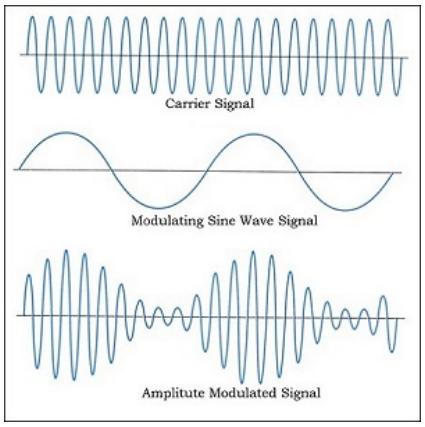
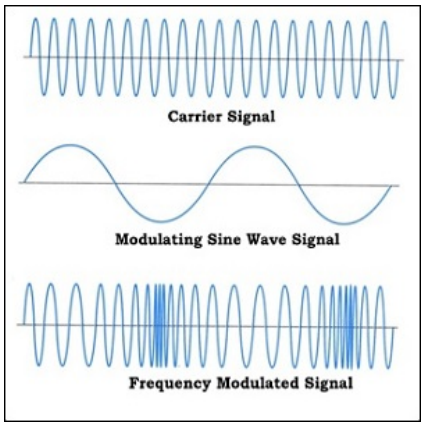


Figure 6: Types of modulation and their relationship [19]

Figure 6 gives an overview of modulation types. In the case of VLC, we need to have a closer look at the continuous wave modulation. The amplitude modulation, as the name says varies the amplitude and not the angle. That means, in our case that the same color is used but with a different intensity. For example, the On-Off-Keying-Modulation, '0s' are represented by no amplitude, that means no color is shown, LED is turned off. '1s' are represented by a maximum amplitude, which leads to a bright color, for example, red. Figure 7a shows the general approach of amplitude modulation.



(a) Amplitude modulation changes the carrier signal by changing the amplitude. Through the second waveform, the carrier signal gets modulated. The result is shown in the third waveform[19].



(b) Frequency modulation changes the carrier signal by changing the frequency. Through the second waveform, the carrier signal gets modulated. The result is shown in the third waveform[19].

Figure 7: Difference between frequency and amplitude modulation

To transmit more than one bit at a time, the amplitude range needs to be divided into ranges. In order to send two bits at a time, the amplitude needs to be divided into (2^2) sections. To send three bits at a time, it needs to have (2^3) sections and so on. The more the amplitude is divided, the higher the error rate, because of the noise which is caused by the channel and the compatibility of the transmitter and receiver. The principal source of noise are other light sources which interfere with the signal.

In angular modulation, the focus is on frequency modulation. Also called frequency shift keying (FSK), which changes the color of the LED. Therefore, the amplitude does not change while using the frequency modulation, as shown in Figure 7b. The step for sending more bits at a time is the same as in amplitude modulation.

Comparing amplitude modulation and frequency modulation, frequency modulation is not as vulnerable to noises as amplitude modulation which reduces the errors in the transmission. On the other hand, for the frequency modulation an RGB LED is needed. In case of amplitude modulation, it also works with a monochromatic LED.

4.2.2 Forward Error Correction

Forward error correction (FEC) is a technique which is used for noisy or unreliable channels to control errors during data transmission. The sender adds

redundant information to the data to transmit. The redundant information makes it possible to reconstruct the data if some of it is corrupted or lost. In this project we need FEC because we do not want to rely on a feedback channel to let the transmitter know which packets need to be retransmitted. We do not want to communicate through the Wi-Fi channel during the transmission, therefore, the transmitter keeps sending packets until the transmission is completed. Relying on a FEC algorithm makes the design simpler and slightly more flexible. One widely used FEC algorithm is RaptorQ. RaptorQ is a fountain code that works with linear time in encoding and decoding [21]. Fountain code is a coding technique which can produce a potentially limitless sequence of packets in order to recover from packet loss. The encoder can generate these packets on the fly and the decoder can decode the data once a sufficient number of packets has been received. Regardless of which packets in the transmitted sequence have been received. RaptorQ code differentiates between two types of packets, source and repair packet. A source packet contains a portion of the original data and the repair packet contains parity bits that are used to recover from loss of source packets. RaptorQ can successfully decode the source data with a high probability if the number of packets received equals the number of source packets, regardless if those packets are source or repair packets. If all source packets are received correctly, the decoder is always able to decode the data. The number of repair packets determines the probability with which the original data is decoded correctly [16]. The characteristics of RaptorQ, linear en- and decoding time and a nearly limitless number of packets, make it a good choice for this project. This system needs error detection as well, e.g., cyclic redundancy check (CRC) in order to protect data from transmission errors.

The probabilities are mentioned in the official technical overview by Qualcomm:

- 99% for k encoding packets
- 99.99% for $k + 1$ encoding packets
- 99.9999% for $k + 2$ encoding packets,

where k is the number of source packets. It is also mentioned, that the recovery probabilities do not vary across the range of possible numbers of source packets, their size or the losses of the encoding packets [21]. Every RaptorQ packet contains a source block number, a packet number, and the number of bytes which are used as payload. This means, the first two bytes of a RaptorQ packet are not used for data but in order to reassemble the right data together for the decoder. The first byte contains both the source block number and the packet

number. The second byte contains the number of payload bytes. Figure 8 shows the procedure of encoding, sending, and decoding.

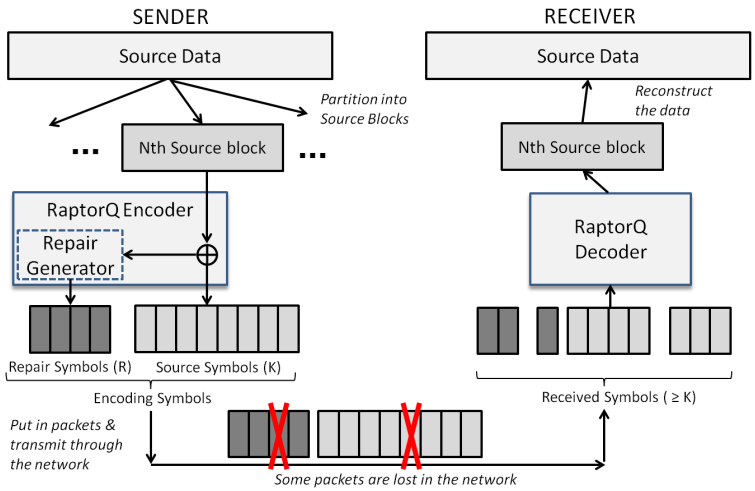


Figure 8: The source data is partitioned into different source blocks. The RaptorQ encoder generates for each source block source and repair symbols. With these the RaptorQ decoder can rebuild the source block data and therefore the source data [2].

5 Related Work

This section describes prior research on VLC, as well as the differences to our approach. Furthermore, it mentions other current state of the art authentication methods.

5.1 Visible Light Communication

One of the biggest challenges in VLC is the variety of transmitters and receivers. In transmitters, e.g., LEDs vary in their light calibration, which can cause a high bit error rate [1]. Receivers use different built-in sensors which affect the color detection and also have different frame rates [9], e.g., smartphones use different built-in cameras.

Prior research focuses on approaches where the communication is not visible to the human eye [11, 7, 13]. Because the modulation frequency of the LED exceeds human perception limits. To ensure that the blinking sequence is imperceptible for humans, additional white symbols are sent. The project DisCo [11] changes the brightness of displays so fast that humans cannot recognize it, this is then received by the rolling shutter sensor of the receiver.

VLC research also focus on high bandwidth [9, 30], upto 5.2 Kbps. Therefore, the transmission distance is lower than needed for the purpose of this thesis. The high-bandwidth approach in [30] covers a distance of 1 m, which is not enough for ceiling-mounted IoT devices. In this project, we do not need high bandwidth, because it is a system designed for authentication and not for communication.

As stated in [13], VLC is secure to MITM attacks because light does not penetrate walls. That means, an attacker who is in the line of sight is able to receive the signals which are transmitted. Therefore, the transmitted data is only public information, like public keys. On the other hand, if someone tries to interfere and manipulate the communication, it is seen by the user. That is one of the reason why we want to use visible light and do not want to cover it up, so it stays visible for the user and is perceptible, not like in the other approaches mentioned previously.

5.2 Authentication of IoT Devices

The following section describes different initial authentication methods used for IoT devices.

5.2.1 Wi-Fi Protected Setup

Wi-Fi Protected Setup (WPS) is an architecture for IEEE 802.11 networks designed to simplify the network configuration for users [29]. WPS uses a cryptographic protocol to establish a secure authenticated channel between the device and a designated network node, usually the access point (AP).

The protocol starts with a Diffie-Hellman key agreement. The key exchange takes place between the joining device and a WPS registrar. The result of the key agreement is mutually authenticated with a shared secret. The secret is a PIN which can be either statically or dynamically generated. A statically generated PIN can be printed on the device's label. A dynamically generated PIN must be shown to the user via some kind of user interface. WPS also offers the push button configuration(PBC). The user has to push a button on the registrar and on the device within a short period of time, known as the walking time. The PBC method serves as a proof that the user has physical access to both devices and that proof is then used for mutual authentication. Other possible WPS authentication methods are USB flash drive and NFC. The USB and NFC methods are not widely implemented. The USB flash drive is used to authenticate the device because only the user has access to this USB flash drive during the authentication. The NFC method is a proximity based solution where the IoT device needs to be close to the designated network node. All these methods rely on an MITM-resistant channel to authenticate the device. A channel is MITM-resistant if it is hard for an attacker to modify the messages in transit over such channel. That is, the case for both, USB and NFC, because both devices need to be in close proximity. The USB flash connects directly without a wireless connection.

WPS is based on cryptographic building blocks, but it is not secure in general [24]. The static PIN method is vulnerable to brute-force attacks [29]. The dynamic PIN method is rarely implemented, as well as NFC and USB.

5.2.2 ZigBee

ZigBee is an extension to the IEEE 802.15.4 standard, which is a transport protocol for wireless personal area networks.

The security of ZigBee relays on an advanced CCM mode and uses block cipher algorithm AES 128, which is reasonably secure. However, the specification requires that each device knows and accepts a fallback key which cannot be changed. Therefore, an attacker can easily break into a network, which makes it highly vulnerable [31].

5.2.3 MITM-resistant Out-Of-Band channel

There are many secure out-of-Band (OOB) channels which are resistant to MITM attacks. For example, there are methods like button press in WPS [29], physical contact [17], visual [18, 12], or audio [26]. Visual methods includes bar- or QR-codes. Such OOB channel can be used to exchange signatures of the Diffie-Hellman public keys. In all cases, these OOB channels are used to authenticate a key exchange because the primary communication channel is vulnerable to MITM attacks. Unlike the methods described before, authentication based on a VLC channel can be used with devices that are not physically reachable, e.g., a device mounted under the ceiling. Even though the device is not in close range or proximity through the VLC channel it remains secure. The QR code method is not suitable for such scenarios because the device might be too far away to read the code. The lack of display may not allow displaying the QR code visibly. Therefore, we are implementing the VLC channel approach.

6 System Architecture

Now we describe the basic architecture of the system. Figure 9 shows how the communication between the transmitter, for example, a smoke detector or a similar IoT device, and the receiver, a smartphone with a custom application. If the IoT device is not connected to any Wi-Fi network, the transmitter can connect to it via Wi-Fi peer-to-peer (P2P). The Wi-Fi P2P channel is generally insecure and vulnerable to MITM-attacks [23]. The receiver uses a visible light channel to authenticate the IoT device. The visible light channel is highly resistant to MITM attacks.

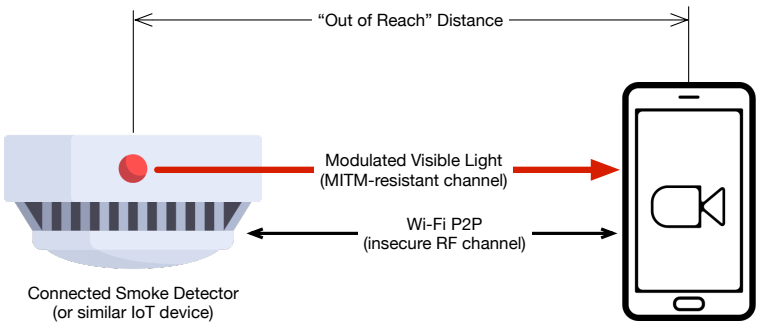


Figure 9: The basic architecture of the project. A remote IoT device with a LED and a smartphone with a camera. The two devices use a Wi-Fi P2P connection to start the communication. The connection is authenticate through the MITM-resistant VLC channel.

The smartphone connects to the IoT device over Wi-Fi P2P and establishes a TLS connection [25]. If the smartphone can verify the certificate, e.g., using a CA, it can authenticate the IoT device. If the smartphone cannot verify the certificate, because it is self-signed, the smartphone authenticates the certificate via the visible light channel. The smartphone app initiates transmission via the transmitter's HTTP API. The user then points the smartphone's camera towards the blinking LED and keeps the camera pointed there until enough data is received. The transmitter transmits the SHA-256 fingerprint of its TLS certificate, a hash, via the visible light channel. To authenticate the IoT device, the smartphone app compares the fingerprint received over the visible light channel with the fingerprint of the certificate received over the TLS connection. If they match, the IoT device is authenticated and the smartphone app stores the certificate permanently in a database for the IoT device. In case the fingerprint does not match the certificate, the smartphone shows a notification and the IoT device is not the one the smartphone is connected to, therefore a MITM

attack was recognized. A MITM attack is one possible explanation when the two SHA-256 certificates do not match, but it also can occur due to transmission errors.

Figure 10 shows the overall architecture of the project including the transmitter and the receiver parts.

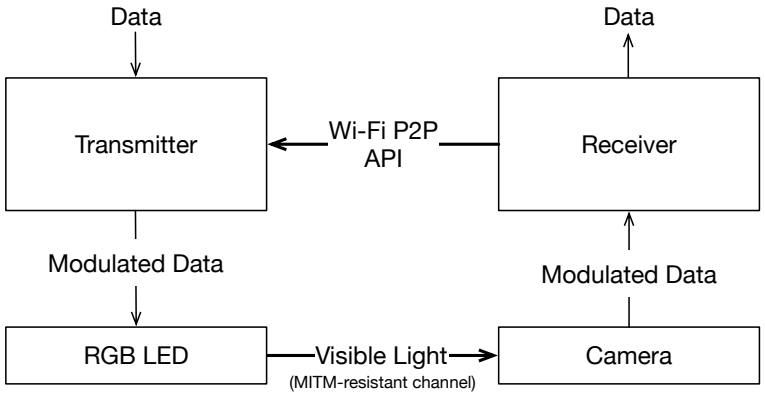


Figure 10: The basic architecture of the system. The transmitter modulates the data in order to send it out over the VLC channel through the RGB LED. The receiver camera receives incoming light and demodulates it. The communication to start a transmission is done over the HTTP API provided by the transmitter.

Figure 10 shows that the transmitter encodes and modulates the data. The transmitter transmits the data by modulating the light of an RGB LED. The VLC channel is prone to noise. The receiver is implemented as an application running on a smartphone with a camera pointed at the LED. The smartphone application uses computer vision algorithms to demodulate and decode the data. The smartphone application communicates with the transmitter via an HTTP API over a Wi-Fi P2P connection. The API can be used to control the transmission. Section 7 describes the transmitter design and how it is implemented more detailed. Alexander Linssen describes in his master thesis [14] the receiver in detail.

6.1 Transmitter API

The transmitter exposes an unsecured hypertext transfer protocol (HTTP) API via the Wi-Fi P2P connection. The smartphone application uses the HTTP API to start and stop the transmission, as well as to configure its parameters. To control transmission, the smartphone application sends an HTTP POST request with a JSON body to the transmitter. The transmitter extracts the

information from the JSON document and starts the transmission task. As a security feature and to avoid errors, other requests are not accepted while the task is running. Also, only the client which started the transmission can stop it. After a certain period of time, the transmission stops automatically in order to prevent a denial of service. To ensure that only the client which started the transmission can stop it, a unique transmission ID is the response to the request that starts the transmission. To stop the transmission previous to the timeout, the transmitter requires a transmission ID.

6.1.1 API Requests

The API requires a specific JSON document from the client to work correctly. There are three different request. The calibration requests, which turns on the LED to a specific color for a certain duration. The transmission request starts the transmission with different values. Also, the off request is implemented. The calibration request "x.x.x.x/calibration" expects a JSON document of the following form:

```
{
        "duration":1,
        "hueValue":[280,..],
        "brightness":100
}
```

Listing 1: JSON document to calibrate the LED

The duration attribute controls the maximum duration of the transmission. The transmission will be automatically turned off after this time. Hue is the color of the LED in the HSV/HSB scheme (value between 0 and 360). We use saturation and brightness at maximum as default. The transmission request "x.x.x.x/transmit" expects a JSON document of the following form:

```
{
        "FPS":30,
        "timeout":60,
        "modulator": "fskX"
}
```

Listing 2: JSON document to start the transmission

The options for the modulator are fsk2, fsk4 and fsk8. The JSON document represents the configuration of the smartphone. The timeout is the value in seconds that limits how long the transmission lasts. The transmitter includes a JSON document in the response which includes a unique, randomly generated

identifier of the transmission session. The receiver must include the transmission session identifier in any request to modify or terminate the transmission session. This is to ensure that only the receiver who started the transmission can stop it in scenarios where multiple receivers are connected to the same IoT device simultaneously.
Therefore, the off request "x.x.x.x/off" expects a JSON document of the following form:

```
{
        "tID": *** #TransmissionID
}
```

Listing 3: JSON document to turn off the transmission

7 Transmitter

This section describes the transmitter design and implementation, as well as the challenges which occurred while using a Raspberry Pi as the transmitter, and how they were solved.

7.1 Implementation Challenges

We faced some non-trivial challenges while implementing the transmitter. The first challenge we faced was the possibility to precisely control the color of the LED. In general, the Linux operating system running on the Raspberry Pi is not real-time capable. That means it is hard to precisely control PWM on general purpose input/output (GPIO) pins. The reason for that is that an interrupt or a context switch would generate unacceptable delays. There are two hardware PWM pins on the Raspberry Pi which we can use at the same time, but in order to control an RGB LED we need one more. The Raspberry Pi has four PWM pins, but they are paired in twos. That means, two of them get the same output, which is not suitable for our case. We use the PiGPIO library which provides hardware-assisted (DMA) precise PWM on any GPIO pin [28]. As illustrated in Figure 11, the pigpio library is a solution between hardware and software. It uses the DMA controller to write data to the GPIO pins and therefore, it improves timing accuracy.

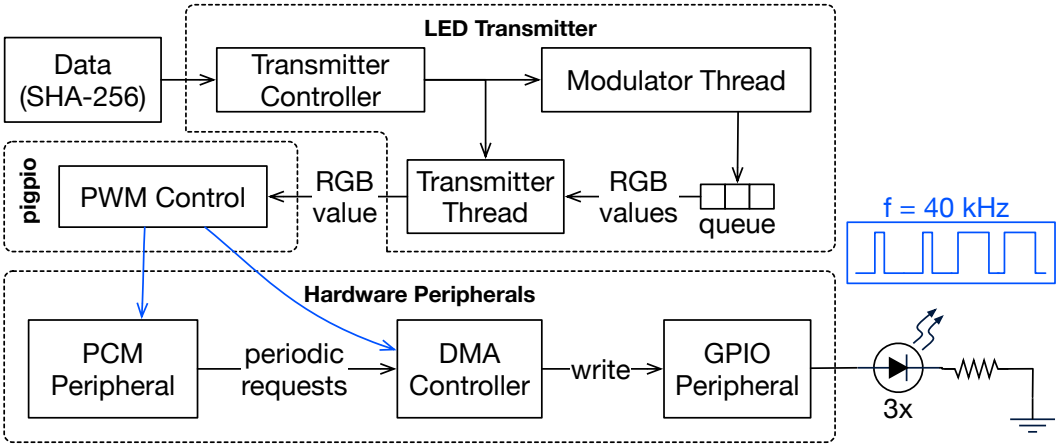


Figure 11: For a precise timing the transmitter has two different threads. The Modulator thread generates the color values and the Transmitter thread uses busy waiting to send these color values out at an accurate time. It also shows that the PiGpio library is a hardware and software solution. It loads the data directly in the DMA controller.

Precise control over transmission intervals was another implementation challenge. Standard Java APIs do not provide sufficient level of accuracy. To synchronize our transmission intervals, we use a combination of regular Java sleep and busy waiting. Therefore, the process does not sleep the certain time period but sleeps a little bit less than that. This is to guarantee that the sleep function never returns after the deadline, but shortly before the deadline. For the remainder of time, we then use busy waiting to make the wait interval as accurate as possible.

7.2 Transmitter Design and Implementation

We now describe the implementation of the data modifying part of the transmitter and how it is sent out using a LED. It also describes the threading approach and the API. Figure 12 illustrates the design and main building blocks of the transmitter.

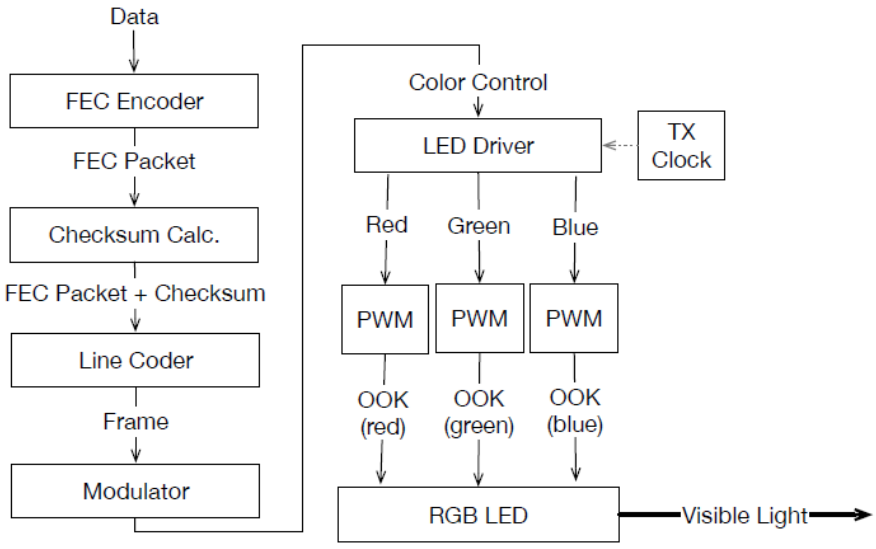


Figure 12: Different steps are performed to modulate the data and to send it out using the VLC channel.

7.2.1 Threading

The transmitter is designed as a multi-threaded application where one thread implements a tight transmission control loop and another thread performs data encoding and modulation in the background. This threading approach is needed in Java but not necessary in other programming languages.

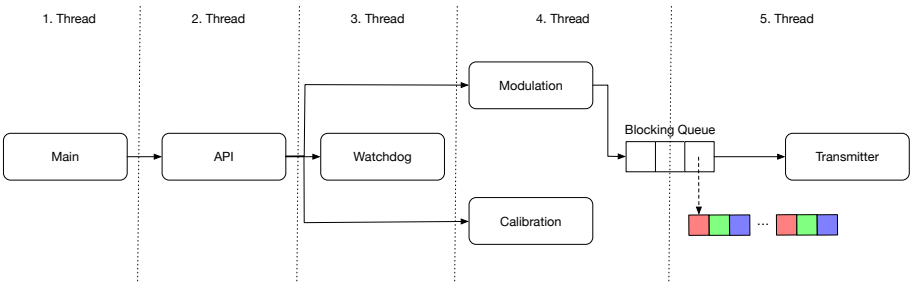


Figure 13: The different threads which run at the same time. The blocking queue between the Modulator thread and Transmitter thread is need for the precise timing of the color control. The Modulator thread modulates the data and the Transmitter thread uses busy waiting to time the LED control accurately.

Figure 13 illustrates that there are up to five threads running at the same time. The first thread is the Main thread, which configures the Raspberry Pi to use the correct LED and pins, e.g., in our case we have an RGB LED which uses the pins 22, 27, 17 for red, green, and blue. This needs to be set in a configuration file. The Main thread also configures and starts the API thread. The API thread handles the input from the POST requests for transmission, calibration or to turn them off. This thread also starts, depending on the request, either the Modulator thread or the Calibration thread, but either way it starts the Watchdog thread. The Watchdog thread takes care of the timeout in order to stop either the Modulator or Calibration thread. The Calibration thread uses the input from the API thread to show the requested colors for the specific amount of time. The Modulator thread starts the Transmitter thread and modifies the data and puts them in a queue for the Transmitter thread. Figure 13 shows, that these values are a list of RGB colors. The Transmitter thread sends the data from the queue out by controlling the LED, and using busy waiting for precise timing.

7.2.2 Data Encoding

As the first step the data is split into different packets using RaptorQ [16]. This is needed in order deal with packet loss and to restore the data in a linear time if some of the packets are lost during the transmission because of noise. The RaptorQ block generates packets on the fly as long as they are needed to complete the transmission, since it can basically generate a nearly infinite number of packets. The first packets are source packets, which means the data itself. If those were not received correctly the RaptorQ block keeps generating new packets, which are repair packets. These do not include the data itself but parity bits. Therefore, not the whole sequence has to be sent again. The

receiver needs to receive the number of packets that is the same or larger as the number of source packets and it does not matter whether the received packets are source or repair data. This is the first step in modifying the data and important for recovering and fast transmission because there is no need to use the feedback channel, Wi-Fi, and request for retransmitting packets. In this case the transmitted packets are relatively short, it contains four byte of payload, therefore modified RaptorQ packets are used. For that reason, we do not use the standard header included in RaptorQ packets. Instead, we encode the packet sequence number in a variable number of bits. The sequence number is encoded in the last bits of the packet and is only as big as necessary, e.g., the sequence number is '2' only takes two bits. Figure 14 illustrates the frame which contains the modified RaptorQ packet.

The RaptorQ decodes data whether it is correct or corrupted. Therefore, we have to check the data for its validity before passing it to the decoder. The transmitter calculates a checksum for each RaptorQ packet. We use CRC8 and add the 8 bit checksum to the packet. The receiver needs the checksum to ensure that the packet is correct. The transmitter adds the checksum in front of the RaptorQ packet. The receiver is aware of the checksum and after calculating the checksum over the data it received, the receiver decides whether to drop the data or pass them to the decoder.

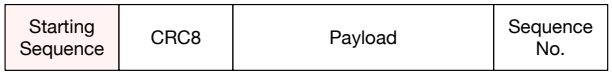


Figure 14: Modified data ready to be transmitted

Now the packet includes the CRC8 checksum and the RaptorQ packet, which is demonstrated in Figure 14.

7.2.3 Preparing Frame for Transmission

In order to transmit the frame it needs to be encoded so that the receiver can correctly identify the start and end of the frame. Our line coding technique is similar to the high level data link control (HDLC) protocol [5]. We prefix each frame with a frame marker sequence. The frame marker sequence is three symbols long, "132" which means red, blue, green or "011110". '00' = 0, LED off; '01' = 1, red; '10' = 2, green; '11' = 3, blue. We encode the data, so that the frame marker sequence never appears in the frame. Therefore, we use symbol stuffing. Figure 15 shows the state machine for symbol stuffing.

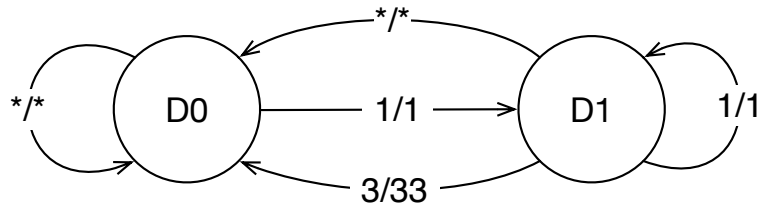


Figure 15: Symbol stuffing for 4-level FSK is described by this state machine

This is the state machine for the 4-level FSK modulation. The bit combinations are represented with symbols, their integer value.

Figure 15 illustrates, that the state machine stuffs an additional symbol as soon as the first and second symbol are the same as the starting sequence. This data is now modified and ready to be transmitted, as shown in Figure 14. Depending on the symbol stuffing and the sequence number of the RaptorQ packet the minimum number of bits which are needed to complete a transmission are 392 bits, this value does not include any symbol stuffing. The maximum amount cannot be declared correctly because of the symbol stuffing but it is around 430 and 450 bits.

7.2.4 API

Below, we describes the use for communication between the transmitter and the receiver. The API thread waits until someone is connected to the transmitter and request a transmission. As described in the previous section, the transmission request sends parameters with the request. One of them is the timeout, in order to have a transmission and still a precise timeout the Watchdog thread starts during the accepted request and sleeps for the timeout time and stops the transmission thread when it wakes up, if it has not been shutdown before by the user.

7.2.5 Transmission

The transmitter is implemented such that it can work with different kinds of LEDs, monochromatic or tri-color, as well as PWM or on-off. Although our tests focus on the tricolor LED, the software to use those is implemented as well. This implementation cannot only use different LEDs, also a variety of modulations. The prototype implementation provides 2-/4-/8-level FSK and OOK. Each modulation type maps bits (symbols) to hue values. The transmitter converts the hue values in RGB values to turn on the correct color of the LED and then writes these values to the GPIO pins. The maximum current flow of the LED is about 20 mA. Therefore, the value for a color 0 the duty circle of

the PWM is 0% and there is no current flow through that LED. On the other hand if the value has its maximum at 255, this results in a duty circle of 100% and sets the current to 20 mA.

The transmitter uses two threads. One thread implements a tight transmission control loop. The other thread generates data in the background, i.e., it performs data encoding, error correction, and modulation. The two threads communicate via a shared blocking queue.

8 Evaluation

This section describes the experimental setup. We also analyze the collected data.

8.1 Experimental Setup

The setup includes hardware and operating system specifications for the experiments. The transmitters are listed in table 1. We use a tri-color RGB LED in a diffused package.

Table 1: Transmitters being used in the experiments

ID	Device	OS	Kernel	LED
1	Raspberry Pi 3 Model B, revision 1.2	Raspbian (Debian 9.4)	Linux 4.14.52-v7+	Cathode, RGB, 5 mm diameter
2	Raspberry Pi 3 Model B, revision 1.2	Raspbian (Debian 9.4)	Linux 4.14.50-v7+	Anode, RGB, 10 mm diameter

On the receiver side, we use different Android operated smartphones:

Table 2: Receivers being used in the experiments

ID	Device	OS	Frame rate
1	Huawei Honor 7X	Android version 7.0	30 fps
2	Samsung Galaxy S7	Android version 8.0	30 fps
3	OnePlus 3T	Android version 8.0	60 fps

8.2 Laboratory Experiment I - Color Detection

This experiment gives a better overview over the color detection. The smartphones are fixed on a tripod and the camera points towards the transmitter's LED. The transmitter sends out every huer value to the receiver, that are 360 different colors, 0-359. For this experiment the transmitter was modified. A new tab was implemented to the API, "/experiment", which also needs a JSON document like the others, which only contains a duration time. This duration turns on the LED for its value in seconds. After that, the transmitter shows the next hue value. The receiver saves the hue value with a timestamp, the brightness and also camera configurations like exposure time, exposure compensation, ISO, auto white balancing, focus distance and zoom. In order to get a wide vary of results we used three different phones, Samsung Galaxy S7, Honor 7x and

OnePlus 3T. The purpose of this experiment is to figure out how many colors can be recognized reliably with different phones and under different conditions. Therefore, it is able to see if 4- or 8-level FSK are reliable under most conditions or not. In the following the data was collected for each hue value for 1s, in particular, for the Samsung Galaxy S7 and the Honor 7x about 30 samples per hue value because they run on 30 fps and for the OnePlus 3T 60 samples.

In the following there are nine figures which show the results of the tests. The blue line displays the hue values sent out by the transmitter and the other line/lines displays the received hue values on the different smartphones. The distance of the test are 2 m and 4 m, the transmitter are the ones in Table 1.

Figure 16 shows the result for the Honor 7x smartphone using the first transmitter and a distance of 2 m. In the beginning there is a big drop from 358 to 1. The reason for that is that, a circle represents the hue values, this is illustrated in Figure 5. Therefore, 360 equals 0 and even though in the graph it seems to be off it actually is accurate. The smartphone receives the first 15 hue values accurately, but after the green LED shines at a maximum, values between 60 to 180, the distance between the transmitted and received hue values gets bigger. For the yellowish colors the recognition the Honor 7x is not accurate.The Honor 7x did not recognize the yellowish colors well. Sometimes it recognized nothing. That is the reason for the drop between 50 and 70. The primary colors 0 for red and 240 for blue seem more accurate than mixed colors. Also, green at 120 is more accurate than the surrounding colors.

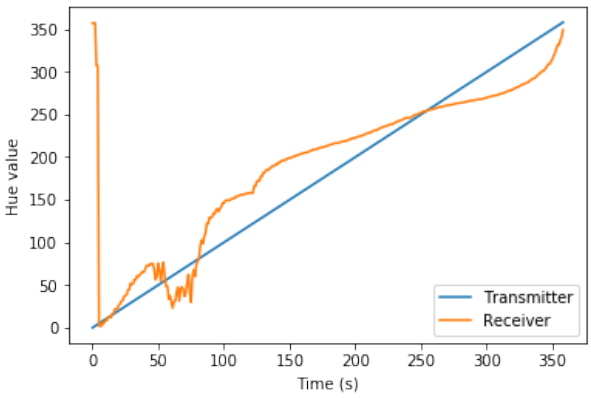
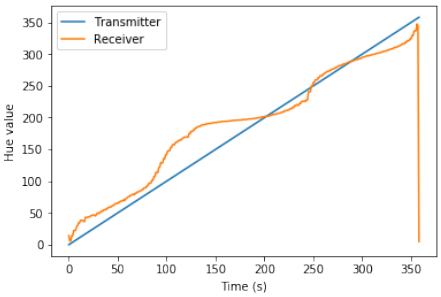
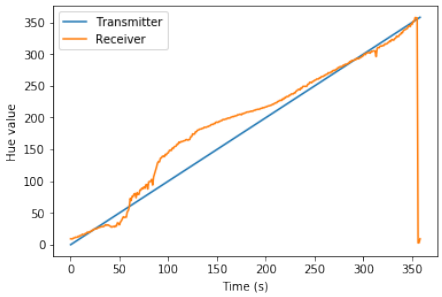


Figure 16: Honor 7x color detection with cathode at 2 m distance



(a) This are the result for the OnePlus 3T smartphone. The drop at the end of the orange graph has the same reason as for the Honor 7x in the beginning. The OnePlus 3T is accurate around the yellow colors, but as well as the Honor 7x it also recognizes the green colors far off than other colors. Red and blue are recognized accurate.



(b) This are the result for the Samsung Galaxy S7 smartphone. This smartphone behaves similar to the OnePlus 3T at the end of the graph. It also has the same anomaly as the other phones around the greenish colors, otherwise it is accurate on the colors.

Figure 17: OnePlus 3T and Samsung Galaxy S7 color detection with cathode on 2 m distance

The next results are made with the same smartphone and the same distance but with the second transmitter in table 1.

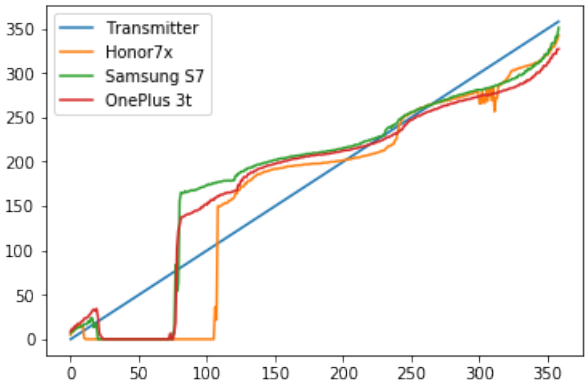


Figure 18: This are result combined in one graph in order to show the similarity between each graph. For the yellowish colors the smartphones do not recognize any color, OnePlus 3T and Samsung Galaxy S7 start recognizing colors earlier than Honor 7x. The following greenish colors are off compared to the transmitter. The blueish colors from 200 to 270 more precisely but at 310 the Honor 7x had problem recognizing the color.

These experiments show that the second transmitter is not as reliable as the

first one. The same distance and the same smartphones are used for this, but the results show significant differences between them. The cause of that could be the brightness of both LEDs. The LED in the second transmitter is four times larger than the of the first, therefore also the light is brighter.

We did the following experiment with the first transmitter and the same smartphones as before, but the distance is increased to 4 m.

Figure 19 illustrates the performance of the Honor 7x at a 4 m distance. The Honor 7x troubles recognizing colors between 20 and 120. Also, after 120 the color detection is not close to the expected value. At 200 the Honor 7x recognizes the color correctly but shifts of the expected value afterwards.

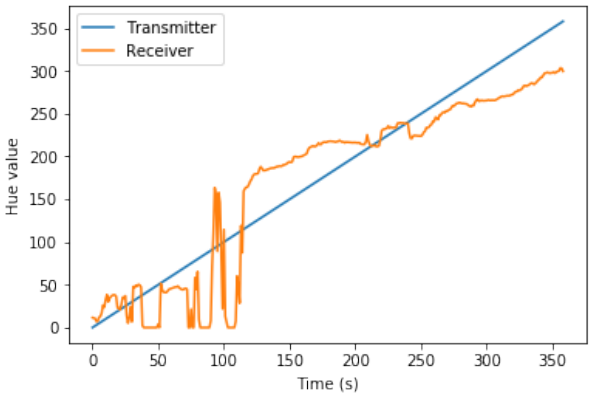
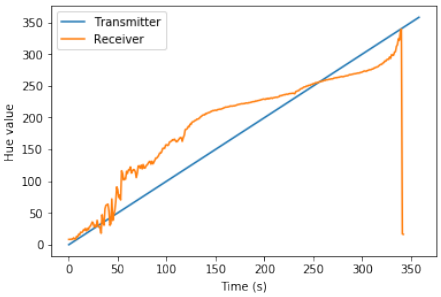
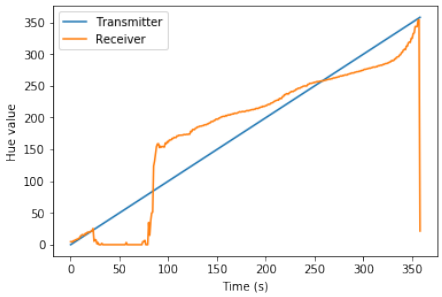


Figure 19: Honor 7x color detection with cathode at 4 m distance



(a) This are the colors received by the OnePlus 3T smartphone at 4 m distance. The color detection is good when it comes to the color red. For the yellowish part the OnePlus 3T troubles recognizing it but does it frequently. The rest is close to the previous test on closer range.



(b) The Samsung Galaxy S7 does not recognize the color yellow at a 4 m distance but otherwise it recognizes the color red and blue precise. The greenish colors are far off as in previous tests.

Figure 20: OnePlus 3T and Samsung Galaxy S7 color detection with cathode on 4 m distance

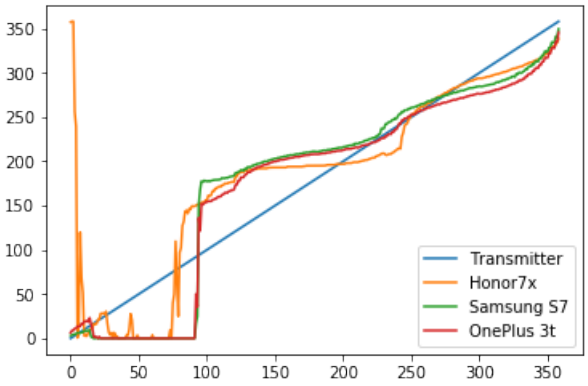


Figure 21: The color detection test results using the second transmitter at a 4 m distance. This graph looks similar to the result for the test on a shorter distance, although this time the Honor 7x started recognizing the yellowish colors earlier.

All of these twelve graphs have in common that the greenish values around 120 are not recognized precisely, regardless of the distance or transmitter. Also, the yellowish colors stand out. As for the second transmitter as well as for the longer distance yellow is a troubling color. Otherwise, the OnePlus 3T and Samsung Galaxy S7 recognize the colors precisely. The Honor 7x struggles more to recognize colors especially at a greater distance. It seems that the colors red,

green, and blue can be recognized reliable.

That the greenish colors are off seems to be a cause of auto white balancing in the camera of the smartphones [14], on the other hand it can be caused by the LEDs as well, the LED can be calibrated incorrectly and therefore, cause green to be more dominant. The struggling of the yellowish colors can be caused by the nature of this color, it seems to be more faded by using a LED and therefore the grayish color elimination on the receiver part can cause an unreliable detection of this color [14].

In order to solve these problems either the LED can be re-calibrated, this can help with the over detection of the greenish colors, or the threshold of the receiver application can be reconfigured.

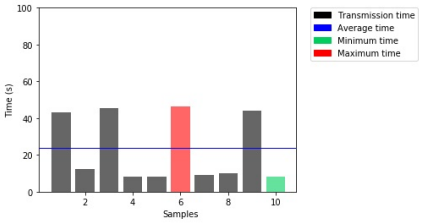
8.3 Laboratory Experiment II - Transmission Time

This experiment gives an overview of the time needed to successfully finish the transmission. The first performance test is performed while the camera is mounted on a tripod. Nevertheless, the distance, the transmitter and the smartphones used vary. The second test is done with two people who are not familiar with the project, in order to figure out how complicated the procedure is and how fast people can adapt to it. The transmitter fps rate is half of the receivers in order to sample twice. The blinking frequency is 30 fps for the Honor 7x and Samsung Galaxy S7 and 60 fps for the OnePlus 3T. The modulation for this experiment is 4-level FSK.

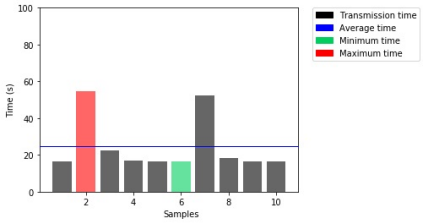
8.3.1 Stationary Performance Test

The following performance show the transmission time and how reliable the smartphones are using the different transmitters. In order to compare the results correctly it is necessary to mention that the OnePlus 3T operates at a frame rate twice as fast as the other smartphones.

Using the anode at a 2 m distance the results are as followed Figure 22b, Figure 22a, Figure 23:



(a) The OnePlus 3T performs different through the testing process. The transmission take 4x more than 43 seconds. Nevertheless, the other 6 times it takes about 10 seconds. Therefore, the average is 23.5 seconds.



(b) The Samsung Galaxy S7 performs more reliable than the OnePlus 3T using the anode LED at a 2 m distance. Only two times it needs about 52 seconds for the transmission to succeed. Otherwise, it takes about 17 seconds which leads to an average of 24.7 seconds.

Figure 22: OnePlus 3T and Samsung Galaxy S7 performance test with anode at 2 m distance

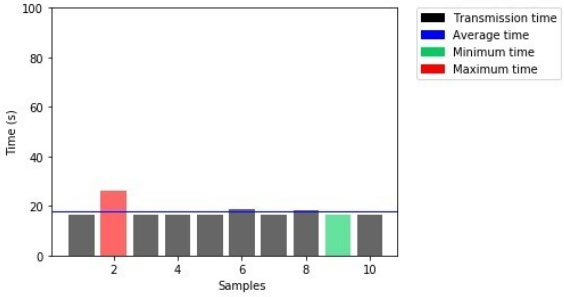
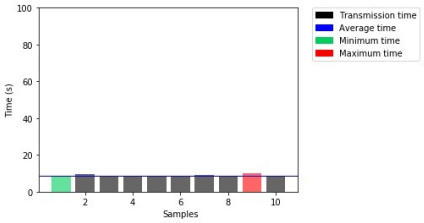
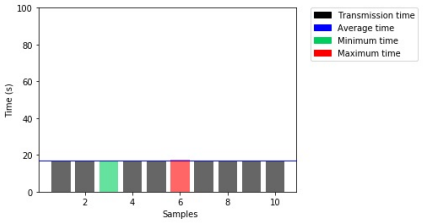


Figure 23: The results of the testing using the Honor 7x. The results show that the Honor 7x is reliable using the anode transmitter on at a 2 m range. The average of it is 17.8 seconds.

The next figures 24a, 24b, and 25 show the results of the test using the anode transmitter at a 4 m distance.



(a) The OnePlus 3T performs consistently good during the test, this leads to an average of 8.6 seconds



(b) This figure shows that also the Samsung Galaxy S7 performs reliable using this transmitter at a 4 m distance. The average is 17.2 seconds

Figure 24: OnePlus 3T and Samsung Galaxy S7 performance test with anode at 4 m distance

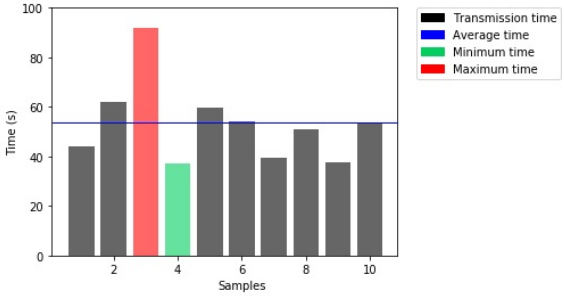
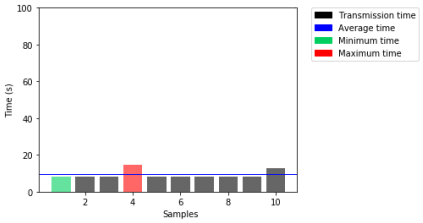
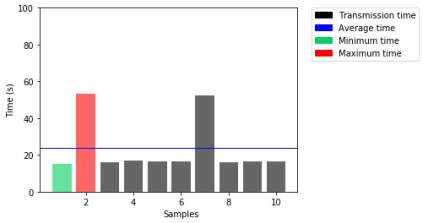


Figure 25: However compared to the other smartphones, the Honor 7x show a bad performance during this test. The fastest transmission is 37.3 seconds and the slowest takes 91.9 seconds, which leads to an average of 53 seconds.

Comparing the results shown for the second transmitter it is to notice that the OnePlus 3T and Samsung Galaxy S7 show similar results but the Honor 7x performs opposite to them. The following figures show the same test using the first transmitter.



(a) The OnePlus 3T performs constant through the test, which leads to an average of 9.3 seconds.



(b) The Samsung Galaxy S7 performance is nearly the same using the other transmitter, average of 23.6 seconds

Figure 26: OnePlus 3T and Samsung Galaxy S7 performance test with cathode at 2 m distance

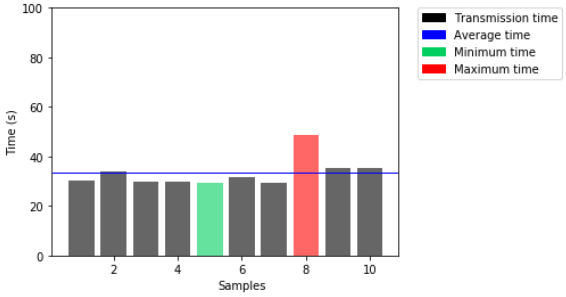
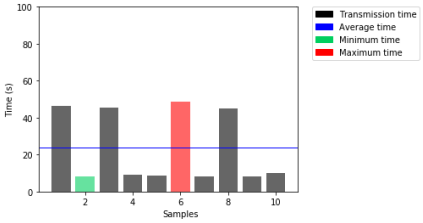
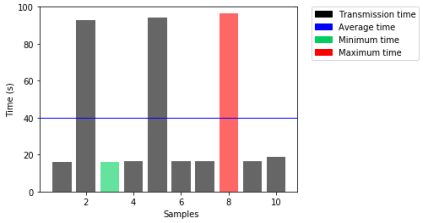


Figure 27: This are the Honor 7x test results. It is noticeable that every transmission has packet losses and therefore, the average of the Honor 7x is 33.4 seconds.

The next figures 28a, 28b, and 29 show the results of the test using the cathode transmitter at a 4 m distance.



(a) The OnePlus 3T performs inconsistent throughout the test. Similar to its performance using the other transmitter at 2 m distance.



(b) The Samsung Galaxy S7 performs different during the test. It has three high peeks. It took three times over 90 seconds to finish the transmission.

Figure 28: OnePlus 3T and Samsung Galaxy S7 performance test with cathode at 4 m distance

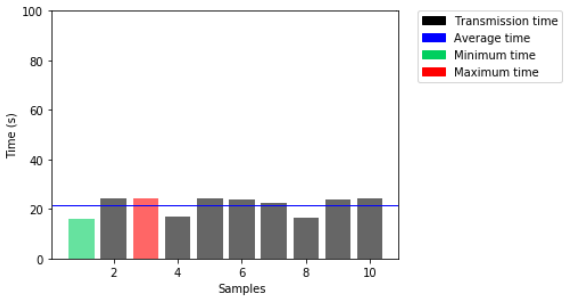


Figure 29: However compared to the other smartphones, the Honor 7x show a good performance which contain some packet loss but over all the average is 21.5 seconds.

The results are not consistent to a transmitter or a distance. However, the Samsung Galaxy S7 and the OnePlus 3T perform similar. It is recognizable that there are mostly two levels in the transmission time graphs. One where no packet loss happened and one where it happened. The odd part of it is that, the receiver recovers from packet loss mostly the same. If one transmission with packet loss took about 40 seconds, the next transmission with packet loss is more likely to take a similar amount of time to recover.

This can be caused by the camera interval, the fluctuation of camera frames, as well as the not yet implemented synchronization. It seems that the problem is a shift between the receiving frame interval compared to the LED on-time on the transmitter side.

8.3.2 Practical Performance Test

This test is done by two persons who are not familiar with the project. One uses the OnePlus 3T and the other the Samsung Galaxy S7 smartphone. The distance between the smartphone and the transmitter is about 2 m and it is handheld by the user. The user has to point the camera towards the LED and then pushes the start button, see user interface [14]. After the start button is pushed the time measurement starts, it ends as soon as a transmission has been completed.

The first test person uses the Samsung Galaxy S7 with a frame rate of 30 fps. The first transmission takes 55.7 seconds. During the transmission the user zooms in on the LED and aligns the auto exposure compensation in order to receive the data. Therefore, the camera is not always pointed towards the LED, which leads to frame loss. For the second transmission the user knows how to adjust the lighting conditions and which zoom level is necessary, which leads to a transmission time of 17.9 seconds.

The second test person uses the OnePlus 3T smartphone which runs at 60

fps. The first transmission time is 10.5 seconds, because the person does not change any parameters in order to receive the transmission, nevertheless the data is received with only one frame loss. The second transmission took 9.6 seconds.

The experiment shows that people can adjust fast to the user interface and authentication method. It also shows that the transmission time is not too long for a user to point the camera towards the LED, as well as some devices do not need to adjust as much to certain conditions as others.

9 Results and Analysis

This section provides answers to the questions which guided the thesis, as well as that the requirements are fulfilled.

One of our original design goals was to make the transmitter easy to implement using a minimum amount of additional hardware. We want to support a wide variety of platforms and make sure that the transmitter does not increase the costs of the IoT device significantly. In this case, the hardware for the transmitter is an RGB LED, which is connected to GPIO pins. That means, besides the LED, pins on the transmitting device, and some resistors there is no additional hardware needed. The transmitter and evaluation Sections 7, 8 show that the transmitter design is usable. Therefore, it is possible to design a usable transmitter with a minimum of hardware. The whole design is also easy from the form factor point of view because most devices already have some sort of status indicator LED. That status indicator LED is easy to swap for a tri-color LED. In general, almost all IoT device can be equipped with a LED.

The evaluation Section 8 shows that the transmitter can be easily ported to other devices. If the target device runs Java, porting is trivial, since our prototype application is written in Java. The only changes one may need to perform would be to update the Java native libraries, replacing PiGPIO with something specific for the target platform. The design of the transmitter is described in the thesis in a sufficient detail and it should be fairly straightforward to re-implement the transmitter in a different programming language. The algorithms for FEC and error detection are publicly documented.

The reliability of the VLC channel and the speed has been described in the evaluation section 8. The VLC channel is noisy. We focus on 4-level FSK, so we do not require LED calibration. Doing 8-level FSK reliably would require some sort of calibration.

The data received in the experiment shows that the average transmission time is over 30 seconds, see figure 25. The lack of synchronization explains variability in the transmission times. When properly aligned, the transmission times are short. When misaligned, transmission times are long. The maximum camera frame rate supported by OnePlus 3T is 60 frames per second. Other smartphones support only 30 frames per second. Thus, the transmission times with OnePlus 3T are typically half the transmission times of the other smartphones. However, it seems that the color detection with the Honor 7x is not as reliable as the other smartphones used in the tests. This causes the average higher transmission time. The second test in Section 8.3.2 shows, that the user can compensate the lighting condition, by manually changing exposure time compensation and the zoom level. Making the transmission more reliable and

the user helps to make sure that colors are not washed out. This is frequently a problem in dim areas or with dark background. The zoom level guarantees that there are enough pixels to calculate the average hue value. The VLC channel is reliable to a certain degree of color choice and it is fast enough as well. The bottleneck is the lack of camera control provided by the Android Camera2 API. In particular, it is not possible to receive frames at a specified time. The capture rate is controlled by the camera hardware and varies from time to time, depending on the camera sensors and smartphone hardware capabilities.

As described the requirements are fulfilled by this approach.

10 Security Considerations

In order to figure out how secure this approach is we come up with some scenarios for a threat model. In the beginning we talk about scenarios this approach is able to detect and then about those we are not able to prevent.

This is the authentication process which should prevent MITM attacks. An MITM attack is, in this case, an attack where the attacker first tries to hijack the Wi-Fi communication and then tricks the user to send him/her the network credentials. That means the attacker uses a device which pretends to be the IoT device. The attacker's device has already established a TLS connection to the IoT device. When the user unintentionally connects to the attacker's device, he or she obtains the public certificate from the attacker's device to establish a TLS connection to this device. The traffic between the user and the IoT device is forwarded by the MITM. Without authentication the user does not recognize that he or she obtained the wrong certificate and would send the attacker the network credentials.

Message tampering is a common method to attack a system. For example, the attacker could use a more powerful light source to send their certificate over the ones of the IoT device. Also, additive color manipulation, for example, using a different light source or even infrared light can be message tampering. However, the VLC channels are hard to tamper with. If someone tries to intercept the communication he has to be in physically sight line. The user is in control for the entire communication session, that means, the user start the transmission, the blinking, and also points the camera towards the LED. Other blinking light sources are easy to spot or to eliminate because the they are seen by the user. An offline attack can also take place. that is an attack where the attacker does not interfere with the communication, but tries to gain valuable information out of recording the authentication procedure. This is prevented by only sending public information via the VLC channel.

Denial of service is something which this approach is possible not able to prevent. E.g., if the attack damages the LED, which then cannot send out light anymore. Social engineering is also hard to handle. Someone could trick the user to authenticate a wrong, corrupted, device or to bypass the authentication. Another attack is that someone uses a cloned application, in particular, an application which looks identical to ours but does not match the certificates but accepts every authentication. Compromised device are also a big problem, not only compromised IoT devices but also compromised smartphones. Our method is not able to detect those.

11 Conclusion

Secure authentication of remote IoT devices using visible light communication is a comfortable and reliable new way to authenticate devices. The source code is published on GitHub [15].

This authentication method enables certificate rotation. This allows the user to reset the device in a way that no previous owner can connect to it. Certificate rotation means that the device can regenerate a certificate and does not have a prefabricated one. This rotation can take place during the factory reset.

There is no need for any special equipment. The user can use their smartphones to authenticate a device and an RGB LED is low cost as well. Since most IoT device already use a LED as a status indicator, it can be replaced with an RGB LED. LEDs require very little space on a device, which allows it to be equipped on nearly every IoT device.

VLC makes this method not only comfortable for the user but also secures the authentication process. For the user it is comfortable because it works over distance, so there is no need to get close to the device. Since VLC works with the line of sight, it is difficult to temper any message without the user noticing it.

11.1 Limitations

The transmission speed is limited by the smartphone camera sensor capability the user is using. As shown in the experiments, the OnePlus 3T running on 60 fps can be twice as fast as the Samsung Galaxy S7 running on 30 fps.

Another limitation of this project is the processing time of the smartphone. Its hardware determines the process delay. Which means that even if all packets are received the smartphone needs to process them and the user has to wait for it.

This authentication method is limited by the line of sight to the LED. Visible light does not penetrate wall and can therefore not be received outside the room it is used in.

The ambient light affects the transmission reliability. If the IoT device is surrounded by bright light or a dark background this can influence the transmission.

The color of the IoT device is supposed to be a grayish. If a color is already detected, even though the LED is off, the transmission will not work.

11.2 Implications for Future Research

Supporting a high modulation scheme is a good way to improve this project. For example, the use of 8-level FSK or a combination of amplitude modulation and FSK. In order to archive a more reliable transmission an automatic calibration can be implemented. This means that the receiving part calibrates itself by recognizing the ambient light and the known colors during the starting sequence detection.

Transmitter-receiver synchronization can be implemented in order to archive a faster recovering of a shifted transmitter and receiver interval.

To authenticate a device faster more area of interest can be implemented in the receiver application. This could be used on IoT devices which have more than one LED.

A more efficient way off data encoding can be used. The current bit stuffing approach is fairly easy and generates unnecessary overhead.

In order to support devices with a low CPU capability a pre-generated RaptorQ approach can be implemented. This means that the authentication secret would become static but would solve the problem of devices not being able to generate the RaptorQ packets. The RaptorQ packets a generated for a device and then stored in a file. Reading out the data of a file does not generate much work for the CPU.

References

[1] B. Bai, Q. He, and Z. Xu. "The color shift key modulation with non-uniform signaling for visible light communication." In: Aug. 2012. URL: https://ieeexplore.ieee.org/abstract/document/6316471/.

[2] *Diagram of RaptorQ*. Accessed: 07 August 2018. URL: https://github.com/openrq-team/OpenRQ/wiki/Concepts.

[3] *Electromagnetic Spectrum*. Accessed: 20 July 2018. URL: http://www.bigshotcamera.com/fun/buildables/colorwheel#01.

[4] *Electromagnetic Spectrum*. Accessed: 20 July 2018. URL: https://www.techopedia.com/definition/9756/data-transmission.

[5] Funk G. "Message Error Detecting Properties of HDLC Protocols." In: *IEEE Transactions on Communications* (Jan. 1982), pp. 252 –257. DOI: 10.1109/TCOM.1982.1095367. URL: https://ieeexplore.ieee.org/document/1095367/.

[6] "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016." In: (Feb. 2017). Accessed: 18 April 2018. URL: https://www.gartner.com/newsroom/id/3598917.

[7] L. Grobe, A. Paraskevopoulos, j. Hilt, D. Schulz, F. Lassak F. Hartlieb, C. Kottke, V. Jungnickel, and K. D. Langer. "High-Speed Visible Light Communication Systems." In: Dec. 2013. URL: https://ieeexplore.ieee.org/abstract/document/6685758/.

[8] *HSV color circle*. Accessed: 07 August 2018. URL: http://farm7.static.flickr.com/6091/6242569520_320557a30f_o.png.

[9] P. Hu, P. H. Pathak, X. Feng, H. Fu, and P. Mohapatra. "ColorBars: Increasing Data Rate of LED-to-Camera Communication using Color Shift Keying." In: Dec. 2015. URL: https://dl.acm.org/citation.cfm?id=2836097.

[10] N. A. Ibraheem, M. M. Hasan, R. Z. Khan, and P. K. Mishra. "Understanding Color Models: A Review." In: Jan. 2013. URL: http://www.jgrcs.info/index.php/jgrcs/article/view/587.

[11] K. Jo, M. Gupta, and S. K. Nayar. "DisCo: Display-Camera Communication Using Rolling Shutter Sensors." In: 2016. URL: http://www1.cs.columbia.edu/CAVE/publications/pdfs/Jo_TOG16.pdf.

[12] Tim Kindberg and Kan Zhang. "Secure spontaneous device association." In: *Ubicomp*. Vol. 2864. Springer. 2003, pp. 124–131.

[13] N. Kumar and N. R. Lourenco. "LED-based visible light communication system: a brief survey and investigation." In: 2010. URL: http://docsdrive.com/pdfs/medwelljournals/jeasci/2010/296-307.pdf.

[14] Alexander Linssen. *Secure Authentication of Remote IoT Devices Using Visible Light Communication; Receiver Design and Implementation.*

[15] Alexander Linssen, Hagen Odenthal, and Jan Janak. *Project Lighthouse.* Accessed: 17 August 2018. URL: https://github.com/columbia-irt/lighthouse.

[16] M Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder. *RaptorQ Forward Error Correction Scheme for Object Delivery.* Tech. rep. Aug. 2011. URL: https://www.rfc-editor.org/rfc/rfc6330.txt.

[17] Rene Mayrhofer and Hans Gellersen. "Shake well before use: Intuitive and secure pairing of mobile devices." In: *IEEE Transactions on Mobile Computing* 8.6 (2009), pp. 792–806.

[18] Jonathan M McCune, Adrian Perrig, and Michael K Reiter. "Seeing-is-believing: Using camera phones for human-verifiable authentication." In: *IEEE symposium on security and privacy.* IEEE. 2005, pp. 110–124.

[19] *Modulation and Different Types of Modulation.* Accessed: 04 May 2018. URL: https://www.electronicshub.org/modulation-and-different-types-of-modulation/.

[20] N. Pm and R. Dr. Manicka Chezian. "Various Colour Spaces and Colour Space Conversion Algorithms." In: Jan. 2013. URL: http://www.jgrcs.info/index.php/jgrcs/article/view/587.

[21] Qualcomm. *RaptorQ Technical Overview.* Tech. rep. 2010. URL: https://www.qualcomm.com/documents/raptorq-technical-overview.

[22] *RGB Color Cube.* Accessed: 07 August 2018. URL: https://hometheaterhifi.com/wp-content/uploads/2013/05/color-cube-medium-lg.jpg.

[23] A. Sanatina, S. Narain, and G. Noubir. "Vulnerabilities and Attacks on WiFi Networks." In: *International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 15 (2018)* (2018), pp. 12052–12054. URL: http://www.ripublication.com/ijaer18/ijaerv13n15_49.pdf.

[24] A. Sanatina, S. Narain, and G. Noubir. "Wireless spreading of WiFi APs infections using WPS flaws: An epidemiological and experimental study." In: Oct. 2013. URL: https://ieeexplore.ieee.org/document/6682757/.

[25] Axel Sikora. "Strong mutual authentication in a user-friendly way in EAP-TLS." In: Sept. 2007. URL: https://ieeexplore.ieee.org/abstract/document/4446137/.

[26] Claudio Soriente, Gene Tsudik, and Ersin Uzun. "HAPADEP: human-assisted pure audio device pairing." In: *Information Security Journal*. Springer-Verlag, 2008, pp. 385–400.

[27] *The Internet of Tools*. Accessed: 22 August 2018. URL: http://www.forge49.com/the-internet-of-tools/.

[28] *The pigpio library*. Accessed: 29 March 2018. URL: http://abyz.me.uk/rpi/pigpio/.

[29] Stefan Viehboeck. *Brute forcing Wi-Fi Protected Setup*. Tech. rep. Dec. 2011. URL: http://warxezz.free.fr/direct/PDFs/PIN_wps_viehboeck.pdf.

[30] Y. Yang and J. Luo. "Boosting the Throughput of LED-Camera VLC via Composite Light Emission." In: URL: https://dl.acm.org/citation.cfm?id=2836097.

[31] *zigbee alliance*. Accessed: 06 March 2018. URL: http://www.zigbee.org/zigbee-for-developers/zigbee-pro/.

Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden.

Ferner habe ich vom Merkblatt über die Verwendung von Masterabschlussarbeiten Kenntnis genommen und räume das einfache Nutzungsrecht an meiner Masterarbeit der Universität der Bundeswehr München ein.

(Hagen Odenthal)