Dominic Pirker BSc

Design and Implementation of a Global and Secured Drone Identification System with Hardware-Based Security

Master's Thesis

to achieve the university degree of
Diplom-Ingenieur
Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Ass.-Prof. Dipl.-Ing. Dr.techn. Christian Steger
Institute for Technical Informatics

Advisor

Ass.-Prof. Dipl.-Ing. Dr.techn. Christian Steger
Dipl.-Ing. Thomas Fischer (Infineon Technologies AG)

Graz, January 2019

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

....

Date   Signature

Abstract

Since the drone market is growing rapidly, safety critical problems are gaining priority. Recent incidents, such as those which interrupted the air traffic on airports in London, have raised awareness for drone identification, authentication and tracking, in order to find and discipline the legal owner. To prevent that kind of incidents, aviation authorities such as the FAA or the EASA, are currently working on regulations. Drone identification systems in general consist of two main components: the drone, and the flight control server. There are two major challenges within this context. First, the communication channel between the two parties has to be secured reliably, in order to verify the exchanged information. Second, most state-of-the-art solutions require detection beforehand.

This thesis proposes a globally available, secured drone identification system, based on reliable security mechanisms and standardized protocols. In order to achieve a secured communication channel, the Transport Layer Security (TLS) protocol is used. This protocol performs an authentication procedure during the connection establishment. TLS is based on digital signatures and certificates, which implies the need for a secured environment to prevent leakage of confidential information. Since state-of-the-art cryptographic protocols are mostly implemented purely in software, they are prone to side channel attacks. These attacks can lead to extraction of confidential information. Therefore the TLS layer is partitioned between the host and a Hardware Security Module (HSM). The HSM provides a secured storage for authentication keys and certificates, which are necessary for securing the communication channel. To validate the certificates which are used for authentication of the drone against the flight control server and vice versa, the certificates have to be distributed to all parties within the system beforehand. To solve this, a certificate provisioning process is proposed. To provide reliable and global connectivity to the drone, an LTE module is attached to connect to the cellular network. In order to reach global, economical connectivity and prevent the expensive usage of roaming tariffs, an Embedded SIM (eSIM) instead of a traditional SIM is used for Mobile Network Operator (MNO) authentication. The eSIM provides a mechanism to change the MNO over-the-air, depending on the location or other desired properties.

Compared to current available solutions, the proposed system provides reliably secured authentication of a drone against a flight control server, which does not require detection beforehand. Due to the fact that standardized protocols, a certified HSM and a globally available physical link is used, upcoming regulations could be influenced. This thesis also seeks to raise awareness for further challenges beyond future regulations concerning drone identification.

Kurzfassung

Durch das rasante Wachstum des Drohnenmarktes steigt auch die Gefahr im Luftraum, wodurch sicherheitskritische Aspekte an Bedeutung gewinnen. In Anbetracht jüngster Ereignisse, wie jenen, wo Drohnensichtungen den Stillstand zweier Londoner Flughäfen erzwungen haben, sollte Bewusstsein für die Wichtigkeit von Identifizierungssystemen von Drohnen geschaffen werden. Behörden wie die FAA oder die EASA arbeiten bereits an Regulierungen, um solche Vorfälle zu vermeiden. Prinzipiell bestehen Drohnenidentifikationssysteme aus zwei Komponenten: der Drohne, sowie dem Server der Flugsicherung. In diesem Zusammenhang gibt es zwei Herausforderungen. Einerseits muss der Kommunikationskanal zwischen der Drohne und dem Server zuverlässig gesichert werden, um die ausgetauschten Informationen zu verifizieren, und andererseits müssen bei den meisten bereits verfügbaren Systemen, die Drohnen zuerst detektiert werden.

Im Kontext dieser Masterarbeit, wird ein global verfügbares System, basierend auf zuverlässigen Sicherheitsmechanismen sowie standardisierten Protokollen, zur Identifizierung von Drohnen vorgeschlagen. Um den Kommunikationskanal zwischen der Drohne und der Flugsicherung abzusichern, wird das Transport Layer Security (TLS) Protokoll verwendet, welches während dem Verbindungsaufbau, eine Authentifizierung durchführt. Dieses Protokoll basiert auf digitalen Signaturen und Zertifikaten, welche in einer sicheren Umgebung gespeichert werden müssen, um Verlust von vertraulichen Daten zu vermeiden. Dadurch, dass Verschlüsselungsprotokolle welche ausschließlich in Software implementiert sind, anfällig für Seitenkanalattacken sind, wird in dem vorgeschlagenen System die TLS Schicht zwischen dem Host Controller und einem Hardware Security Module (HSM) partitioniert. Das HSM stellt einen geschützten Speicher für Authentifizierungsschlüssel und Zertifikate, welche für das TLS Protokoll notwendig sind, zur Verfügung. Zur Validierung der notwendigen Zertifikate welche zur Authentifizierung der Drohne gegenüber dem Flugkontrollserver und vice versa, genutzt werden, müssen diese im Vorhinein im System verteilt werden. Im Rahmen dieser Arbeit wird eine Methode zur Bereitstellung bzw. Verteilung der Zertifikate vorgeschlagen. Um eine zuverlässige und globale Konnektivität der Drohne sicherzustellen, wurde ein LTE Modul integriert. Eine globale und auch kosteneffiziente Konnektivität wird erreicht, indem anstelle einer traditionellen SIM, eine Embedded SIM (eSIM) verwendet wird. Die eSIM erlaubt, den Mobile Network Operator (MNO) basierend auf dem aktuellen Standort oder anderen gewünschten Aspekten, ferngesteuert zu wechseln, wodurch das Verwenden eines teuren Roaming Tarifs hinfällig wird.

Im Vergleich zu anderen, bereits verfügbaren Lösungen, bietet das vorgeschlagene System eine zuverlässige und sichere Authentifizierung der Drohne gegenüber der Flugsicherung. Durch die Vorteile von standardisierten Protokollen, dem zertifizierten HSM sowie dem global verfügbaren Netzwerk, könnte das vorgeschlagene System, oder Teile davon, in Regulierungen einfließen. Diese Masterarbeit sollte auch Bewusstsein für weitere Herausforderungen bezüglich Drohnenidentifikation schaffen, welche trotz neuer Regulierungen, sowie der Umsetzung von Identifikationssystemen für Drohnen erhalten bleiben.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Drone identification or remote identification of UAVs is a topic with increasing importance, especially these days, since several incidents happened. One of the most important aspects is preventing drones respectively their legal owners, from flying into or over critical zones or areas. These zones can be airports, power plans, prisons, crowded places etc. Just recently, airports in London had to interrupt their air traffic due to drone sightings. Several thousands of people were stranded at the airport [1]. This, but also other recent incidents, should wake up aviation authorities to introduce a drone identification and tracking system, or at least to publish a regulation within this context, as soon as possible. The Federal Aviation Administration (FAA) sent out an Request for Information (RFI) already 2016 and also just recently (December 2018), they ask third parties for information and proposals for possible remote drone identification systems [23]. Also the European Aviation Safety Agency (EASA), which is the aviation authority on European Union (EU) level, is working on proposals for regulations. Since drone identification and authentication is, and also will be a big topic in the future, this thesis introduces a concept and a design, as well as an implementation of a reliable and secured drone identification system.

1.2 Goals

The goals of this thesis are on one hand, identifying current available solutions in the context of drone identification and pointing out their weak points and potentially exploitable vulnerabilities. On the other hand, a concept and a design for a secured drone identification system should be developed. To proof the feasibility of the proposed system and the chosen technologies, a proof-of-concept prototype should be implemented. The proposed system should provide state-of-the-art security for the communication channel from and to the drone. As a starting point for the implementation, an Infineon Technologies AG multicopter solution was given, as depicted in Figure 1.1.

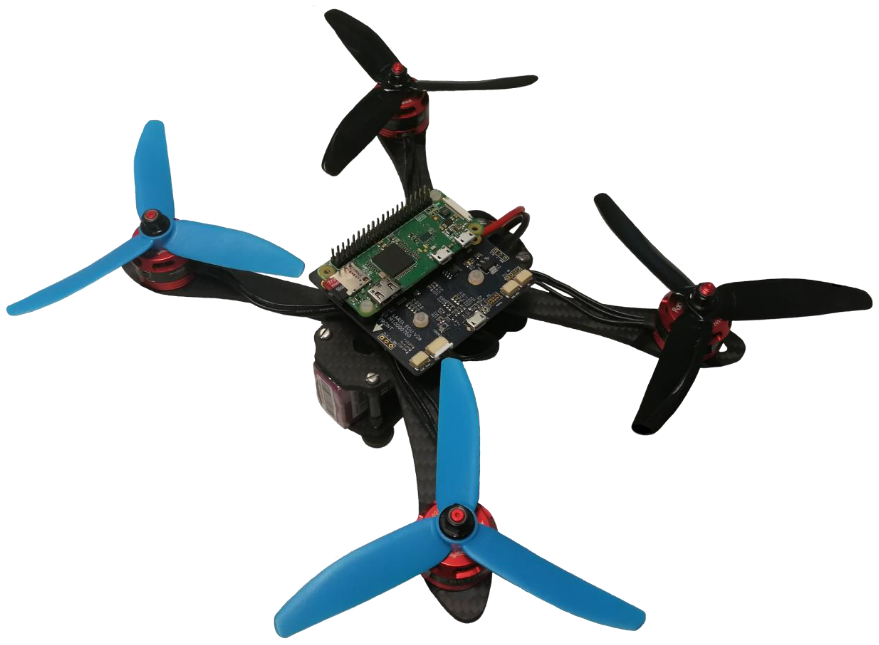


Figure 1.1: Infineon Technologies AG multicopter

1.3 Overview

This thesis introduces a secured drone identification, authentication and tracking system. First, the legal aspects were analyzed and current regulations for drones in general, were reviewed. Since there are no regulations forced yet (only proposals available), there is still room for interpretation and new ideas. An important step in this thesis was to analyze and point out drawbacks of current available solutions, or solutions which are still in a research phase.

The newly proposed system should obviate the disadvantages from other solutions, and in the best case it should not come up with new drawbacks. Since the regulations are still in a phase where they could be influenced and affected in one way or the other, one of the most important points was to rely on standardized, state-of-the-art and well proved protocols and technologies. This makes it feasible or at least easier to convince aviation authorities, in order to influence new regulations, which are coming up for sure in the next couple of years. Therefore, the introduced communication protocol stack is only based on such primitives, which are summarized in the state-of-the-art chapter.

The design and concept of the system are covering the realistic use case for the secured drone identification system. It is split into the design of the communication protocol stack, the hardware design and the software design for the drone as well as for the flight control server, where TLS is used for securing the communication channel. The implementation chapter gives an overview about the developed prototype and demonstration. The goal of the demonstrator is to show the feasibility of the security framework and the introduction of no-fly zones, which are prohibited areas such as airports or prisons. Further the demonstration also shows the quite new eSIM technology, which is used for authentication against an MNO to gain access to the LTE network. In the end, the proposed system is evaluated and compared to the already available system.

Chapter 2

State of the Art

This chapter gives an overview of drone identification, detection and tracking algorithms. There exist several detection and tracking algorithms, but for most systems, the identification, respectively the authentication, of the drone or the pilot is not possible. In addition to the existing systems, technologies which are essential for the purpose of drone identification are summarized. Further, the drone which is used for the demonstration of the proposed drone identification system is described.

2.1 Drone Identification

2.1.1 Regulations

Up to now, there are almost no regulations for drone identification which have to be taken into account while flying with drones. The only requirement in the sense of identification of drones is the labeling obligation, which means Unmanned Aerial Vehicles (UAVs) with a weight above 0.25 kg have to be labeled with name and address, to identify the owner of the drone in case of a crash [10]. In Germany the Civil Aviation Authority, originally called Luftfahrt-Bundesamt (LBA), is taking care about the aerial space, especially about the air passenger's rights. On EU level, the EASA is taking care about safety and environmental protection in air. The EASA is a higher instance compared to LBA concerning aviation safety, meaning that the EASA is auditing the LBA. The authority in the United States, which is taking care about these regulations is called FAA. These authorities are discussing about various of regulations to fight against safety problems and other issues caused by the increasing number of UAVs in the air. They also want to prevent drones from flying over critical and sensitive locations, such as airports, power plants, crowded places and so on. To achieve this, UAVs have to be uniquely identifiable on one hand, and on the other hand they have to be tracked in order to comprehend their position. That means all the authorities will regulate the identification and tracking of UAVs, even tough the regulations will be different depending on the location.

2.1.2 ISO Working Group

In 2017, the special committee ISO/IEC JTC 1/SC 17, which is working on standardizations concerning cards and security devices for personal identification, created the working group WG 12. This working group is developing standards related to drone licensing and drone identity modules [40]. Since the first meeting was in April 2018, there is no released document, but only proposals. In their proposals they want to standardize a module called

Drone Identity Module (DIM). On one hand, the main functionality of the DIM shall be the mutual authentication against a drone monitoring and tracking server which is driven by an aviation authority and on the other hand, the establishment for the secure channel for tracking information e.g. Global Positioning System (GPS) information. They also want to take future mechanisms such as drone-to-drone communication into account.

2.1.3 Identification Systems

The regulations and standardizations for identification and tracking are not official yet, meaning they are all just proposals. Anyway there are companies and research groups which are already working on the development of such systems. DJI, the leading manufacturer for civilian drones, has already published their own identification and tracking system called AeroScope. Another company which is working on such a system is Vodafone. Many systems which are in development or research progress are detection and tracking systems. If a system also offers identification or authentication, the drone has mostly to be detected by the system before, except for Vodafone's RPS system. Vodafone's approach covers similar points as the drone identification system which is developed within this thesis, but there are still several disadvantages which are explained in the corresponding chapter (Chapter 5).

2.1.3.1 DJI AeroScope

DJI's AeroScope is already available on the market, but it is a drone detection and tracking system and not a drone identification and tracking system. That means, a drone has first to be detected, before the drone can be identified and tracked. AeroScope is a system which uses the communication link between the drone and its remote control. The transmitter of the drone broadcasts the telemetry data and additional information such as serial number, which can be detected by any AeroScope receiver [18]. This means that the main elements of AeroScope are the observation receiver, which can either be stationary or portable, and the integrated software system. Due to the fact that AeroScope is based on the signals sent between the drone and the remote control, it only works in a given range. Depending on the antennas attached to the receiver unit, the range varies up to approximately 40 km [18]. Further, AeroScope is only compatible with DJI drones and not with drones from other manufactures, because not all the drone manufacturers are structuring their remote control data in the same way. Figure 2.1 depicts the approach of DJI's AeroScope system and illustrates which drones are detectable by the system.

The big advantage of this system is that nothing has the be modified or changed on the costumers drone system, or more specifically, on the drone itself or on the remote control. This is at least valid for DJI drones. If drones from other manufacturers should be detected, the manufacturers have to adapt their way of sending the data from the remote control to the drone. Further, for every location, where drones should be detected and tracked, a AeroScope system is necessary. This would force for example every airport, military facilities, prisons and other locations with safety concerns to equip their areas with such a system [16]. Another problem which comes with the fact, that the command-and-control link between the drone and the remote control is used, is that it is easily possible to spoof such packets. In addition, privacy concerns coming up, because all the data captured by the AeroScope system are broadcasted. If the regulations which are coming up in future, desire to send information about the user or the owner of the drone, it starts to affect the privacy of these persons.

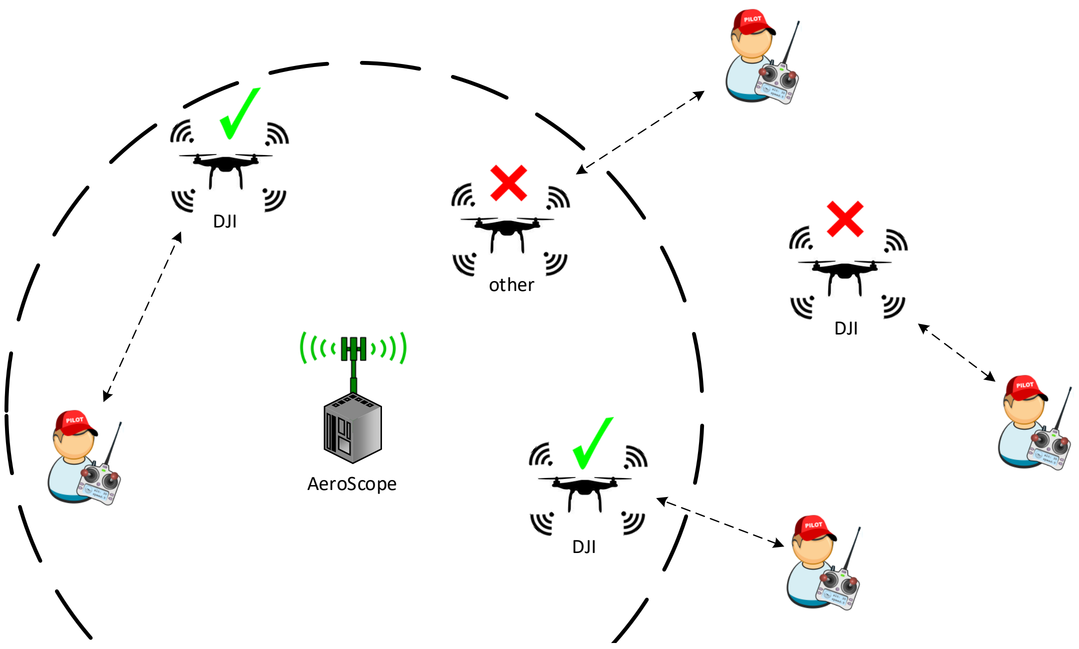


Figure 2.1: Concept of AeroScope

2.1.3.2 Vodafone RPS

Vodafone is developing a system together with the EASA, called Vodafone 4G Radio Positioning System (RPS) [77]. The system is based on the Long Term Evolution (LTE) network. The main components of the system are the LTE module and a Subscriber Identity Module (SIM) card. In [77] it is stated, that an eSIM is used in this system, but there is no official statement from Vodafone which confirms that. Vodafone declares, that their system is able to cover the following use cases [83]:

- Real-time tracking of each drone (with up to 50 meter accuracy) by drone operators and authorized bodies such as air traffic control
- Over-the-horizon/beyond line-of-sight control by the operator, greatly reducing the risk of accidental incursions when operators lose sight of their drones
- Protective geofencing, with drones pre-programmed to land automatically or return to the operator when approaching predetermined exclusion zones (such as airports and prisons)
- Emergency remote control intervention to provide the authorities with the means of overriding a drone operator's control to alter a drone's flight path or force it to land
- SIM-based e-identification and owner registration

The location tracking in Vodafone's 4G RPS system is not based on GPS, but on location data gathered out of the LTE network. Vodafone did not publish any detailed information about their RPS system, but they are stating, that the system should be released for commercial use in 2019 [83].

The method for positioning a device within the LTE network is based on Observed Time Difference Of Arrival (OTDOA), which is a downlink positioning method, where the Time Of Arrival (TOA) of signals received from multiple (at least three) base stations are measured [81]. The signals used for OTDOA are called positioning reference signals. In order to get the OTDOA, the TOAs from several neighbor base stations are subtracted from the TOA of a reference station. Each time difference results geometrically in a hyperbola, where the point of the hyperbolas intersection represents the user device location [81]. Figure 2.2 depicts the positioning system and its measurement uncertainty, which is in a range of about 50 meters. In this example, base station 1 is taken as a reference base station and the two OTDOAs are calculated [81]:

$$\tau_{2,1} = \tau_2 - \tau_1 \tag{2.1}$$

$$\tau_{3,1} = \tau_3 - \tau_1 \tag{2.2}$$

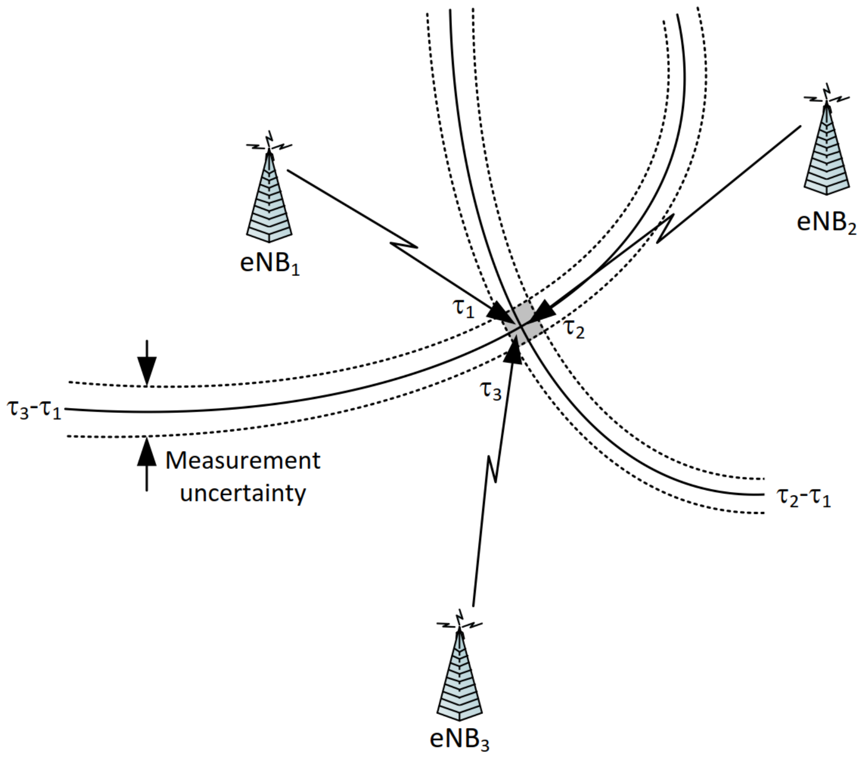


Figure 2.2: OTDOA positioning method [81]

A big advantage of this approach is, that the LTE network already exists. So there is no need for additional receivers such as those from DJI's AeroScope. The drawback of this method is that there is a need to adapt the consumer device (the drone), in the sense of adding a new hardware block containing an LTE module together with a SIM card or eSIM. Adding LTE capability to drones brings more advantages than only tracking drones, such as controlling the UAV beyond the line-of-sight or transmitting high quality video streams. This is possible because LTE is capable of high transmission rates.

2.1.3.3 Radar based System

Another experimental approach for drone detection and tracking, is based on radar systems. This approach comes with many difficulties, because drones are rather small, compared to typical targets for radar detections approaches. Further, birds have similar and often even larger sizes than drones, which makes the classification difficult [45]. If, as partly in [45] with phase-interferometric Doppler radar, it is possible to solve the problem of classification, the disadvantage compared to DJI's AeroScope system is, that the identity of the drone (or it's owner) cannot be determined. That means radar based systems are only detecting and tracking drones. An advantage compared to the AeroScope system is, that the drone does not have to be modified in any way. Another disadvantage as for the AeroScope system is, that many base stations are necessary, depending on the range of the radar.

2.1.3.4 Image Processing based System

In paper [61], another drone detection and tracking system is introduced [61]. This system is based on image processing and neural networks. A Pan-Tilt-Zoom (PTZ) camera and a Convolutional Neural Network (CNN) is used [61]. If a drone is captured with the PTZ camera, it is tracked, which means with each camera only one drone is track-able. The used state-of-the-art CNN in this paper, is working in two stages [61]. In the first stage the object is detected based on a given image, and in the second stage the object is classified. This system has similar advantages and disadvantages as the radar based systems. Details about this system are not explained here, because the focus of this system is not the identification respectively authentication of drones, but detecting and tracking. Details can be found in [61].

2.2 Data Transmission

In terms of storing and transmitting data, data serialization formats are playing a major role, because processing or transmitting structural data is not possible without serializing the data before. To reconstruct the origin data structure, rules - respectively a format, has to be defined in advance. There are already many different data serialization formats with different focuses out there, which means, that in most cases there is no need to define a new format. Mobile applications, as this thesis deals with, are typically resource limited and therefore the data size property has be taken into account when choosing a serialization format. Further, mobile applications need a wireless and widely spread communication technology, which is LTE nowadays.

2.2.1 Data Serialization Formats

This chapter gives a comparison of four standardized data serialization formats with the indicator stated in [78], such as data size, serialization speed and ease of use. Only standardized serialization formats for binary representation are selected in this chapter, considering our use case. The comparison is based on sample data which contains location information (coordinates and attitude) and an identifier.

2.2.1.1 ASN.1

The Abstract Syntax Notation One (ASN.1) is a data serialization standard defined by the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) and the International Organization for Standardization (ISO).

Most commonly, data is encoded with the Basic Encoding Rules (BER) in ASN.1. BER encoded data consists of the following four components, as written in ITU-T X.690 [41]:

- Identifier octets
- Length octets
- Content octets
- End-of-content octets (optional; defined in length octets)

Especially in mobile applications the Packed Encoding Rules (PER) are preferred because they save bandwidth due to the more compact transfer syntax. For the identifier octets, there exist several different data types.

> *"The abstract syntax of the data elements is described in ASN.1 modules, using basic types and construction rules. The basic types are the BOOLEAN, INTEGER, ENUMERATED and REAL types, plus the BIT STRINGs and OCTET STRINGs and a special type called OBJECT IDENTIFIER, which provides unique indexes for various types of entities and protocols, coded on a small number of octets."* [33]

In addition to this basic types, there also exists further types, for instance an UTF8-String type which is used in the comparison example.

Overhead		Payload	Type	Raw data
Length	Type*			
09	80	D1 1689577BE7D7EB	REAL	45.0729823
09	80	CF 1F2DC797D9A5F9	REAL	15.5894134
04	80	00 016B	REAL	363
08		6162636465666768	UTF8-String	"abcdefgh"

Table 2.1: ASN.1 example

Table 2.1 presents an example to explain the structure of an ASN.1 encoded data set. For example 09_{16} means length 9 and 80_{16} represents the type REAL, where the payload consists of the exponent ($D1_{16}$) and the mantissa ($1689..._{16}$). Type* represents the type for REAL number encoding (Base, Exponent, Mantissa).

2.2.1.2 CBOR

The data serialization format Concise Binary Object Representation (CBOR) was created by the Internet Engineering Task Force (IETF) and is designed for small code size and small message size [72].

> *"CBOR is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation."* [13]

CBOR is based on the JSON data model, and in comparison to ASN.1 no scheme is necessary. The structure consists of an initial byte, which contains the major type and additional information. The major type is represented by the high-order three bits and additional information is stored in the remaining five bits. Major types are: Unsigned integer, negative integer, byte string, text string, array of data items, map of pairs of data items, optional semantic tagging of other major types, floating point numbers etc. [13]. Encoded CBOR data can be seen as a stream of data items e.g. [14].

CBOR data	Data item 1				Data item X
Byte count	1 byte (data item header)		Variable	Variable	etc...
Structure	Major type	Add. info	Payload length (opt.)	Data payload (opt.)	etc...
Bit count	3 bits	5 bits	8 bits x variable	8 bits x variable	etc...

Table 2.2: CBOR data stream

Considering Table 2.2 the following example points out the way CBOR data is structured and serialized. The two left columns are overhead, which contains information about the type and/or information about the length of the data. For example the byte 84_{16} represents an array (8_{16}) with the length of four (4_{16}) elements.

Overhead	Payload	Type (length)	Raw data
84		array (4)	
FB	404689577BE7D7EB	double	45.0729823
FB	402F2DC797D9A5F9	double	15.5894134
19	016B	unsigned int	363
48	6162636465666768	bytes (8)	"abcdefgh"

Table 2.3: CBOR example

CBOR also provides basic security services, which are defined in the CBOR Object Signing and Encryption (COSE) protocol. This protocol is specified in RFC 8152 [72]. All the COSE messages are built on the array type from the CBOR serialization format. This array always starts with the same three elements [72]:

- Protected header parameters (wrapped in a byte string)
- Unprotected header parameters (as a map)
- Content of the message (either plain text or cipher text)

To separate different types of cryptographic concepts, COSE messages can consist of several layers. Further, the protocol defines a set of message types, which can either be tagged or not. Tagged message types contain a CBOR tag which identifies the COSE type. To identify the type for untagged messages, there are additional methods. A detailed explanation for those methods can be found in [72]. Table 2.4 depicts the existing COSE message types as well as their corresponding CBOR tags.

2.2.1.3 BSON

Binary JSON (BSON) is a data serialization format based on JavasScript Object Notation (JSON) with the major extension for binary data. *"BSON is a binary format in which zero or more ordered key/value pairs are stored as a single entity."* [12] This serialization

CBOR tag	COSE type	Data item	Semantics
98	cose-sign	COSE_Sign	COSE signed data object
18	cose-sign1	COSE_Sign1	COSE single signer data object
96	cose-encrypt	COSE_Encrypt	COSE encrypted data object
16	cose-encrypt0	COSE_Encrypt0	COSE single recipient encrypted data object
97	cose-mac	COSE_Mac	COSE MACed data object
17	cose-mac0	COSE_Mac0	COSE MAC w/o recipient object

Table 2.4: COSE message identification [72]

format stores field names within the encoded data, which provides flexibility. On the other hand, this leads to overhead and therefore to a disadvantage in space efficiency.
Basic types for BSON are: byte, int32, int64, uint64, double, decimal128.

2.2.1.4 XDR

The External Data Representation (XDR) was initially defined by Sun Microsystems and later also considered as an Request for Comments (RFC) (RFC 1014) [79]. In XDR, the block size is always a multiple of four bytes of data. If the byte number is not a multiple of four, then the block gets filled up (padded) with zero to three zero bytes to fulfill this constraint as depicted in Table 2.5.

byte 0	byte 1	...	byte n-1	0	...	0
n bytes				r bytes		
n+r (where (n+r) mod 4 = 0						

Table 2.5: XDR structure

The following basic data types are supported by XDR: Integer, unsigned integer, enumeration, boolean, hyper integer and unsigned hyper integer, floating point, double-precision floating point, variable/fixed-length opaque data, string, variable/fixed-length array, structures, discriminated union, void, typedef, optional data [79].

2.2.1.5 Comparison

As stated in the introduction of this section, this comparison focuses on important properties for data serialization formats such as data size, serialization speed and ease of use. For comparison the following raw data set was used:

Field name	Value	Size	Type
Latitude	47.0729823	8 bytes	double
Longitude	15.4212163	8 bytes	double
Attitude	3633	2 bytes	unsigned integer
Identifier	"abcdefgh"	8 bytes	string
Total		26 bytes	

Table 2.6: Sample data set

Figure 2.7 depicts a comparison of the data serialization formats ASN.1, CBOR, BSON and XDR. Beside the major properties, there is another important fact which should be considered for the choice of the serialization format while designing an application. If the application or parts of it, should be standardized at some point, then a standardized data serialization format should be preferred. Further it should be considered which libraries are available for the serialization, which is an indicator for the ease of use. A programming language typically used to write code for micro controllers is C, therefore it makes sense to use a standardized serialization format with an already available C library.
In the presented example, it can be observed that CBOR has the lowest overhead as well as the best timings for de- and encoding of data. The only disadvantage in comparison to BSON is that CBOR is sensitive to the order of the arriving raw data fields. In contrast, BSON not only encodes information about the data field and the data itself into the data stream, but also the field names. This is an advantage in terms of flexibility, but it also leads to a huge overhead. Especially for mobile applications with limited resources this can cause problems or decrease the efficiency of the mobile application.

Format	Plain	Encoded	Overhead	Overhead	Encoding	Decoding	Standardized
ASN.1	26 bytes	34 bytes	8 bytes	31 %	0.622 ms	0.405 ms	ISO
CBOR	26 bytes	31 bytes	5 bytes	19 %	0.045 ms	0.041 ms	RFC
BSON	26 bytes	57 bytes	31 bytes	119 %	0.500 ms	0.160 ms	BSON spec
XDR	26 bytes	32 bytes	6 bytes	23 %	0.258 ms	0.101 ms	RFC

Table 2.7: Comparison of data serialization formats

In Table 2.8 the encoded data streams for all compared serialization formats and the used libraries are indicated. All test were executed on a Raspberry Pi 3 Model B V1.2 with Raspbian GNU/Linux 9 (stretch) and Python 2.7.13 installed.

Format	Used library version	Encoded data
ASN.1	Python asn1tools 0.122.0	0980d11689577be7d7eb0980cf1f2dc797d9a5f9048000016 b086162636465666768
CBOR	Python cbor 1.0.0	84fb404689577be7d7ebfb402f2dc797d9a5f919016b68616 2636465666768
BSON	Python bson 0.5.6	39000000016c617400ebd7e77b57894640016c6f6e00f9a5d 997c72d2f4010617474006b01000000269640009000000006162 6364656667680000
XDR	Python xdrlib	404689577be7d7eb402f2dc797d9a5f943b58000000000086 162636465666768

Table 2.8: Byte streams of data serialization formats

2.2.2 Data Communication Technologies

In order to transmit data, a data communication technology is necessary. A widely spread state-of-the-art standard for this purpose is LTE/4G. It's successor is 5G, which is currently in the roll out phase, and therefore not yet available in most regions. All these standards are released by 3rd Generation Partnership Project (3GPP)[52].

2.2.2.1 LTE

LTE has a high bandwidth (up to 20 MHz) as well as a high down- and uplink data rate (uplink: 75 Mbps, downlink; 300 Mbps). Therefore LTE can also be used for further use cases in addition to a drone identification system, such as cloud connectivity or video streaming [52]. In order to get access to a mobile network, such as LTE, the device has to identify itself against a mobile network operator, which is done with a SIM.

2.2.2.2 eSim

Traditionally, SIM cards are needed to identify a device, respectively a user, against a network operator and subsequently to connect the device to the internet. The SIM card is also known as Universal Integrated Circuit Card (UICC) and is available in different form factors for instance Mini SIM, Micro SIM or Nano SIM. That means an UICC contains applications to enable access to GSM, UMTS/3G and LTE/4G networks. With this technology, the SIM card has to be replaced if the user wants to change the MNO. Based on this existing removable SIM card technology, GSM Association (GSMA) has defined a set of specifications to allow users to change the operator with Remote SIM Provisioning (RSP). This technology is called eSIM or Embedded Universal Integrated Circuit Card (eUICC) with the specific form factor MFF2, which is defined in the ETSI standard [21]. Due to this miniaturization, the physical size of devices can be reduced and therefore also new devices can be equipped with a mobile network connection. One of the main reasons for eUICC is the simplification of industrial and logistical processes for the distribution of Machine to Machine (M2M) equipment [28]. This means, instead of replacing the SIM card, the MNO can be changed by switching the Profile, which is explained in more detail in Section 2.2.2.2.2.

2.2.2.2.1 Architecture

This section gives a high-level overview of the architecture of the eUICC. More details can be found in the corresponding GSMA specification [26].

"The Embedded UICC Controlling Authority Security Domain (ECASD) is responsible for the secure storage of credentials needed to support the required security domains on the eUICC." [26] The ECASD contains private keys for creating signatures, associated certificates for the eUICC authentication and public keys for verifying SM-DP+ and SM-DS. SM-DP+ stands for "Subscription Manager Data Preparation - enhanced" and is responsible for the creation, generation, management and the protection of resulting Profiles in the M2M world [71]. The Subscription Manager Discovery Server (SM-DS) is used for discovering available Profiles, if the consumer device approach is chosen. Further, the ECASD contains the eUICC manufacturer's keyset for key and certificate renewal [26].

ISD-R and ISD-P stands for Issuer Security Domain Root, respectively Profile. While ISD-R is responsible for creating new ISD-Ps as well as their lifecycle management, ISD-P represents a secure container for hosting a Profile. The Profile Policy Enabler, the Telecom Framework and the Profile Package Interpreter are eUICC operating system services with certain responsibilities described in the architecture specification [26].

The Local Profile Assistant (LPA) services provide the necessary access to the root SM-DS address and the optionally stored default SM-DP+. Further, it facilitates the reception of the Bound Profile Package transferred from the LPA itself. They also provide a Local Profile Management and information about installed Profiles. In addition, these services provide functions to authenticate the LPA against the SM-DS [26]. The LPA can

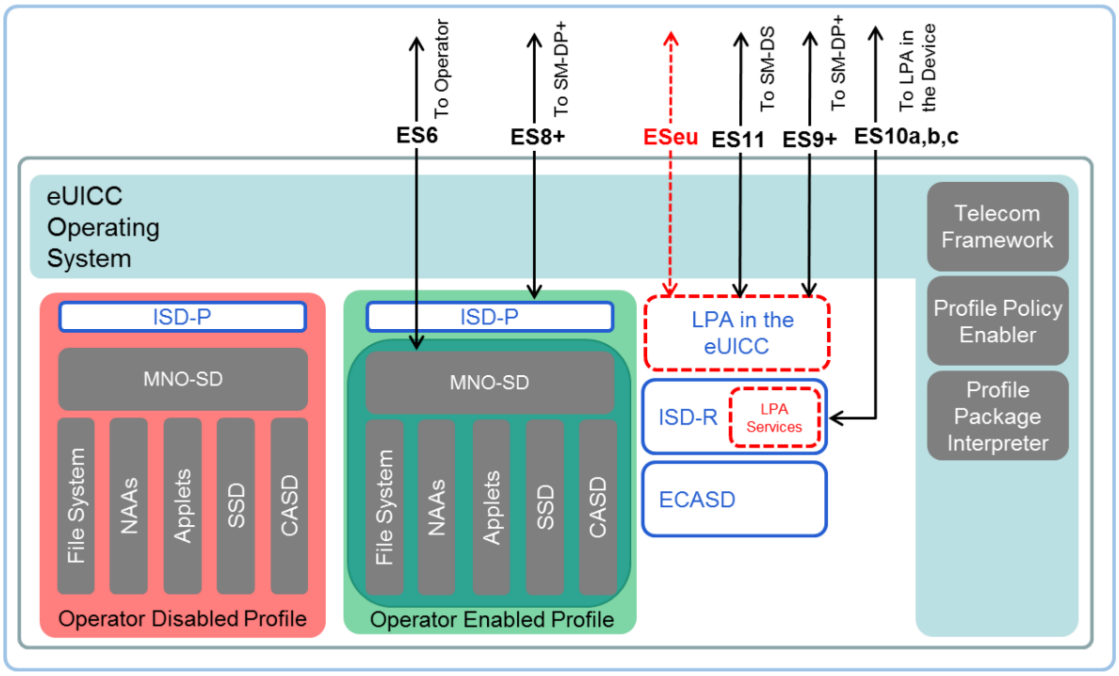


Figure 2.3: Schematic representation of the eUICC [26]

either be in the device (e.g. smartphone) or directly in the eUICC. In the second case it is called embedded LPA.

GSMA distinguishes between two different approaches for eSIM. On one hand there is the M2M solution and on the other hand there is the consumer device solution. The main difference between those approaches is the way a Profile change is triggered, which is sketched in Figure 2.4.

For the M2M approach the service provider triggers an MNO change and the new Profile is pushed to the device. Contrarily, in the consumer approach the consumer is allowed to change the MNO directly on the device and subsequently the new Profile is pulled onto the device. Further, the M2M eSIM is linked to only one subscription manager, where the consumer eSIM can affiliate to any subscription manager based on the Public Key Infrastructure (PKI) [25].

Profiles are MNO specific and are necessary to identify the eUICC against a certain MNO to provide services. On an eUICC several Profiles can be stored, but only one Profile can be enabled at a time, all others have to be disabled.

A Profile consists of Profile components [27]:

- One MNO-SD
- Supplementary Security Domains (SSD) and CASD
- Applets
- Applications, e.g. NFC applications
- NAAs
- Other elements of the file system
- Profile metadata, including Profile Policy Rules

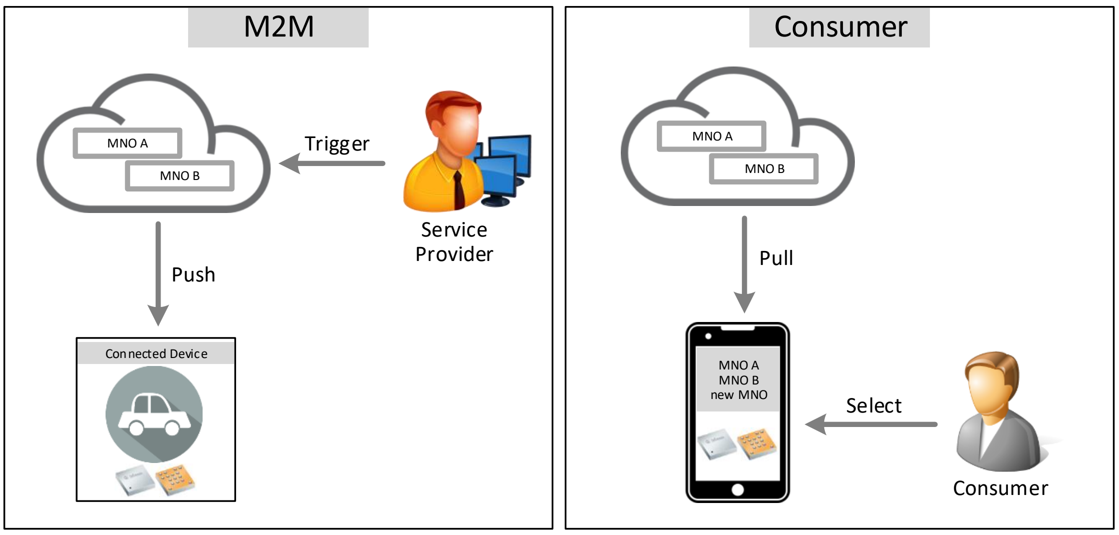


Figure 2.4: eSim comparison (M2M and consumer approach)

The MNO-SD represents the operator, contains the operator's Over-The-Air (OTA) keys and provides a corresponding secure OTA channel. If a Profile is enabled, the eUICC behaves as a traditional UICC. This applies especially for the Network Access Application (NAA), which holds a set of files and credentials which belongs to an MNO to grant access to a mobile network like LTE and also for applets contained in the Profile [27].

In general, the eUICC's main purpose is identifying a device against a mobile network operator to gain access to the mobile network, but since eUICC supports Java Card, it is possible to extend the functionality by adding a Java applet to the Profile. For example MasterCard or Visa have already deployed applets for secured paying.

2.2.2.2.2 Remote SIM Provisioning

RSP describes downloading, installing, enabling/disabling and deleting of Profiles on an eUICC. The detailed technical specification can be found in the official GSMA document [27]. As already mentioned in Section 2.2.2.2.1, it has to be distinguished between eUICC M2M and eUICC consumer. Both types have similar functionality, but in general they are totally different products with various differences concerning the architecture. Figure 2.5 shows the RSP system for the consumer approach, with the LPA in the device.

To initially get a Profile into the eUICC, a registration at the SM-DS happens, and the eUICC Manufacturer (EUM) applies for a certificate from the Certificate Issuer (CI). Then the eUICC gets the certificate from the EUM and stores it in the ECASD.

If the end user wants to change the network operator, a new Profile has to be downloaded if it is not yet on the eUICC (in a disabled state). Therefore a contract with the network operator has to be signed. Then the SM-DP+ prepares the Profile and informs the Discovery Server. If the end user wants to add the Profile on the device, the SM-DS is asked, where the corresponding Profile can be found. The device receives the information from the SM-DS, at which SM-DP+ the Profile is located. In the final step, the device downloads the Profile from the SM-DP+ and then it is ready to be enabled.

The LPA, which is in the device in this case, provides software modules for the LPA services located in the eUICC, which are the Local User Interface (LUI), the Local Profile Download (LPD) and the Local Discovery Service (LDS). The LDS is responsible for

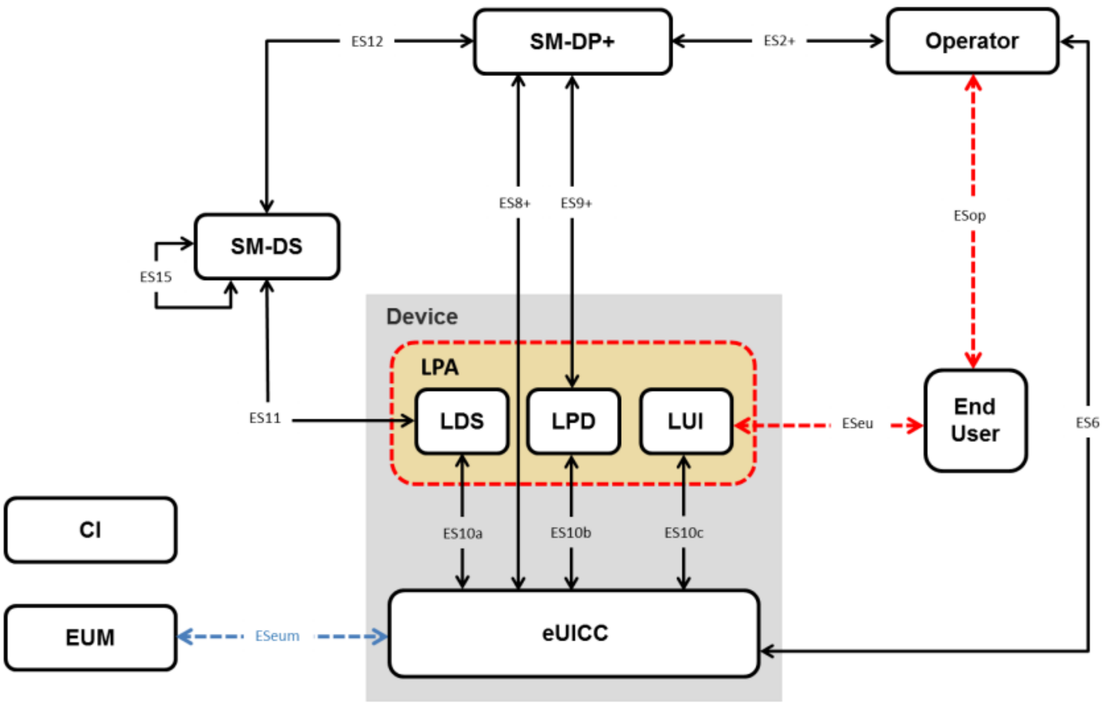


Figure 2.5: RSP system with LPA in the device [27]

retrieving pending event records from the SM-DS. The Profile download happens in two stages, where the LPD acts as a proxy. First, the Bound Profile Package from the SM-DP+ is downloaded into the LPD in a single transaction, and then the Bound Profile Package is transferred into the eUICC in segments. The LUI allows the end user to perform Local Profile Management on the device [27].

2.2.3 Protected Communication

The communication protocol stack is depicted in Figure 2.6, where the TLS block is important to provide protected communication.

To establish a protected communication, the TLS protocol is widely used and also approved by the IETF. TLS is establishing a connection between a server and a client with authenticity, integrity and confidentiality. The first step in the TLS protocol is validating certificates and the version of TLS protocol to initiate and further encrypt the communication tunnel [2]. After this step the most important phase, the TLS handshake starts, where asymmetric public key cryptography is used to authenticate each other and exchange necessary keys. If the handshake was successful, the information exchange can start. The information is encrypted by using symmetric cryptography to provide confidentiality, where the symmetric key is exchanged in the handshake before [82]. To ensure authenticity and integrity, the TLS protocol signs each message with a Message Authentication Code (MAC) [2].

In order to provide hardware-based security, the TLS block from Figure 2.6, is partitioned between the host controller and a secure element (HSM). For secure elements from Infineon Technologies AG, the typical structure of the TLS layer is depicted in Figure 2.7.

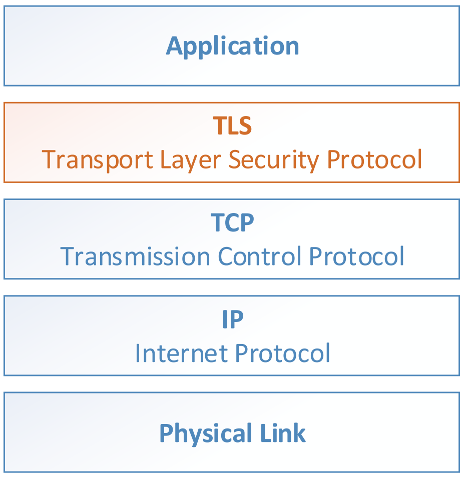

Application
TLS
Transport Layer Security Protocol
TCP
Transmission Control Protocol
IP
Internet Protocol
Physical Link

Figure 2.6: TLS communication protocol stack

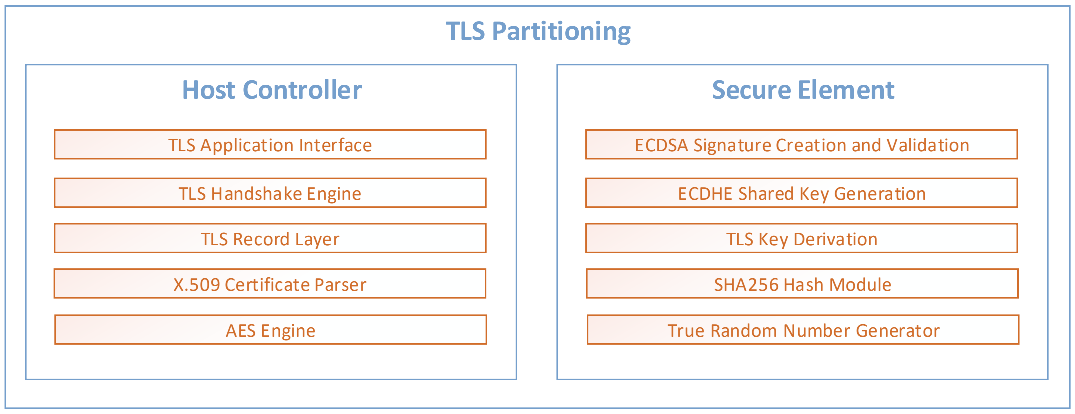


Figure 2.7: TLS partitioning (modified from [64])

2.2.3.1 Hardware Security Modules

> *"HSMs are used to protect highly sensitive data. For example, HSMs are generally defined to handle cryptographic responsibilities, such as key generation, public/private key cryptography, data encryption, and secure storage of cryptographic data. As implied by the name, conventional HSMs provide their functionality by way of hardware, i.e., circuitry. The conventional HSM hardware is defined to provide a specific and restrictive external interface that allows only authorized entities to access the data stored with the HSM and control the HSM for the purpose of generating data."* [20]

Beside safeguarding digital keys and providing capabilities for cryptographic processing, HSMs protect against certain physical attacks, e.g. side channel attacks. With a HSM a clean separation between the application logic and the security responsibilities of a system is enforced. The cryptographic algorithms are executed in a trusted environment and are therefore not prone to certain software loopholes, such as buffer overflows.

Typically, HSMs are certified by international standards such as the Federal Information Processing Standards (FIPS), Common Criteria (CC) or Deutsche Kreditwirtschaft (DK), to indicate that certain criteria are met.

HSMs provide various features to achieve integrity, availability and confidentiality for the communication of the device with the outer world. As already mentioned, HSMs offer a protected memory to store cryptographic keys and certificates. Further, they come with symmetric and/or asymmetric algorithms. Rivest–Shamir–Adleman (RSA), Elliptic Curve Digital Signature Algorithm (ECDSA), Elliptic Curve Diffie-Hellman Ephemeral Key Exchange (ECDHE) or Elliptic Curve Cryptography (ECC) are examples for asymmetric algorithms and Advanced Encryption Standard (AES), Data Encryption Standard (DES) or International Data Encryption Algorithm (IDEA) are examples for symmetric algorithms. All the cryptographic algorithms depend on random numbers with high entropy. There are two ways for creating random numbers. On one hand random numbers can be generated in software, which ends up with deterministic randomness, where on the other hand, random numbers can be generated based by physical sources, such as thermal noise or the photoelectric effect, which are statistically random [49]. HSMs provide a True Random Number Generator (TRNG). Another feature, available in certain HSMs, is the possibility of cryptographic hashing. Cryptographic hash functions are used to create a checksum from data with undefined size, to provide integrity. Common cryptographic hash functions are for example SHA-1, SHA-256 or MD5.

2.2.3.2 Side Channel Attacks

Side channel attacks are attacks, which are trying to extract secret keys and information via an indirect path, using side channel information. Side channel information can be derived from physical characteristic such as power consumption, electromagnetic radiation, execution time etc., which correlates to specific steps in cryptographic algorithms [50]. In general, side channel attacks can be categorized into three main groups, which are manipulative attacks, observing attacks and semi-invasive attacks, as illustrated in Figure 2.8 [54].

Manipulative attacks are attacks where the adversary manipulates the hardware itself. This attack group includes more obvious attacks such as using microscopic needles to extract information directly form the signal line, as well as performing microsurgery on the silicon using the Focused-Ion Beam (FIB) technology [54].

Observing attacks are already well known over decades. The most known property to observe is the power consumption, while performing cryptographic algorithms. In general, observing attacks are attacks which are just analyzing the input and the output and not manipulating the chip or the hardware itself. Further examples for properties to observe are the timing behaviour and the heating effect. The third side channel attack group is called semi-invasive attacks. Here, adversaries are trying to induce a faulty behaviour into a security controller, which means the functionality of the chip is disturbed by external influences such as power spikes and light emissions [82] [54]. This can be used to circumvent security decisions in the software, but keeping the hardware operational.

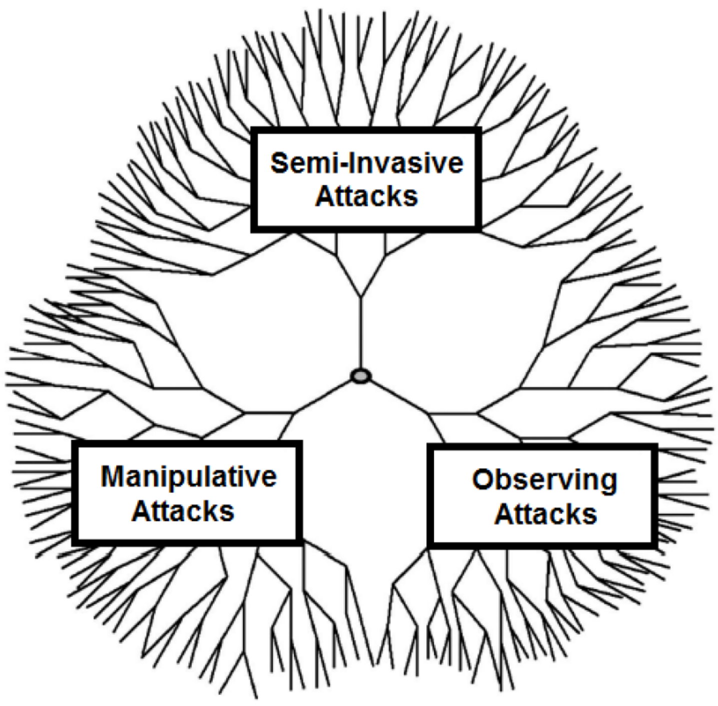


Figure 2.8: Side channel attack tree [82]

2.2.3.3 OPTIGA Trust X

The OPTIGA Trust X is a high-end security controller (HSM) developed by Infineon Technologies AG. The three main use cases for the OPTIGA Trust X are:

- Authentication
  - Brand and IP protection
- Protected communication
  - Transport Layer Security (TLS)
  - Datagram Transport Layer Security (DTLS)
- Secured firmware update, secured boot

In Figure 2.9 the system block diagram for a application with the OPTIGA Trust X is depicted. The system consists of a host and a secure element part, which are connected via an Inter-Integrated Circuit (I2C) interface. The green blocks on the left are providing an interface to the functionality of the OPTIGA Trust X, which are referred as OPTIGA Trust X Host Library. The blue blocks sketch user implemented parts, such as the target application and abstraction layer for specific drivers (PAL).

This HSM can store four ECC keys as well as four X.509 certificates. The four keys can be replaced according to the lifecycle management of the application. Since authentication is getting more and more important, the OPTIGA Trust X offers a trust anchor for mutual authentication and for secured firmware update, which is stored in the protected storage of the HSM [66]. The OPTIGA Trust X provides a set of data objects which can store arbitrary data. This HSM comes also with a cryptographic toolbox, where cryptographic functions and protocols can be invoked via the local host [37].

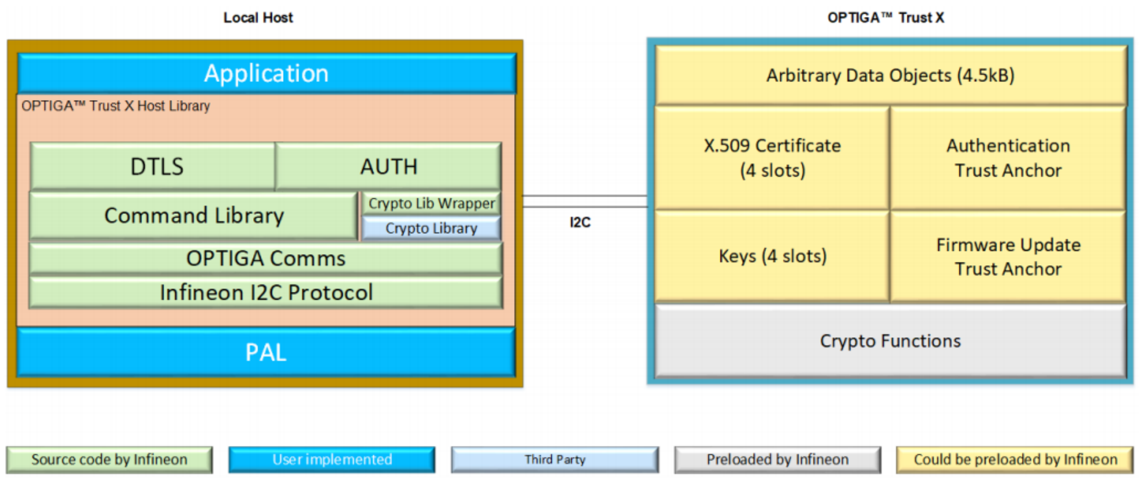


Figure 2.9: System block diagram for OPTIGA Trust X [37]

The OPTIGA Trust X comes with a set of functionalities which are explained in great detail in the solution reference manual [38]. The key features (Crypto Functions in Figure 2.9) of this secure element are:

- Elliptic Curve Digital Signature Algorithm (ECDSA)
  - Signature creation
  - Signature verification
  - Key pair generation
- Elliptic Curve Diffie-Hellman Ephemeral Key Exchange (ECDHE)
- True Random Number Generator (TRNG)
- Hash SHA256
- TLS 1.2 Key Derivation Function (KDF)

The authentication use case is supported by the ECDSA sign functionality. To provide protected communication via the TLS or DTLS protocol, the ECDSA sign and verify functionality as well as the TRNG, the Hash SHA256, the ECDHE and the KDF are required. During the TLS handshake a shared secret is calculated with support of the ECDHE functionality of the OPTIGA Trust X. Afterwards the derive key command is used to expand the generated key to a certain length, which can later also be split up in several keys for the encryption and MACs. To implement secured firmware update or secured boot for a secured system, the hash functionality and signature verification functionality are necessary. The input for the calculate hash function of the OPTIGA Trust X can either be supplied by the host or be taken from an arbitrary data object within the HSM. Since all data objects in the HSM have associated access permissions, the required permissions must be met [38].

2.2.3.4 Comparison to other Hardware Security Modules

Features	Infineon OPTIGA Trust X	Microchip ATECC508A	NXP A71CH	ST STSAFE-A100
Certified	CC EAL6+	No	No	CC EAL5+
ECDSA	Yes	Yes	Yes	Yes
ECDH	Yes	Yes	Yes	Yes
AES	AES-128	No	AES-128	AES-128, AES-256
ECC	NIST P256, P384	NIST P256	NIST P256	NIST P256, P384
TRNG	Yes	Yes	Yes	No
Hashing	SHA-256	SHA-256	SHA-256	SHA-256, SHA-384
KDF	Yes	Yes	Yes	No
Storage	4 X.509 certs 4 ECC based keys 4.5 kB arbitrary data objects 2 PKI trust anchor	9.7 kB (16 slots) for: Keys Signatures Certificates Calibration Other info	4 ECC based key pairs 8 128 bit sym keys 3 ECC based pup keys 3 AES-128 config keys 4 kB secure storage 1 key for I2C	6 kB NVM
Interface	I2C (<1 MHz)	I2C (<1 MHz), SPI	I2C (<400 kHz)	I2C (<400 kHz)
Monotonic counter	Yes	Yes	Yes	Yes
Unique ID	Yes	Yes	Yes	Yes
TLS handshake in HSM	Yes	No	No	No
SW library	Host library OpenSSL engine	OpenSSL engine WolfSSL	Host library	Host library
$T_{operating}$	STR: $-25\,°C$ to $85\,°C$ ETR: $-40\,°C$ to $105\,°C$	TR: $-40\,°C$ to $85\,°C$	T1: $-25\,°C$ to $85\,°C$ T2: $-40\,°C$ to $90\,°C$	TR: $-40\,°C$ to $105\,°C$
$I_{operating}$	6 - 15 mA (adjustable) <100 μA sleep	5 mA <150 nA ultra sleep	10 mA 40 μA typical sleep 10 μA deep sleep	14 - 21 mA <460 μA standby
Packaging	PG-USON-10-2 (3 x 3 mm)	SOIC (6 x 4.9 mm) UDFN (2 x 3 mm)	HVSON8 (4 x 4 mm) WLCSP (2 x 2 mm)	SO8N (5 x 4 mm) UFDFPN8 (2 x 3 mm)

Table 2.9: OPTIGA Trust X competitor comparison

2.3 Positioning Systems

For drone identification systems, the positioning of the drone is the second important part, beside the identification of the drone respectively the user. For real use cases, the drone is operated outdoor, but for demonstrating the identification of the drone, the drone will be operated on a small environment (on a table), which means that outdoor positioning systems will not work properly.

2.3.1 Outdoor Positioning Systems

For outdoor positioning systems GPS is widely used, which is part of the Global Navigation Satellite Systems (GNSSs). GNSS systems can provide a sub meter accuracy in outdoor environments [60]. Another approach are cellular based systems, which are worse in accuracy (several meters). Vodafone is using such an approach and they state, that they can reach an accuracy of approximately 50 meters [83].

2.3.2 Indoor Positioning Systems

For indoor positioning systems there is no prevailing standard [60]. A heavily used technology for indoor positioning systems is Bluetooth Low Energy (BLE), which is described in the subsequent section (Section 2.3.2.1). Alternatively, Wi-Fi is often used. Both technology have similar advantages and disadvantages, which are mainly the problem of radio interference and inherent latency [60]. Both approaches are based on the Received Signal Strength Indicator (RSSI) measurement. The accuracy is in a meter range, but by using a certain number of measurement points and a special beacon setup, the accuracy can be improved. Details about the beacon setup are depicted in the corresponding implementation section (Section 4.1.3.1).

2.3.2.1 Bluetooth Low Energy

BLE is a wireless technology designed for very low power consumption. When using BLE, different network topologies such as point-to-point, broadcast or mesh can be used. Compared to the basic Bluetooth [9], a power consumption of 1% to 50% can be reached, depending on the use case. Figure 2.10 depicts the basic blocks of the BLE stack. The physical layer operates in the 2.4 GHz frequency spectrum of the Industrial, Scientific and Medical (ISM) band with 40 channels which are spaced with 2 MHz. When a device is working in advertising mode it broadcasts information. For this purpose, three out of the 40 channels are reserved. The link layer is responsible for the state of the device, which can either be standby, advertising, scanning, initiating or connected [84].

"The Generic Attribute Profile (GATT) establishes how data will be organized and exchanged over a BLE connection." [56] The GATT server shall contain Generic Access Profile (GAP) services to configure certain GAP parameters such as advertising payload. In addition the GAP defines security modules and procedures which are implemented in the Security Manager Protocol (SMP) [8]. For the use case, covered in this thesis and depicted in Figure 2.11, the broadcaster and observer role, are key features, since a Bluetooth beacon is used for distance estimation.

2.3.2.1.1 Distance Measurement with BLE

Since in paper [53], BLE is evaluated for indoor distance estimation, we tried such an approach for this use case with some modifications which are described in more detail in

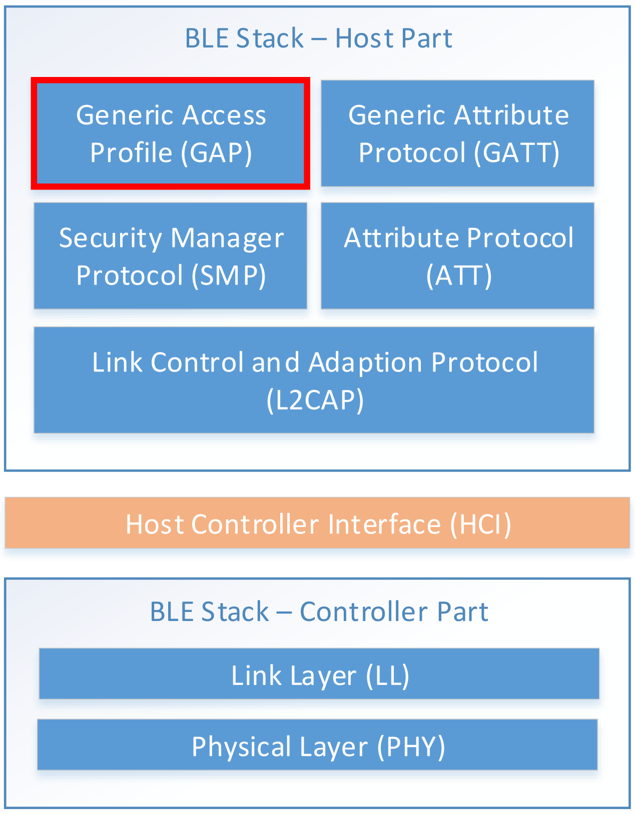

BLE Stack – Host Part
Generic Access Profile (GAP)
Generic Attribute Protocol (GATT)
Security Manager Protocol (SMP)
Attribute Protocol (ATT)
Link Control and Adaption Protocol (L2CAP)
Host Controller Interface (HCI)
BLE Stack – Controller Part
Link Layer (LL)
Physical Layer (PHY)

Figure 2.10: BLE stack [82]

the corresponding implementation section (Section 4.2.2.1.7).

To use BLE for distance measurement, the broadcaster is advertising packets. These packets are received by the observer, which scans for BLE packets. The observer evaluates the signal strength and provides the RSSI to the application.

> *"The RSSI represents the power of a signal received from a remote transmitter measured in dBm, it is inversely proportional to the square of the distance from the transmitter and depends on the type of device or antenna. Its value can be used to estimate the distance between transmitter and receiver, but it is very sensitive to the environmental conditions."* [53]

The theoretical correlation between the received and transmitted signal power can be derived from the Friis equation [24], [53]:

$$P_R = P_T \frac{G_T G_R \lambda^2}{(4\pi)^2 d^n} \tag{2.3}$$

where P_T and P_R are the transmitted and received signals in Watt; G_T and G_R are the gain of the transmitting and receiving antennas; λ is the wavelength; d is the distance in meters; n represents the propagation exponent, which has typically a value of 2-4 (e.g. 2 for free space), depending on the environmental conditions [53].

For evaluating and testing the reproducibility, the single-board development kit Nordic nRF52-DK was used in advertising mode, whereas a Raspberry Pi 3 Model B V1.2 was scanning for BLE devices. On the broadcasting device, the parameters advertising interval and transmission power have to be set. The Nordic nRF52-DK is able to send advertising packets in intervals between 100 ms and 10.24 s. The supported values for the transmission power are: -40 dBm, -20 dBm, -16 dBm, -12 dBm, -8 dBm, -4 dBm, 0 dBm, +3 dBm and +4 dBm (as stated in the GAP interface specification). For distance estimation with BLE, the advertising interval was chosen as short as possible in order to reach a high timing resolution.

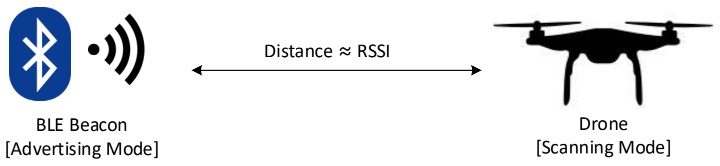


Figure 2.11: BLE distance measurement

The Raspberry Pi in scanning mode, contains a Bluetooth (BT) module which is connected via Universal Asynchronous Receiver Transmitter (UART) to the system bus. The communicate between the host and the controller part is done via a standardized Host Controller Interface (HCI). The Linux distribution Raspbian provides a set of libraries and tools called *hcitools*, which is used to configure Bluetooth connections [55]. To enable the scanning mode on the Raspberry Pi, the command *hcitool lescan* has to be executed. To inquire remote devices and monitor gathered Bluetooth packets, the command *./btmon* was used [55].

2.4 Infineon Larix Drone

Since Infineon Technologies AG is working on ready-to-use multicopter solutions where almost every integrated component is out of Infineon's product portfolio, it was obvious to work with an in-house solution in the context of this project. The Larix EDU Copter was chosen, because it has an on-board Raspberry Pi interface, which is used for further extensions. Figure 1.1 depicts Infineon's multicopter Larix EDU.

2.4.1 Hardware Architecture

The most important parts of the drone are depicted in Figure 2.12. The heart of the Larix EDU is the Infineon XMC4500. The XMC4500 is a microcontroller based on the ARM Cortex-M4 processor core and is optimized for industrial connectivity, industrial control, power conversion, as well as for sense and control [35]. The XMC4500 contains the main software, the flight control software CleanFlight, which can be updated, configured and tuned by the Google Chrome app CleanFlight Configurator (details in Section 2.4.2). The controller communicates with an Inertial Measurement Unit (IMU), which provides the different sensor values, such as acceleration information or rotational speed information, to the controller. Out of these sensor values, the current orientation of the multicopter can be calculated. In order to execute a movement in a certain angular direction such as roll, pitch or yaw as sketched in Figure 2.13 (left), the reference values from the remote control are taken, to adjust the rotational speed of the four motors. To bring the total

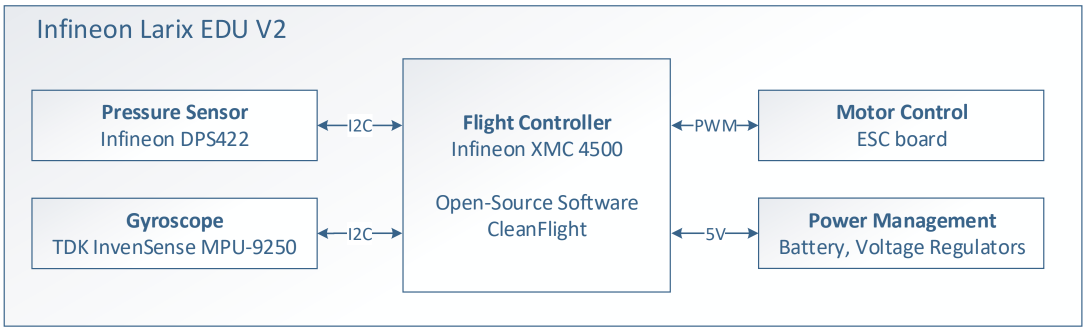


Figure 2.12: Drone hardware architecture

angular momentum of the copter to zero, the each two diagonal located motors have to spin in the same direction, which is depicted in Figure 2.13 (right).

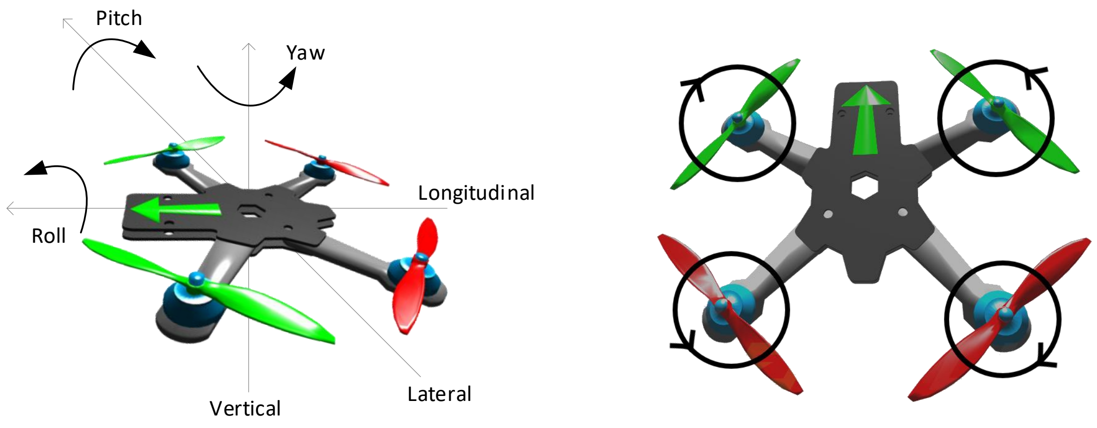


Figure 2.13: Aircraft principal axes and multicopter motor directions

On the Infineon Larix EDU, the Infineon pressure sensor DPS422 is used for the stabilization of the altitude. The DPS422 is a digital barometric air pressure sensor with high accuracy and low current consumption, which can measure pressure and temperature [36]. The pressure and the altitude are indirectly proportional, meaning the pressure decreases if the device is increasing its altitude, and the measured pressure increases if the device decreases its altitude.

Another important component for drones is the gyroscope, which feeds the IMU to calculate the current orientation. On the Larix EDU the multi-chip module MPU-9250 is integrated. It consists of two dies, where one die contains the 3-axis gyroscope and the 3-axis accelerometer, and the other die houses a 3-axis magnetometer [31]. The MPU-9250 is a 9-axis motion tracking device combined with a Digital Motion Processor (DMP) which relieves the host processor from calculating motion algorithms.

The Larix EDU provides an Pulse Width Modulation (PWM) interface for Electronic Speed Control (ESC) for each of the four motors. That means the ESC controls the rotational speed of the motors according to the data received on the PWM interface. PWM is a technique which creates a rectangular pulse with varying width. Typically the pulse width for a motor speed from 0-100 % is 1-2 ms, with a frequency of 50 Hz which results in a repetition interval of 20 ms. The motor control is done in a separate board, mounted below the main board of the Larix EDU and connected with a 4-in-1 ESC connector.

The UAV used for this thesis is powered with a 12V Lithium-Polymer (LiPo) battery. On the flight controller board, voltage converters are integrated to provide 5 V respectively 3 V for the electronics.

2.4.2 CleanFlight

CleanFlight is an open-source software, which provides a flight controller firmware and related tools such as the CleanFlight Configurator, which is a Google Chrome app based software. The firmware has been ported to the XMC4500 in this case. The CleanFlight Configurator provides simplified updating, configuring and tuning of the flight controller firmware. Further, the configuration tool shows sensor values such as pressure values or data from the gyroscope and the magnetometer. The data gathered by the receiver is also displayed visually within the Configurator. The connection between the flight controller and the CleanFlight Configurator which runs on a PC, is established via USB.

2.4.3 Drone Control

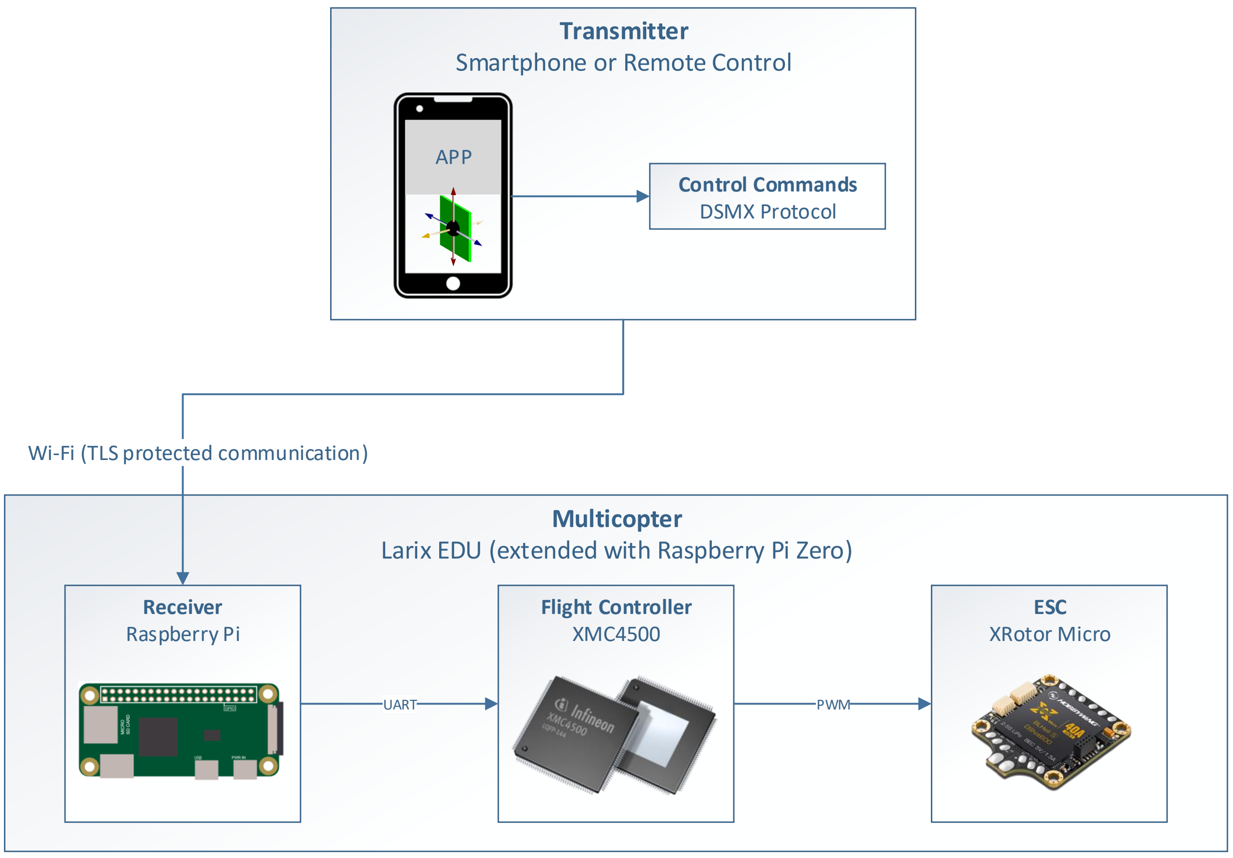


Figure 2.14: Control flow for Larix EDU controlled by smartphone

The Infineon Larix EDU can either be controlled by smartphone or by a standard remote control. In each case the commands are formatted according to the Spektrum

DSMX protocol. Spektrum is a company which produces Radio Control (RC) equipment, who developed the DSMX as well as its predecessor DSM2 which are both 2.4 GHz based transmission protocols. The main difference between those two protocols is the channel selection. Where the DSM2 protocol can only switch between two random channels, the DSMX protocol is more resistant to noise and interference, because the reliability is improved by using more agile frequency channel switching algorithms. The transmitter is sending a 16-byte data packet, formatted according to the Spektrum DSMX protocol, in periodic time intervals (e.g. 11 ms or 22 ms) to the receiver [39]. The 16-byte data packet contains the control information for roll, pitch and yaw of the multicopter, as well as the speed and arming information for the rotors.

In Figure 2.14, the control flow for the Larix EDU controlled via a smartphone application is depicted. The smartphone application uses the gyroscope to control roll and pitch, and sliders to control the rotor speed and the yaw. Out of those values, the 16-byte data packet for the Spektrum DSMX protocol is build and sent via TLS secured Wi-Fi channel to the receiver.

In this case, the receiver is a Raspberry Pi, which is mounted on the multicopter. The received command is relayed via the Raspberry Pi to the flight controller board, which runs the flight controller software CleanFlight, via UART, with 115200 bps, 8 bits, no parity and 1 stop bit. The flight controller board is connected via a 4-in-1 ESC connector to the motor control board, where a PWM signal is transmitted in order to control the four rotors.

Chapter 3

Design

This chapter describes the general use case and the design of the software and hardware architecture of the proposed solution.

3.1 Overview

3.1.1 Use Case

The general use case for authenticated steering of a drone is depicted in Figure 3.1. The user respectively the pilot wants to steer a drone. In order to be allowed to do that, the drone has to authenticate itself with information about the drone and the pilot against a flight control server, which is managed by an authority. If in any case the pilot is doing something not allowed in the context of steering the drone, the flight control server is informing or disciplining the pilot. This can be handled in various different ways, such as taking away the control of the drone. Details about the chosen approach can be found in the subsequent chapter.

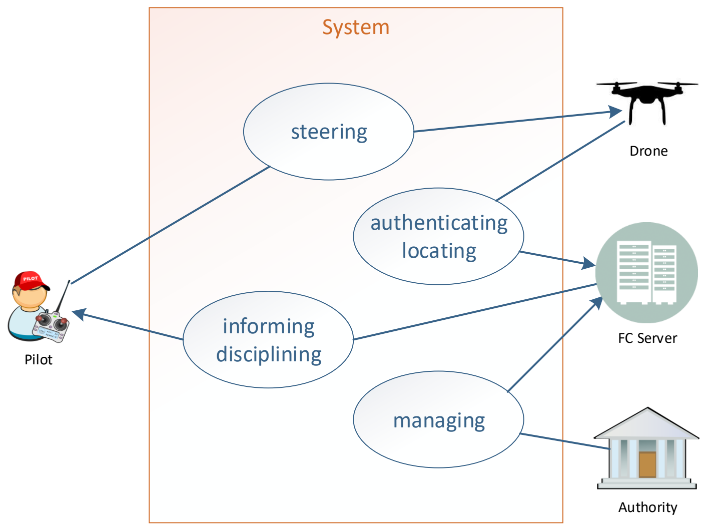


Figure 3.1: Use case for authenticated drone steering

3.1.2 System Overview

Since the drone market is increasing enormously, safety critical problems are coming up. The most important aspect is preventing drones respectively their pilots from flying into or over critical zones or areas such as airports, power plans, prisons etc. In this document, the prohibited zones will be called no-fly zones. In Figure 3.2 the idea behind no-fly zones is sketched. To solve that problem, the UAVs have to be tracked and identified, because there has to be a responsible person (pilot) behind each drone. As already mentioned in Section 2.1.1, there are many discussions about regulations for these aspects but there are neither final rules which have to be followed by the drone owner, nor by the drone manufacturer.

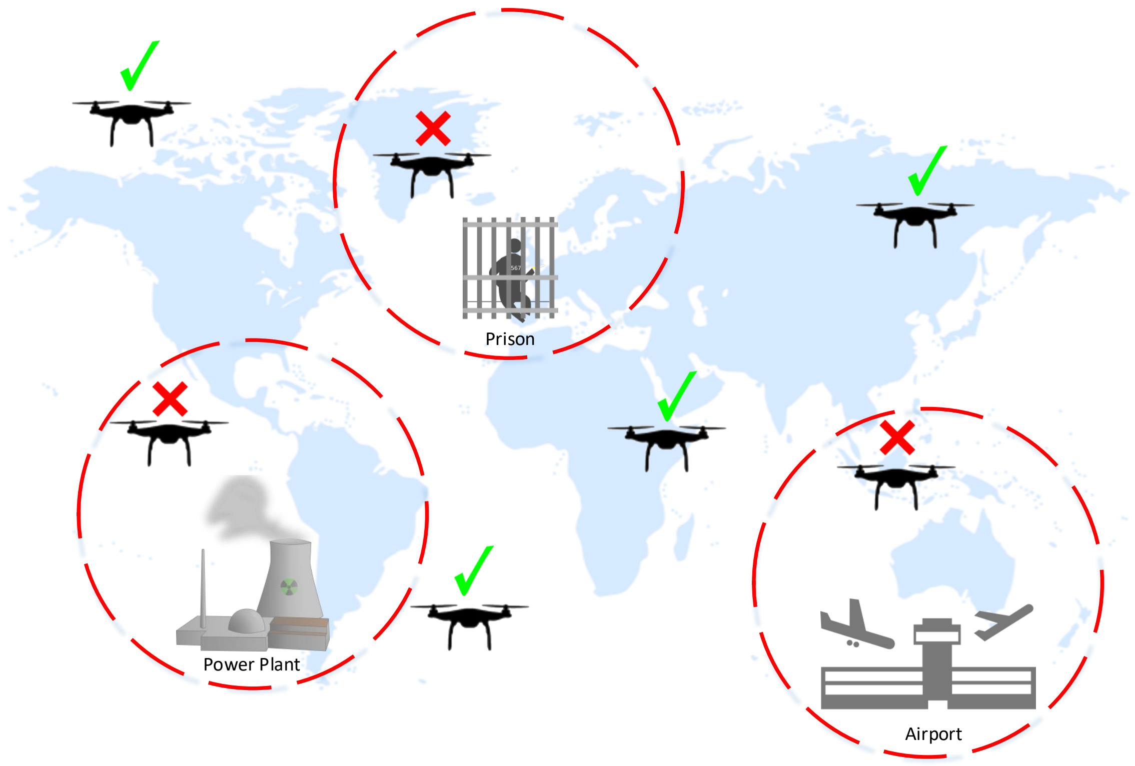


Figure 3.2: No-fly zones

Due to the fact that most UAVs are too small to be detected by classical aircraft tracking solutions, which are used to detect planes, each drone could be equipped with a transponder such as a Automatic Dependent Surveillance - Broadcast (ADS-B). This solution would load to a spectral problem. To mitigate this problem, the transmit power could be lowered, but this results in a smaller detection range. Further, the flight control data which is broadcasted by the transponder is neither secured nor authenticated.

This means the existing methods which are currently in use for larger objects are not capable of handling such enormously increasing number of objects in the air. As already mentioned in Section 2.1.3, there are already companies which are developing drone identification systems, but each has its disadvantages which should be solved in the system proposed within this work.

In order to develop a global system, a global communication network is necessary. A widely spread communication network is LTE, which is used for the drone identification network proposed within this work. A big advantage for using a global system is, that not every no-fly zone has to be equipped with a detection or tracking system such as DJI's AeroScope (Section 2.1.3.1). Figure 3.3 depicts the connection overview of the proposed secured drone identification system. The basic idea of the new system is to connect the drone to the internet via LTE over an MNO. On the other side, there is a flight control server which is connected to the internet as well. Based on the LTE network connection, the drone establishes a TLS protected communication channel to the flight control server, which is controlled by an aviation authority. The secured communication channel is used to send information, such as location information and identification data from the drone to the flight control server. Further, this secured communication link can be used to send steering data from the flight control server to the drone, for example to execute a safe landing of the drone, if a no-fly zone has been entered.

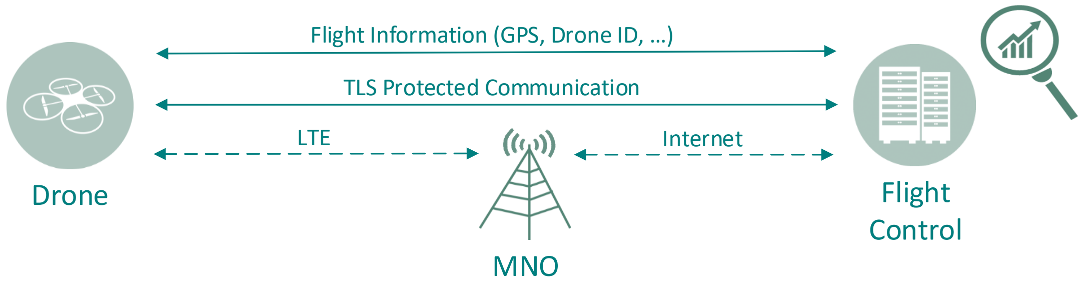

Flight Information (GPS, Drone ID, ...)
TLS Protected Communication
LTE
Internet
Drone
MNO
Flight Control

Figure 3.3: Connection overview of secured drone identification system

The goal of this work is to develop a proof-of-concept solution for a secured and reliable drone identification system, based on standardized and well established protocols and security mechanisms. An example for a standardized protocol is TLS, which is widely spread in applications such as internet banking and secured web browsing. Further details about standardized methods which are used in the proposed system for various purposes can be found in the corresponding chapters.

3.2 Communication Protocol Stack

In Figure 2.6 (Section 2.2.3), a general communication protocol stack for secured communication is sketched, according to the OSI model. Focusing on the proposed drone identification system, Figure 3.4 depicts a more detailed protocol stack structure together with the used techniques for each layer.

3.2.1 CBOR

CBOR is an RFC standardized data serialization format with efficient message encoding, compared to other serialization formats. This means that the overhead added to the payload is fairly small, which is an important fact, if messages should be sent on a wireless channel several times every second, as in the case of the proposed drone identification system. With public available and well established software libraries for encoding and decoding structured messages with CBOR, it is possible to parse arbitrary data structures without putting effort in extending the available software libraries, which are available in common programming languages such as C or Python. As pointed out in Table 2.7, CBOR

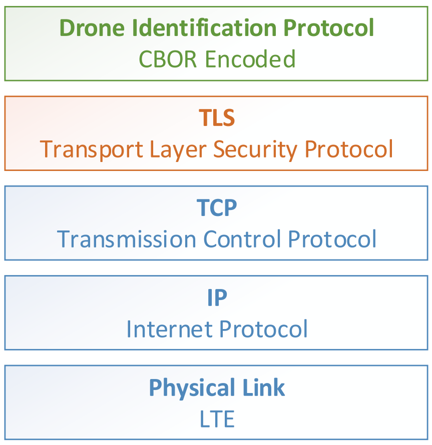

Drone Identification Protocol
CBOR Encoded
TLS
Transport Layer Security Protocol
TCP
Transmission Control Protocol
IP
Internet Protocol
Physical Link
LTE

Figure 3.4: Communication protocol stack for drone identification system

has the least encoding overhead compared to the other data serialization formats. Further, CBOR is with 0.045 ms and 0.041 ms also faster for encoding and decoding of a sample data set, which consists of GPS data (latitude, longitude, attitude) and an identifier. Due to the fact that the next layer in the communication protocol stack is the TLS layer, it would not even be necessary to transmit an identifier with each message, because the drone has to transmit its identifier (a certificate in that case) during the TLS handshake, which is explained in greater detail in the subsequent chapter. That means, during the connection establishment between the drone and the flight control server a significant data transfer is done in order to complete the TLS handshake, but the messages sent during operation (several times per second) get more compact.

3.2.2 TLS

TLS is a well established and widely used protocol to provide privacy, data integrity and authenticity between two communicating parties [34]. TLS consists of two layers, the TLS record protocol and the TLS handshake protocol. The basic properties of the record protocol are the privacy and the reliability of the connection, where the privacy is established with a symmetric cryptography algorithm (e.g. AES) and the integrity and authenticity are ensured by a MAC (e.g. based on SHA-256) or Authenticated Encryption with Associated Data (AEAD) cipher.

3.2.2.1 TLS Record Layer Encryption

For block ciphers, such as AES, different modes of operation are available. To name common examples for those modes, the recommendation from National Institute of Standards and Technology (NIST) defines five confidentiality modes for the use with an underlying symmetric key block cipher [19]:

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR)

Based on those modes of operation, there exist extended modes which provide not only confidentiality, but also authenticity and integrity. These modes are called authenticated encryption modes. A widely used example in this context is called AES-GCM [80]. Galois/Counter Mode (GCM) is a mode of operation, based on the CTR mode, which already comes with confidentiality. The cipher text with GCM is produced in the same way as in the common CTR mode as sketched in the upper part of Figure 3.5. This works by creating a counter block, where the sequential block number is concatenated (alternatively added or XORed) with the Initialization Vector (IV) and encrypted with the block cipher (AES in this case), which results in the keystream. The cipher text block is then computed by XOR-ing the plain text block with the keystream [29].

Compared to the common CTR mode, the GCM also provides integrity and authenticity, which is achieved by generating a so called authentication tag. This is depicted in the lower part of Figure 3.5 [80]. To generate such an authentication tag, the GHASH function is used. Detailed information about the definition of the GHASH function can be found in [15] and [80]. Figure 3.5 sketches the authenticated encryption algorithm AES-GCM with a single block of additional application data (labeled as Auth Data 1) and N blocks of plain text. *"AES(K) denotes the block cipher encryption using the secret key K, and $GHASH_H(X)$ denotes Galois field multiplication in $GF(2^{128})$ by hash key H, and incr denotes the counter increment function."* [80]

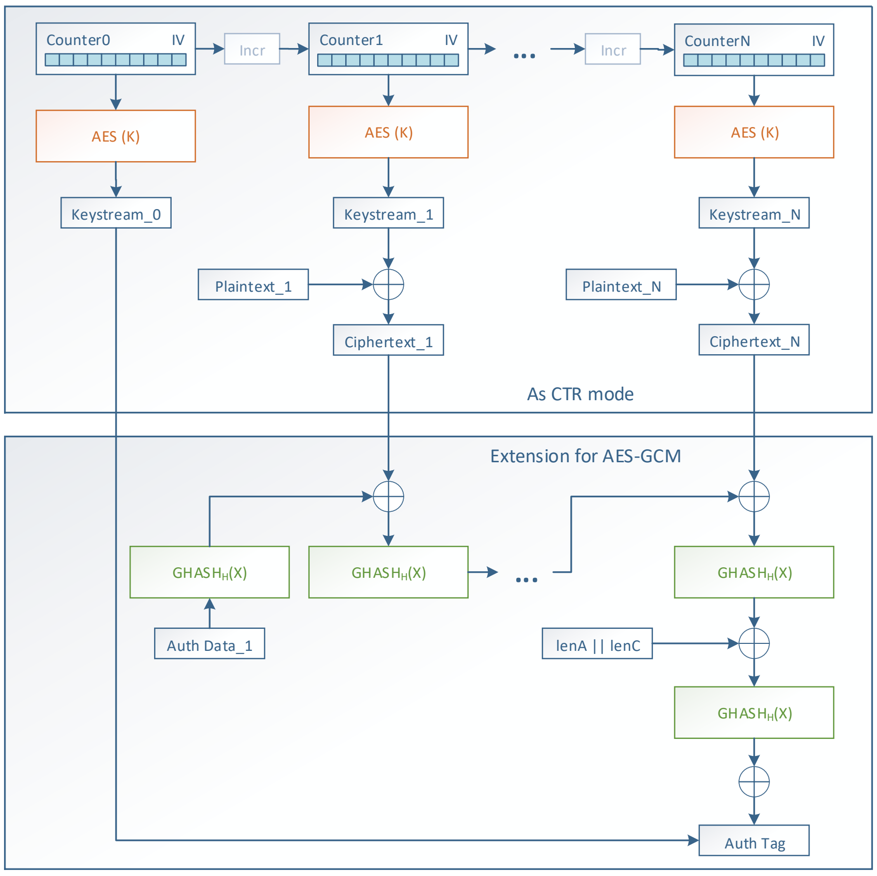


Figure 3.5: AES-GCM algorithm (modified from [80])

3.2.2.2 TLS Handshake Layer Sequence

Figure 3.6 sketches the messages exchanged by the TLS handshake protocol, where the most important steps, regarding to the drone identification system, are explained in detail here.

The *ClientHello* message is sent from the client to the server and contains cryptographic information including the cipher suites which are supported by the client. The different cryptographic primitives, together with corresponding examples, a cipher suite typically contains are [48]:

- Key Exchange
  - RSA, DH(E), ECDH(E)
- Authentication
  - RSA, DSA, ECDSA
- Encryption
  - AES, CAMELLIA, DES
- Hash
  - SHA-1, SHA-256, MD5

If a client connects to a server, the client has to validate the identity of the server during establishing a connection. This is done by checking the server *Certificate*, which is sent right after the *ServerHello* message, where the *ServerHello* message is an answer to the *ClientHello* message during the TLS handshake. The sent server certificate is the first certificate in a list of certificates, called certificate chain. The server *Certificate* is followed by an arbitrary number of intermediate certificates, where each certificate has been used to sign the previous certificate in the list. *"The client verifies the chain until it reaches a certificate which is signed by a trusted root certificate which is contained in the clients certificate storage."* [82] If the last certificate cannot be verified by the client, the authentication is not successful and therefore the TLS handshake is aborted and a TLS alert message is sent. In addition to the verification of the server, the client can authenticate itself against the server. This is optional but used in the context of the drone identification system to provide authentication for the drone (client in this case) against the flight control server. If the drone authentication is done during the TLS handshake procedure, the application does not have to send the authentication data within the periodically sent messages. To use the optional available TLS feature during the TLS handshake, the server has to send a client *CertificateRequest* message. As an answer to this request, the client sends the certificate embedded in a message with the same structure as the server *Certificate* in the previous step. The validation of the client *Certificate* works in the same way as the validation of the server *Certificate* message. The retrieval of the certificates is illustrated in the following chapter. Detailed information about the other steps happening during the TLS handshake are explained in [34]. Implementation details about the TLS handshake are given in Section 4.2.1.2.

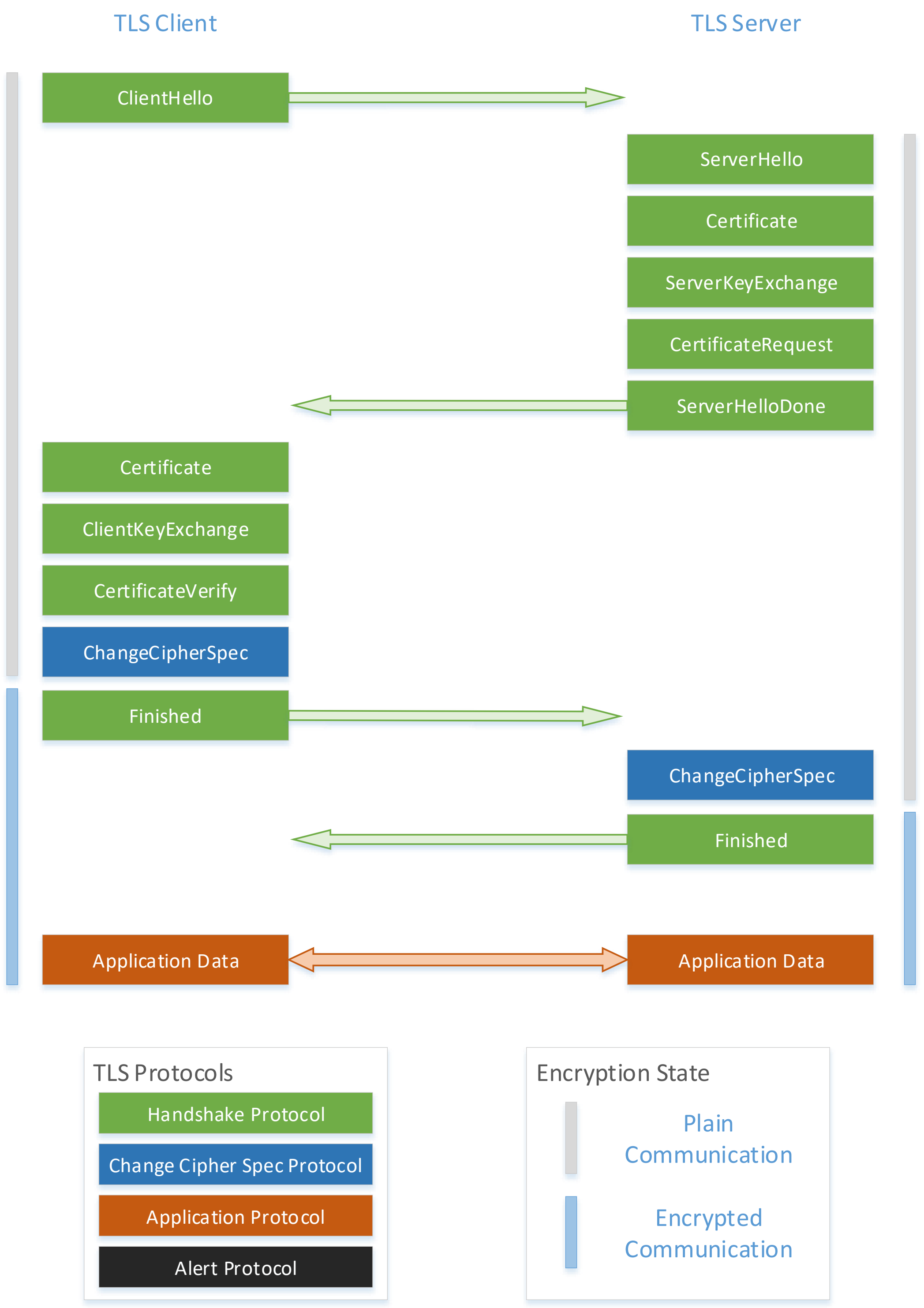


Figure 3.6: TLS handshake overview [82]

3.2.2.3 Certificate Provisioning (Aviation Authority Lookup Service)

In order to perform a TLS handshake, the Certificate Authority (CA) certificates which are stored in the certificate storage, have to be retrieved beforehand. The procedure for the certificate retrieval process is split into two parts, the client and the server side. The client side consists of the drone itself and the drone manufacturer CA. The server side contains the flight control server, the authority or government CA and an aviation authority lookup service, which is proposed within this project.

In Figure 3.7 the certificate retrieval process is sketched, which is designed in a similar way as the Domain Name System (DNS) lookup service. The basic idea is, that the drone is requesting the domain and the CA certificate of the regional authority from the aviation authority lookup service, to later establish a TLS secured communication channel to a flight control server for authentication and location tracking. In order to perform these two steps, certain steps are necessary beforehand, because the drone needs to know the domain or IP to the aviation authority lookup service. Those steps are later on called offline steps, because this is done during manufacturing of the drone and only performed once. Further the drone needs the certificate of the lookup service in order to ensure that it is connecting to a trusted server.

Since the goal of the system is that the drone has to authenticate itself against an authority, the certificates have to be generated according to discretions of this authority. Those certificates are given to the flight control server as well as to the aviation authority lookup service, where the certificates are put into a database. The lookup service also gets the domain from the flight control server which is forwarded by the authority. Each region has its own aviation authority. Each authority is registered at the global lookup service. The global lookup service contains the domain as well as the certificate of the regional aviation authority. Further, each drone manufacturer has a certificate authority with the responsibility to create a certificate for each manufactured drone. In addition to the drone certificate, the drone manufacturer has to store the domain of the global lookup authority on the drone's and put the certificate of the global lookup authority in the drone trusted certificate storage. All the steps explained up to now, are steps which have to be done offline (highlighted in green in Figure 3.7). That means all these steps happen before the actual authentication of the drone against the flight control server can be performed.

After these offline steps, each drone has a CA certificate as well as a IP or domain of the aviation authority lookup service and a certificate for authentication. On the other side, the flight control server has a certificate of the regional authority CA. One step which is seen as a precondition, because it is out of scope and therefore not explained, is the process of getting the drone manufacturer CA certificate onto the flight control server for validating the drone's authentication certificate. This could either be implemented with a separate lookup service, with a database on the flight control server which is filled beforehand or also by storing the certificates on the server of the aviation authority lookup service.

The steps for the actual authentication process of the drone are sketched in orange (Figure 3.7). In the first step, a TLS secured connection between the drone and the lookup service is established, where the trustworthiness of the server of the lookup service is checked by the drone with the CA certificate, which is already stored on the drone. Trough this secured connection the domain and the CA certificate of the regional authority is requested. In order to get the right domain to the regional authority's flight control server, the lookup service needs to know the location of the drone. To get the location information, either the GPS data can be sent by the drone or the lookup service can gather the location information from the IP address of the drone. Another possibility to get the

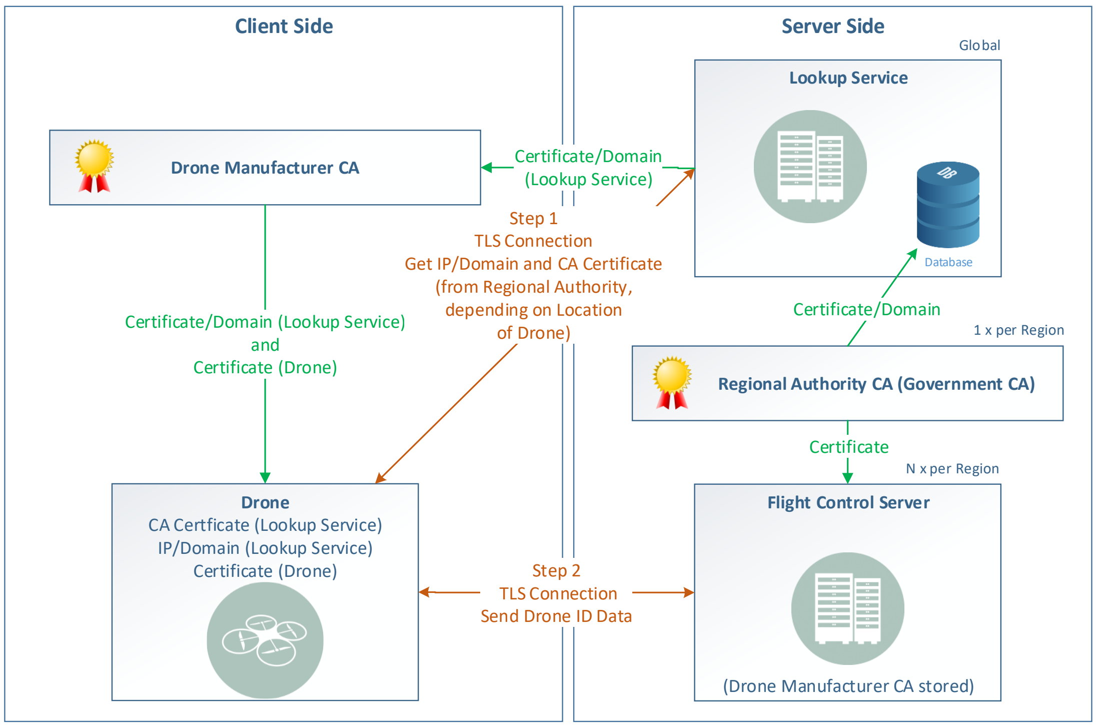

Client Side
Server Side
Global
Lookup Service
Database
Drone Manufacturer CA
Certificate/Domain (Lookup Service)
Step 1
TLS Connection
Get IP/Domain and CA Certificate (from Regional Authority, depending on Location of Drone)
Certificate/Domain
1 x per Region
Certificate/Domain (Lookup Service) and Certificate (Drone)
Regional Authority CA (Government CA)
Certificate
N x per Region
Drone
CA Certficate (Lookup Service)
IP/Domain (Lookup Service)
Certificate (Drone)
Flight Control Server
Step 2
TLS Connection
Send Drone ID Data
(Drone Manufacturer CA stored)

Figure 3.7: Certificate retrieval

location of the drone, would be to extract information about the connected LTE cell.

After receiving the domain and the certificate of the regional authority, the secured connection to the flight control server can be established. On one hand the flight control servers identity is validated by the drone with the certificate received in the previous step, and on the other hand the drone is authenticating itself against the flight control server with the drone certificate which is stored on the drone during the manufacturing process. The flight control server is validating the drone with the certificate of drone manufacturer CA which is already on the flight control server (precondition).

3.2.3 TCP/IP

In order to transport packets between two communicating parties over a packet switched network such as IP, a transport protocol has to be selected. There are two common, well established and widely used protocols, the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). The biggest difference between those protocols is that TCP is a connection oriented and UDP is a state less protocol. TCP guarantees an ordered packet delivery, which is done by adding a sequence number to each packet. Further, TCP is also providing retransmission of lost packets. The detection of lost packets is done by acknowledging each packet. Another property of TCP is that data is sent in a continuous stream. That means there is no maximum message size, compared to UDP which is message oriented. In UDP corrupted or lost packets are not retransmitted and the application is not informed about any loses. Since TLS requires that packets are transmitted in the right order, we need a reliable transport protocol. In addition certain TLS handshake messages exceed the maximum message size, therefore a continuous data stream is beneficial. That means TCP is the right choice for this system. The Internet

Protocol (IP), which is the most used protocol for the internet layer, is used to transport packets between two hosts based on the IP address, which is in the packet header.

3.2.4 LTE

The base of the communication protocol stack (Figure 2.6) is the physical link, which in general can either be wired or wireless. Since the drone is a flying object, only a wireless link is useful in practice. To cover as many regions as possible with the proposed drone identification system, a widely available physical link should be used. Since LTE is the state-of-the-art wireless communication technology with a large areal coverage, especially near civilization, it is chosen for the UAV side of the drone identification system. Due to the high bandwidth (up to 20 MHz) as well as the high down- and uplink data rate (uplink: 75 Mbps, downlink; 300 Mbps), LTE can be used for further use cases in addition to the drone identification system, such as cloud connectivity or video streaming [52]. Further it would also be possible to use the LTE connection to remote control a drone beyond line of sight.

3.3 Hardware Design

This chapter describes the hardware design of drone, which is used in the developed proof-of-concept solution for a secured and reliable drone identification system. The basis for this hardware system is Infineon's flight controller board Larix EDU V2 (depicted in Section 2.4.1).

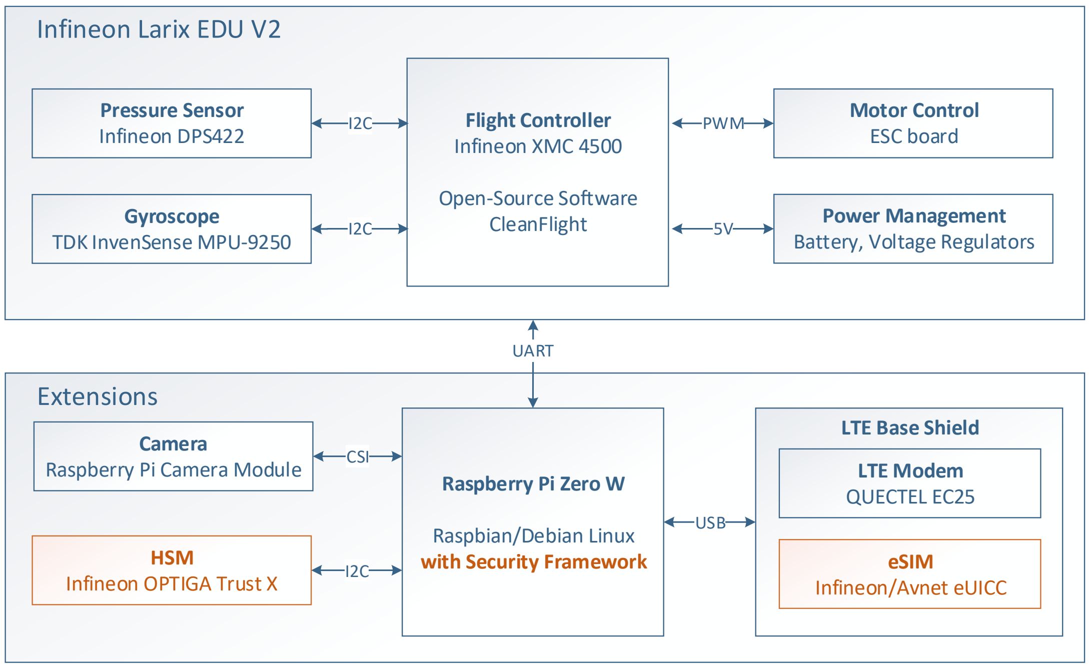


Figure 3.8: Drone hardware design

3.3.1 Components

3.3.1.1 Raspberry Pi

The flight controller board contains a Raspberry Pi pin header, to easily extend the system with the single board computer Raspberry Pi. The Raspberry Pi hosts the application, which is explained in detail in the corresponding software design and implementation chapter (Section 3.4.1, Section 4.2.1). The variant used for this system is the Raspberry Pi Zero W, which is more compact compared to the Raspberry Pi 3. The drawbacks of the Raspberry Pi Zero W are the lower processing power and less connection possibilities. This is not a problem because the application running on the Raspberry Pi has decent requirements regarding processing power. Further, there is no need for connecting peripheral devices such as display, mouse or keyboard during normal operation. Therefore the less connection options are not a problem. The Raspberry Pi Zero W is connected via a UART interface (4 wires: Power supply, ground, transmit and receive pin). The Operation System (OS) running on the Raspberry Pi Zero W is Raspbian, which is the official supported OS. The Raspberry Pi Camera Module is connected via Camera Serial Interface (CSI) to the Raspberry Pi, which can be used for video streaming or steering beyond the line of sight. An explanation of the Camera Module would be out of scope within this context, but detailed information can be found in [68].

3.3.2 LTE Base Shield

In order to connect the drone, respectively the Raspberry Pi, into the LTE network, an LTE modem as well as a SIM is necessary. That means those two components need to be connected to the Raspberry Pi. A simple way to do that, is using a Raspberry Pi LTE shield, which acts as a bridge for an LTE module. On this shield, the LTE module and the SIM can be connected. The functionality was fully tested with a shield which is freely available at the market (Figure 3.9). For space saving reasons, the base shield was modified for this project. The LTE module can be connected with the standardized Mini PCI Express connector, and the SIM slot is build for the widely used form factor 3FF (Micro SIM). As depicted in Figure 3.8, the base shield is connected via Universal Serial Bus (USB) to the Raspberry Pi. Further details can be found on the manufacturer website [75]. Details about the adopted base shield can be found in the implementation chapter in Section 4.1.1.1.

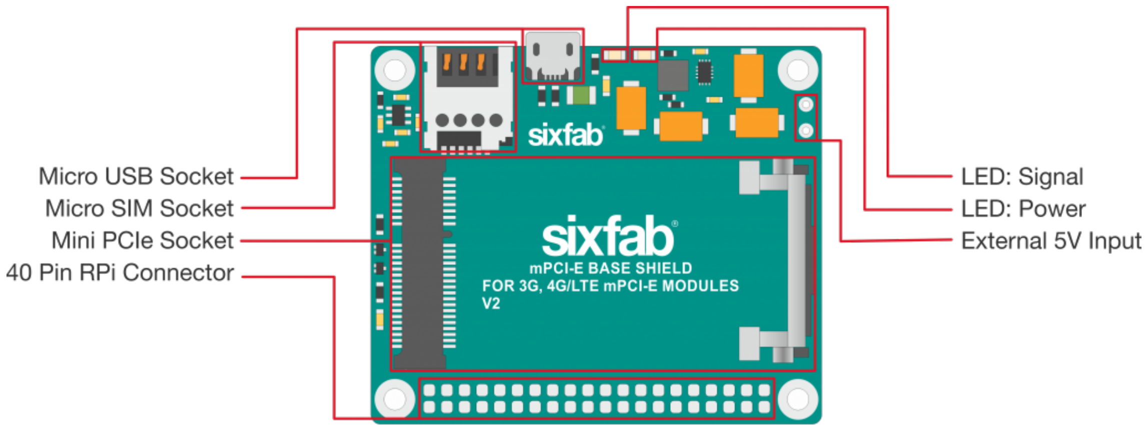


Figure 3.9: Raspberry Pi LTE shield [75]

3.3.2.1 eSIM

Details about the eSIM technology are described in the corresponding state-of-the-art chapter (Section 2.2.2.2). One of the purposes of this project is to showcase the eSIM, especially the main advantage of the eSIM, the simple change of an MNO. To implement that, at least two different MNOs are necessary. Due to the fact that most MNOs are currently ramping up the provision of Profiles for the eSIM, there are only test Profiles available up to now. In order to showcase an MNO switch, the Avnet eUICC test pack was chosen ([5]), because this test pack provides two different MNO test Profiles. Another important fact for choosing the Avnet eUICC test pack is, that Avnet is an Infineon authorized distributor and therefore there is a support link between Infineon Technologies AG and Avnet [4]. The test pack contains eSIMs with different form factors, which are the form factor 2FF (Mini SIM) and MFF2. The chips used in the eSIMs are Infineon security controllers, based on 32-bit ARM cores.

As described in the state-of-the-art section of the eSIM (Section 2.2.2.2), a Profile contains applets. One further potential use case could be an applet which allows storing keys and certificates to build up a secured communication and/or identify against a third party. Due to the fact, that an applet has to belong to a Profile, it is necessary to cooperate with a mobile network operator to realize an applet implementation. This procedure is out of scope in this context, but for the future, it could be a possible solution for several use cases.

3.3.2.2 LTE Modem

Since the LTE connection, respectively the authentication at an MNO should be established with an eSIM, the LTE module has to be compatible with the eSIM. In general eUICCs should behave as its predecessor UICC, because it is an GSMA standardized technology, but since the eUICCs are in an early phase, it is possible that problems occur. In order to avoid that, an Avnet pre-qualified LTE modem was chosen. The selected LTE base shield requires an LTE modem on a Mini PCIe module basis. The Quectel EC25 Mini PCIe Cat 4 module was chosen (Figure 3.10), since it is pre-qualified by Avnet and the form factor fits to the remaining hardware parts without the expense of redesigning the module. Cat 4 represents one out of eight available LTE user equipment categories which differ by data rates (Cat 4 - uplink: 50 Mbps, downlink; 150 Mbps). That means with the chosen hardware for the LTE part of the system, a plug and play solution was selected, even though the base shield was redesigned for space reasons. As depicted in Figure 3.10, the Quectel EC25 LTE module includes three antenna interfaces, the main antenna interface, the GNSS antenna interface and a receiver diversity antenna interface. GNSS is a collective term for satellite based global navigation systems such as GPS. Since within this solution, GPS is not used for localization, there is no GNSS antenna connected to the corresponding antenna interface. For connecting to a cellular network (LTE in this case), an antenna has to be connected to the main antenna interface. To improve the Signal-to-Noise Ratio (SNR), a diversity antenna can be connected. The principle of diversity is exploiting statistically independent respectively uncorrelated paths for communication [46]. Based on this principle, the reflection effect as well as the multipath signals of the propagated radio signal can be reduced. For the proposed solution Printed Circuit Board (PCB) antennas (dimension: 37 mm x 7 mm x 1 mm) were used for the main and the receiver diversity antenna [73].

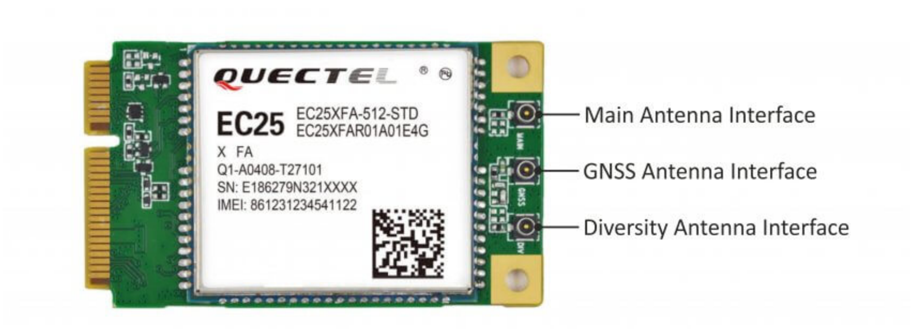


Figure 3.10: Quectel EC25 Mini PCle LTE module [74]

3.3.3 HSM

Infineon's OPTIGA Trust X, which has the form factor PG-USON-10-2, was used as HSM for the proposed solution. Details about the OPTIGA Trust X are depicted in Section 2.2.3.3. The HSM is connected via I2C. This bus interface can be used to connected further sensors. The OPTIGA Trust X is physically located on the redesigned LTE base shield, which is explained in detail in the corresponding implementation section (Section 4.1.1.1).

3.4 Software Design

In this section, the software design of the developed proof-of-concept solution for a secured and reliable drone identification system is described. This solution consists of two parts, a client (drone) and the flight control server.

3.4.1 Client Software (Drone)

Figure 3.11 depicts the software design for the client software running on the drone. The main components are the application and the TLS block. Additionally a TCP socket is necessary as a transport channel, in order to establish a reliable connection via the LTE network, which is sketched in yellow (Figure 3.11). To execute security mechanisms with support of the HSM, additional software blocks are necessary to provide an interface to the TLS software block. The green highlighted parts of Figure 3.11 sketch the additional hardware devices and the corresponding interfaces to the Raspberry Pi, which is hosting the software.

3.4.1.1 Application

The application which is running on the drone has three main tasks. Since the goal of this project is to develop a secured drone identification system, one task is the drone identification task. This task is gathering location information such as GPS coordinates, and is sending them to the flight control server as depicted in the use case diagram in Figure 3.1. In order to send data (location information, identifier, etc.) over an already established channel, the data have to be serialized before transmission. This means, that structured data objects have to be converted to a stream of bytes beforehand. Since the CBOR data serialization format is used within this system, a CBOR library is necessary in order to encode and decode the structured data.

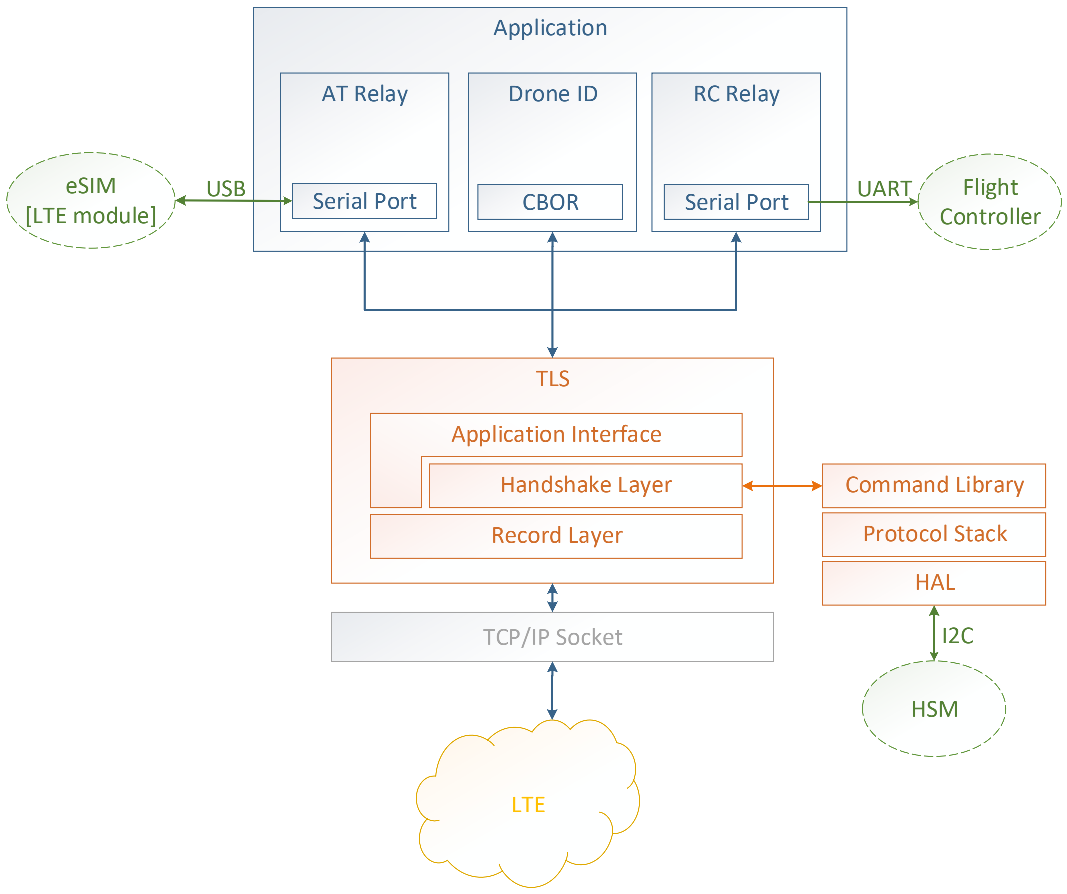


Figure 3.11: Drone software design

As depicted in the use case diagram in Figure 3.1 and its corresponding section (Section 3.1.1), the flight control server should be able to take away the control over the drone from the pilot, to discipline the pilot in case of non-observance of rules or regulations. To handle this, the application which is running on the drone's Raspberry Pi, needs to relay the steering data to the flight controller, which are received from the flight control server instead of the remote control. This task is forwarding the received steering data to the flight controller via a serial port. For this purpose, the flight controller is connected via an UART interface to the Raspberry Pi.

In the developed solution, an eSIM is used to identify the device against an MNO which provides the access to the LTE network as a service. To showcase one of the main advantages of the eSIM, which is the MNO Profile switch, a dedicated task is necessary. Avnet's eUICCs can be administrated not only remotely, but also locally by AT commands. Practically, the Profile switch could be triggered by the drone owner, but for demonstration purposes, it will be triggered by the flight control server. Therefore the task which is running on the Raspberry Pi and responsible for the Profile switch, has to relay the incoming AT commands to the eUICC. This works by forwarding the AT commands via the USB connected LTE base shield. Information about AT commands and the detailed description about the Profile switch can be found in the implementation chapter in Section 4.2.2.1.4.

All the tasks described here, need to communicate with the flight control server. To clearly separate the responsibilities of these tasks, each task is using a dedicated TLS secured TCP/IP socket for communication.

3.4.1.2 TLS

TCP/IP sockets are used in order to establish a connection over the LTE network. According to the designed communication protocol stack (Figure 3.4), TLS has to be implemented on top of the transport layer (TCP in this case).

To provide the security mechanisms to the application, an Application Programming Interface (API) is necessary. The API provides the functionality to perform a TLS handshake and allows the application to read and write from/to the secured channel, which is handled in the record layer of TLS (see Section 3.2.2). Before read and write actions to/from the secured channel are possible, the TLS handshake has to be completed.

As depicted in Figure 2.7, the functionality of the TLS layer is split between the host controller and the secure element, if a HSM is used to relieve the host controller and provide hardware based security for storing key material and performing cryptographic operations. The secure element is taking care of all cryptographic operations which are related to long term keys as for example used by ECDSA. The partitioning between the host controller and the secure element can differ depending to the application. The protocol specified in the TLS standard is implemented in a host library.

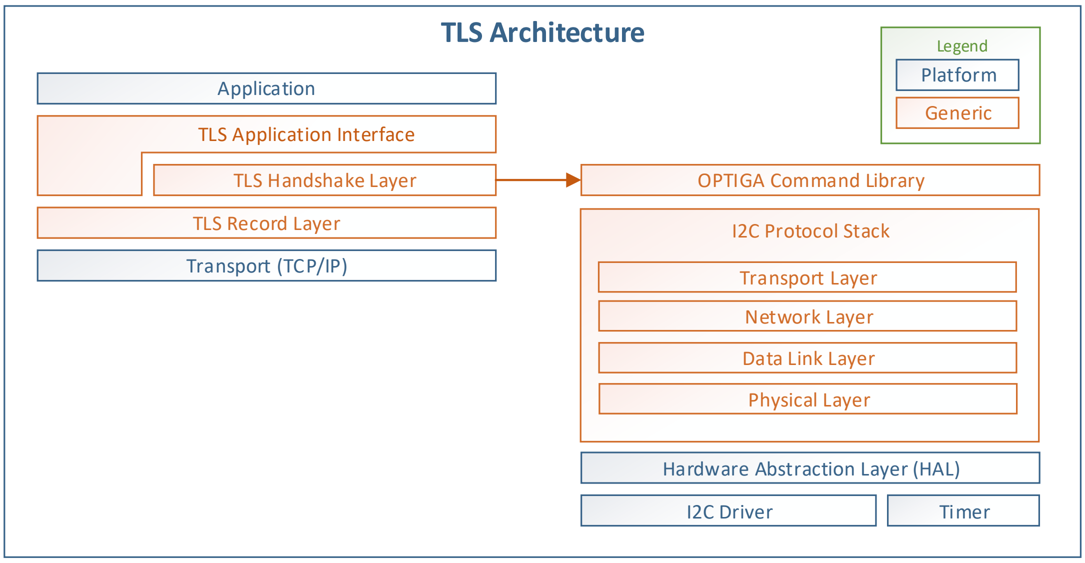


Figure 3.12: TLS host software architecture

Figure 3.12 depicts the detailed TLS host software architecture which can be used in a variety of application not just limited to the use case for proposed solution, where in Figure 3.11 a higher level overview about the TLS software blocks is given. In the developed solution, the security mechanisms for the TLS handshake procedure are outsourced from the host controller to the HSM. The HSM is connected via a serial interface to the host (in this case I2C). Since every platform requires a different driver a Hardware Abstraction Layer (HAL) has been implemented to allow portability of the software. On top of the physical transport layer an I2C protocol stack is necessary to provide features such as reliability and packet fragmentation to the transport layer, which consists of physical, data link, network and transport layer [17]. To trigger an HSM security mechanisms from the application, an HSM specific command library is necessary.

3.4.2 Flight Control Server Software

In Figure 3.13 the software design of the flight control server is depicted. The structure is analogous to the design of the software running on the client (Figure 3.11). The software of the flight control server consists of the application itself, a TLS block for securing a communication channel and a TCP/IP socket in order to establish a connection over the LTE network.

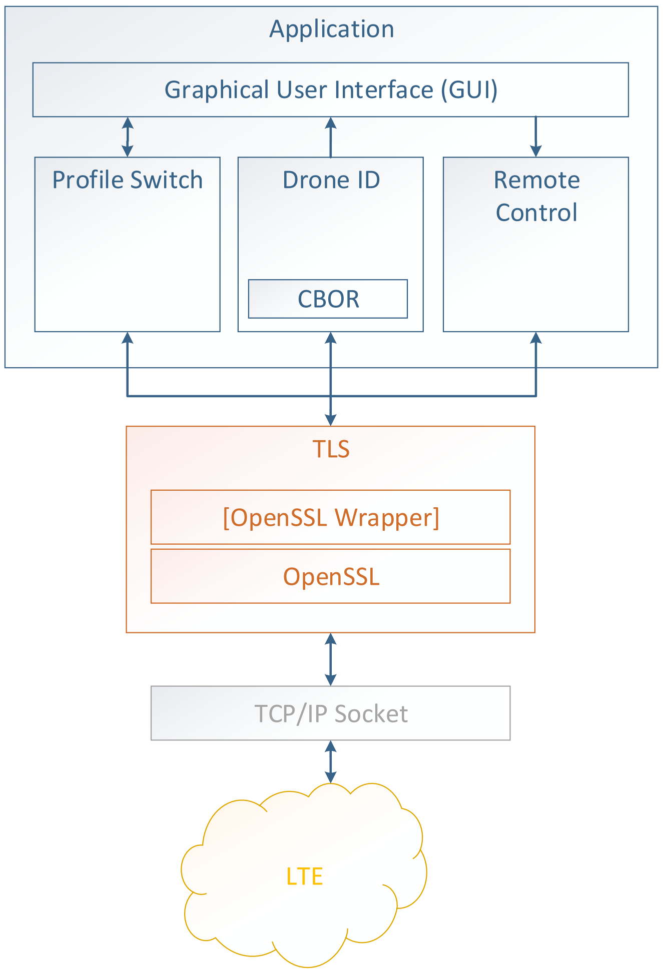


Figure 3.13: Flight control server software design

3.4.2.1 Application

The application contains the counterparts to the three main tasks of the client software described in Section 3.4.1.1 and a Graphical User Interface (GUI) to provide the possibility for visualization and interaction with the user. The flight control server receives the drone identification data (e.g. location information) in a CBOR structured data stream. This information is then decoded and visualized in the GUI. If the no-fly zone is entered by a drone, an alarm is triggered. As explained in the corresponding client software design section (Section 3.4.1.1), the flight control server should be able to remote control the drone. Therefore some basic steering commands can be sent to the drone, which are triggered via an event (e.g. button pressed) from the GUI. In addition to that, an MNO change on the eSIM is supported. To trigger an MNO Profile change, the GUI provides an input possibility but also an info box for indicating the available Profiles, as well as highlighting the enabled Profile.

3.4.2.2 TLS

Since TLS is providing a point-to-point encryption, also the flight control server needs a TLS block. As the focus of the developed solution is not on the server side, this TLS block is fully implemented in software, contrarily to the client side, where the security mechanisms are partitioned between the host controller and an HSM. For this implementation, the *OpenSSL* library is used, because the library contains an open-source implementation of the TLS protocol. Depending on the chosen programming language, a wrapper for the *OpenSSL* library is necessary. As TLS needs a reliable channel, TCP/IP sockets are chosen to establish a communication channel between the client (drone) and the flight control server.

Chapter 4

Implementation

This chapter describes the software and hardware implementation details of the proposed solution for the client (drone) and the flight control server, as well as for the general demonstration setup.

4.1 Hardware Implementation

4.1.1 Client Hardware (Drone)

In the current hardware implementation, the drone hardware consists of stacked modules which are placed on top of the battery and the carbon frame of the drone, as depicted in Figure 4.1. The basic drone components, such as frame, motor controller board and flight control board, were developed from MCI Innsbruck.

As depicted in Figure 3.8, the lowest part in the stack is the motor control board (ESC board). The provided motor control board is the XRotor Micro40A 4in1 BLHeli-S DShot600, which is freely available on the market [30]. On top of the motor control board, there is the flight controller board Infineon Larix EDU V2, which was developed in a cooperation between MCI Innsbruck and Infineon Technologies AG. More information about the basic drone components are depicted in the corresponding state-of-the-art section (Section 2.4.1). The next module in the stack, is the single board computer Raspberry Pi Zero W, which hosts the drone identification software. This part is explained in detail in the corresponding software implementation section (Section 4.2.1). In order to connect the drone, respectively the Raspberry Pi to the LTE network, the top module is the LTE module, This module is plugged into the LTE base shield, which is needed as a bridge between the Raspberry Pi and the LTE module. The LTE base shield was redesigned from [76]. The main components on the LTE base shield are the LTE module, the eSIM and the HSM OPTIGA Trust X. Further details about the redesigned base shield are depicted in the subsequent section (Section 4.1.1.1).

4.1.1.1 LTE Base Shield

The LTE base shield was redesigned to match the form factor of the Raspberry Pi Zero W, which is used in this project. The original shield was designed for the Raspberry Pi 3.

Further, the eSIM with the form factor MFF2 was placed on the base shield in addition to the SIM connector with the 2FF form factor, to show the size difference between them. This aspect is an important advantage of the eUICC. This also allows to test and evaluate different SIMs. To select the eSIM which should be actually used for authenticating

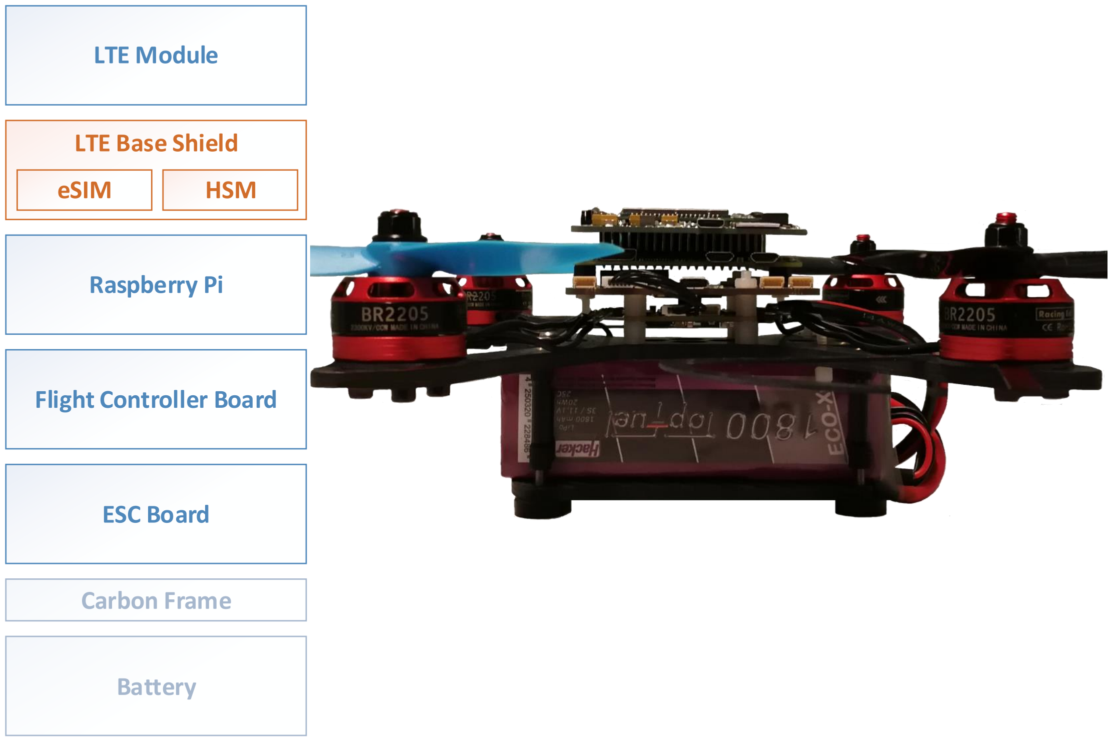


Figure 4.1: Drone hardware stack

against an MNO, a jumper has to be set correctly. The eSIM comes with the capability of changing the MNO without physically replacing the SIM card, which is depicted in the corresponding state-of-the-art section (Section 2.2.2.2). Traditional SIM cards have to be replaced when changing the provider, therefore a SIM connector (or SIM slot) was necessary in the past, in order to provide the possibility to change the card (respectively the MNO). The SIM connector needs to be accessible in order to change the SIM card, and therefore it cannot be placed on any arbitrary place within the device, which leads to restrictions compared to the eSIM. The size and placement aspect is especially in the Internet of Things (IoT) context very important, because those devices tend to be small in physical size.

The main component of the base shield is the LTE module, which is top module in the client hardware stack (Figure 4.1). The chosen module is the Quectel EC25 Mini PCIe Cat 4 module, which is described in corresponding design section (Section 3.3.2.2). It has a Mini PCIe form factor, and therefore it is a plug and play solution for the LTE modem Quectel EC25-E. This avoids expensive and difficult embedding of the LTE modem on the base shield.

Another block which was integrated into the redesigned LTE base shield, is the HSM OPTIGA Trust X. The HSM in general, is functionally independent from the LTE module, but it was placed there to avoid an additional hardware block in the stack.

The original LTE base shield provides two connection types, UART and USB. For this project the USB connection is used for two reasons. The first reason is that the Raspberry Pi only provides two UART interfaces, which are already in use for the flight controller connection and the (internal) BT module connection. In the following chapter the UART interfaces of the Raspberry Pi are explained in detail (Section 4.1.1.2). The second reason for choosing USB as an interface to the LTE base shield, is the higher data rate compared to UART (UART: about 900 Kbit/s for down- and uplink) [75]. The data rate for USB 2.0 is up to 480 Mbit/s [67].

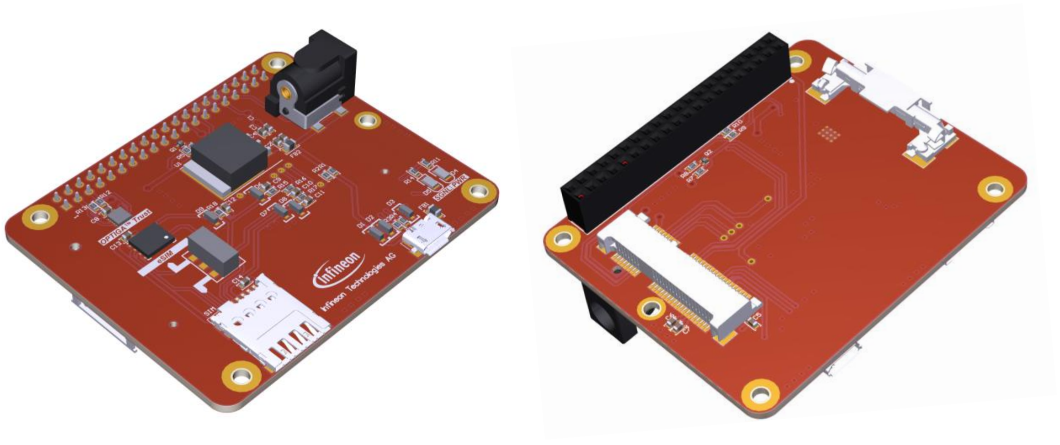


Figure 4.2: Redesigned LTE base shield

4.1.1.2 Raspberry Pi UARTs

The Raspberry Pi has two built-in UARTs, a PL011 and a mini UART [70]. The PL011 UART is ARM's PrimeCell UART (for details see [3]), which is similar to the UARTs used in most personal computers [11]. *"By default, on Raspberry Pis equipped with the wireless/Bluetooth module (Raspberry Pi 3 and Raspberry Pi Zero W), the PL011 UART is connected to the BT module, while the mini UART is used for Linux console output."* [70] Since within this project, the Bluetooth module is also used, this would mean that the mini UART could be used for connecting the flight controller board. The baud rate of the mini UART is linked to the core frequency of the Raspberry Pi, which can vary [70]. This is an issue for the communication with the flight controller board, because it requires a constant baud rate. To avoid any issues in this context, the assignment of the UARTs are changed. That means the PL011 UART is used to connect the flight controller board, and the mini UART is used to connect the Bluetooth module. This settings can be done by using the *raspi-config* utility (details can be found in [70]). In Linux based operating systems, interfaces such as USB or UART are represented as file streams in the systems */dev* folder. The file */dev/ttyS0* represents the mini UART and the file */dev/ttyAMA0* represents the PL011 UART. As depicted in Section 3.3.1.1, the UART interface consists (in the bidirectional case) of four wires, which are power supply, ground, transmit and receive. As depicted in Figure 4.3, the transmit and receive pins for the (before assigned) UART interface are represented by the General Purpose Input Output (GPIO) pins 14 and 15. In the hardware setup for the client (drone) of the proposed solution, the Raspberry Pi pins 4 (power), 6 (ground), 8 (UART transmit) and 10 (UART receive) are connected to the flight controller board.

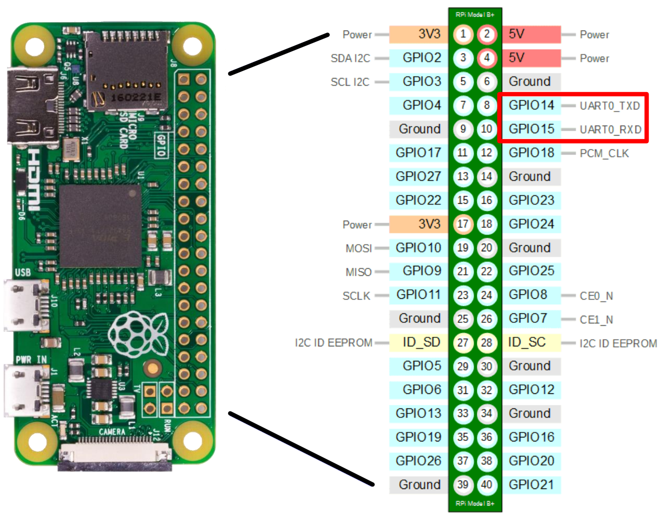


Figure 4.3: Raspberry Pi Zero W pinout (modified from [69])

4.1.2 Flight Control Server Hardware

The flight control server is running on a Raspberry Pi 3, with a touch screen monitor for interaction.

4.1.3 Demonstration Setup

The goal of this project is to propose a secured drone identification and tracking system, as well as showcasing it in a demonstration.

In general, the drone has to authenticate itself against a so called flight control server first, and then the drone sends its location data periodically to this flight control server. Since the requirements for the demonstration are, that it has to be functional indoor and it has to be possible to operate in a small area (such as a table), GPS can not be used for locating the drone, because GPS is not or only partially available indoor. As a workaround, BLE distance measurement is used to approximate the location of the drone on a small given area, or more specifically on a table. Details about the BLE distance measurement can be found in Section 2.3.2.1.1. The focus of this project is not to implement an indoor positioning system, but to enable the possibility to simulate no-fly zones as depicted in Section 3.1.2.

Figure 4.4 sketches the implemented demonstration setup, which consists of the drone itself, the flight control server and a table, where BLE beacons are mounted in a specific way, as explained in detail in Section 4.1.3.1. Due to the reason of safety and ease of use, the drone is tethered with a cord. That means the drone can only fly in a given range, which is important to avoid accidents during the demonstration. This is legitimate, since the focus of this project is not steering the drone or do special flight maneuvers.

To simulate a no-fly zone, a given segment is marked on the table as sketched in Figure 4.4. For visualization, the GUI hosted on the flight control server, represents the table two dimensionally (top view), which is depicted in the flight control server software section (Section 4.2.2.1) in Figure 4.11. If the drone enters the no-fly zone, an alarm on the flight control server is triggered and visualized in the GUI. If the drone is leaving the no-fly zone, the alarm is disabled again.

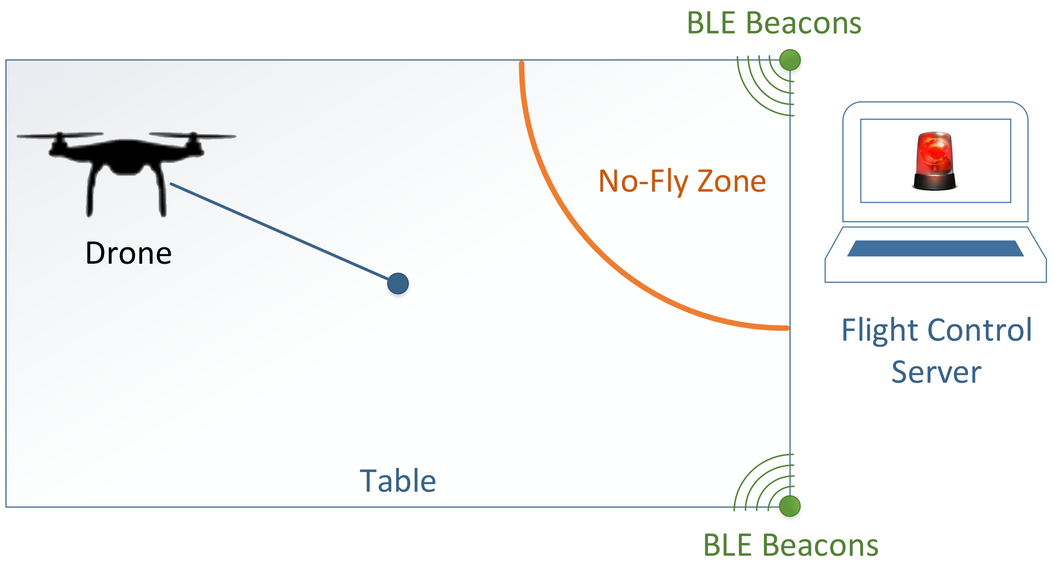


Figure 4.4: Demonstration setup

In practice, an example for this scenario could be, that the flight control server is sending an automatic stop sequence to the drone to perform a save landing as soon as a no-fly zone is entered. Further consequences, such as disciplining the pilot or confiscating the drone, are imaginable in practice.

In the demonstration setup, the steering of the drone is done by the flight control server, which would already allow taking over the control from the pilot (e.g. performing a save landing) after entering a no-fly zone. The GUI of the flight control server provides basic steering commands, such as starting, landing or flying in a given direction, which are explained in detail in Section 4.2.2.1.

4.1.3.1 BLE Beacons

For the Bluetooth beacons, the single-board development kit Nordic nRF52-DK is used, which is a highly flexible, multi-protocol system on chip [58]. The used development kit is depicted in Figure 4.5. It supports the BLE protocol stack and a proprietary 2.4 GHz protocol stack [58]. Nordic's nRF5 Software Development Kit (SDK) provides drivers and libraries as well as examples and a development environment together with documentation [57].

As analyzed in [44], the accuracy for positioning with BLE beacons, increases with the number of beacons used. Since the drone is tethered, the height of the drone (while flying) varies only a few centimeter. Therefore a two dimensional positioning is sufficient. In order to calculate the position, the principle of bilateration is used, which is explained in Section 4.2.2.1.7. Therefore at least two distinct distances to a given point are necessary. Figure 4.6 depicts the BLE beacon setup for the drone position estimation. To improve the accuracy and minimize the small height variation influence, two BLE beacons are stacked with a certain distance in order to get one distance to the drone, which results from averaging the two values. As depicted in Figure 4.6, two pillars, each consisting of two stacked BLE beacons, are mounted on two corners of the table. In order to avoid the location calculation on the drone, the distances are replaced by the RSSI values (measured

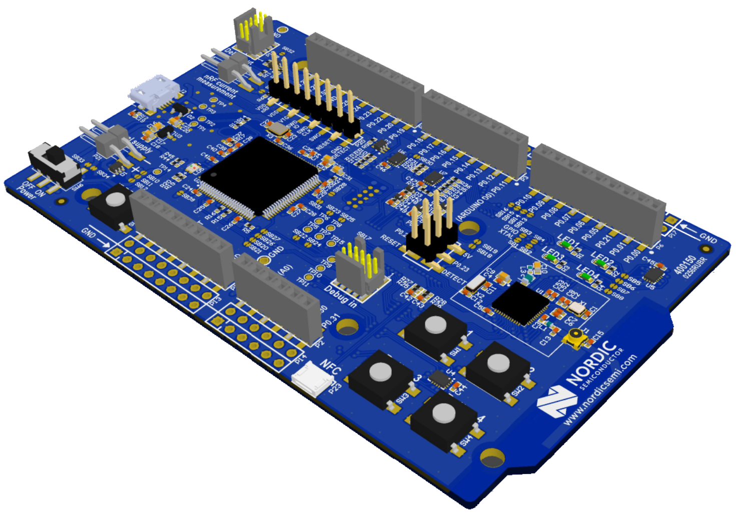


Figure 4.5: Nordic nRF52-DK (used as BLE beacon) [58]

by the drone) and forwarded to the flight control server. Another reason to send the RSSI values instead of the distances, via the TLS secured communication channel, is to provide the opportunity for improvements of positioning method later on (without changes on the drone). The flight control server is calculating the distance with a simplified Friis equation [24]. The exact position calculation procedure is described in the corresponding software implementation section (Section 4.2.2.1.6).

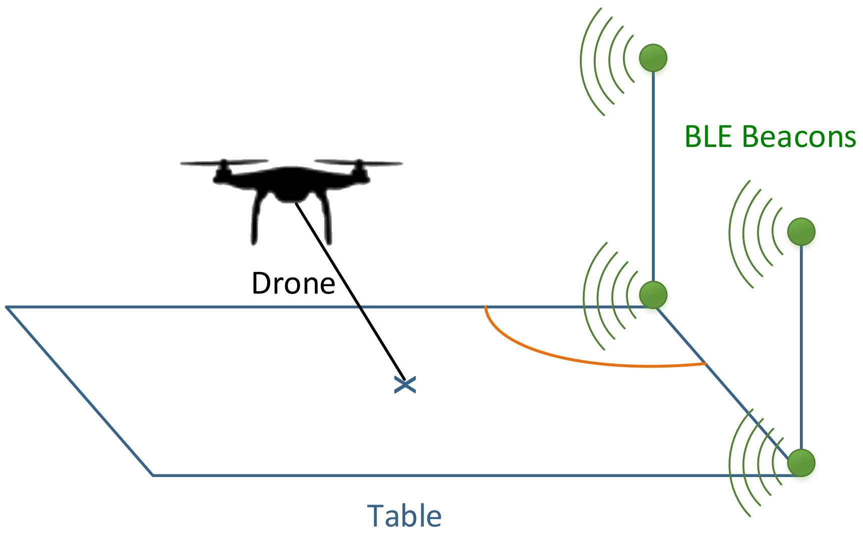


Figure 4.6: BLE beacon setup

4.2 Software Implementation

This section describes the software implementation of the proposed drone identification and tracking solution, which consists of three parts, the drone software, the flight control server software and the software on the BLE beacon. As already depicted in the software design section, the client and the server part are both using a TCP/IP socket as a transport channel. Due to the fact that these sockets are a platform independent network interface, it does not matter which OS is used on one side or the other. Further, this also allows the free choice of the programming language on both sides.

4.2.1 Drone Software

In this section the implementation of the software hosted on the drone is explained in detail. In this solution, the drone software is hosted on a Raspberry Pi Zero W. As an OS, the official Raspbian GNU/Linux 9 (stretch) is used. The main software parts are the implementation of the application itself, the TLS implementation and the BLE tools, which are used in order to estimate the location of the drone in a given area.

4.2.1.1 Application

As depicted in the software design block diagram (Figure 3.11), the application is divided into three main parts, the AT command relay, the steering relay (remote control relay) and the drone identification and tracking. Since the drone identification and tracking data should be sent in a secured way, the application needs a transport layer security protocol (TLS in this case). The TLS layer of the software on the drone, is partitioned between the host (Raspberry Pi Zero W) and an HSM (OPTIGA Trust X). The command library for the OPTIGA Trust X is written in C. In order to avoid expensive re-writing of the command library or implementing a wrapper for the command library, the application hosted by the drone was also implemented in C. The used C library for this project is *glibc 2.24*.

In Figure 4.7, the implementation of the client application is depicted. The AT relay and the RC relay are running in separate threads (Thread-1 and Thread-4 in Figure 4.7). The drone identification part is split into two threads, one for reading and parsing the input coming from the BT monitor (*btmon*), and one for forwarding new BT beacon data (RSSI values) to the flight control server (Thread-2 and Thread-3 in Figure 4.7). Details about each part of the application are given the corresponding sections.

4.2.1.1.1 RC Relay

In the proposed solution, the flight control server is sending the steering commands to the drone's Raspberry Pi (as depicted in Section 4.1.3). The Raspberry Pi needs to relay the steering commands to the Larix EDU V2 flight controller board. As depicted in Figure 3.8, the flight controller board is connected via UART to the Raspberry Pi. The chosen UART interface on the Raspberry Pi is the PL011, which is represented within the OS through the file */dev/ttyAMA0*. Details about the UARTs on the Raspberry Pi are described in the corresponding hardware implementation section (Section 4.1.1.2). To relay the steering data via UART to the flight controller, the serial port for UART has to be initialized and configured beforehand. In order to clearly separate the application from the UART interaction, corresponding files were created (*uart.c*, *uart.h*). These files represent the declaration and implementation for the functions, which are necessary to relay data to the UART interface. The implemented functions are *uart_init()*, *uart_tx()* and *uart_close()*.

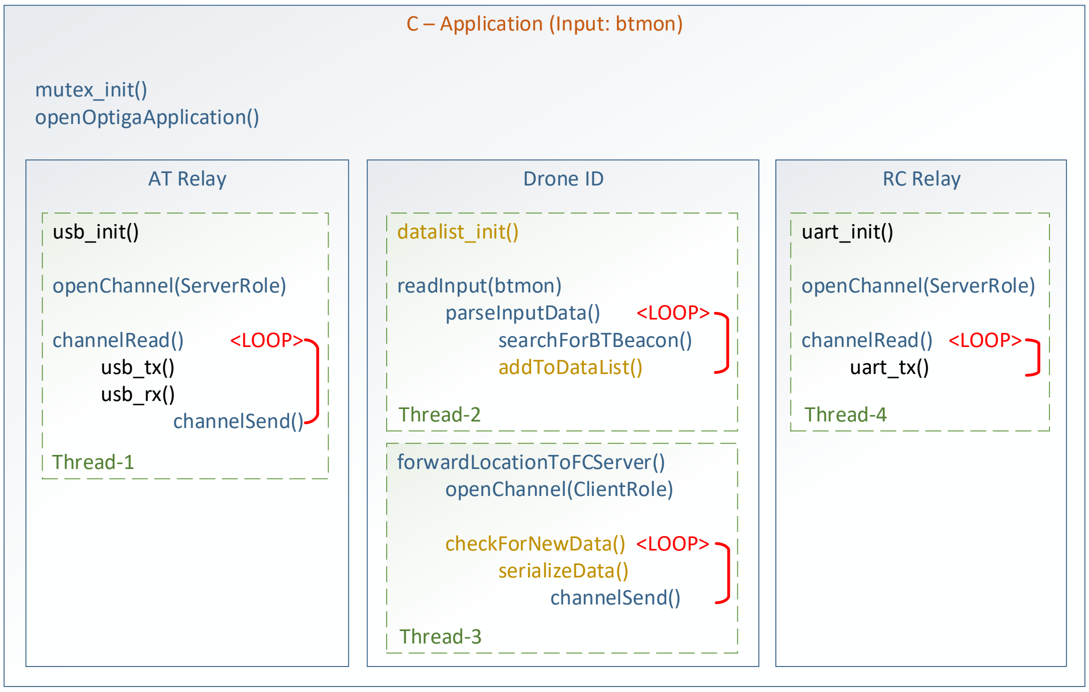


Figure 4.7: Client (drone) software - application implementation

When opening the system file */dev/ttyAMA0*, specific parameters (baud rate, parity bits, stop bits, etc.) needs to be set, in order to allow a correct communication via UART.

The steering commands are data packets with the length of 16 bytes, formatted according to Spektrum's DSMX protocol [39], which are generated at the flight controller server (within this project). The drone's Raspberry Pi is receiving the steering commands via an TLS secured TCP/IP socket (implemented in an endless loop), which has to be opened beforehand. Details about the TLS connection are described in Section 4.2.1.2.

4.2.1.1.2 AT Relay

The AT relay application block is responsible for relaying AT commands, which are generated at the flight control server (details in Section 4.2.2.1.3), to the LTE module (which hosts the eSIM), in order to get information about the eSIM or to trigger an MNO change. This works similar as the RC relaying block, except that the AT relay block has to provide the possibility to send back the AT command response to the flight control server. Details about AT commands are described in the corresponding section (Section 4.2.2.1.4). As depicted in the hardware block diagram (Figure 3.8), the LTE base shield (which is the bridge to the LTE module) is connected via USB to the Raspberry Pi. The USB interface is as well as the UART interface, represented as a file in Linux based operating systems (in this case */dev/ttyUSB3*). That means, to open the serial port for USB, the port has to be initialized respectively opened with specific parameters beforehand (with *usb_init()*, analogous to UART). The functionalities which are necessary for initializing, as well as reading and writing to USB port, are clearly separated in corresponding declaration and implementation files (*usb.c*, *usb.h*).

As the RC commands, the AT commands are received via a TLS secured TCP/IP socket. Therefore the socket has to be opened before the AT commands can be received.

Details about the TLS connection are described in Section 4.2.1.2. After establishing the secured channel, the thread is continuously waiting for incoming data. If an AT command is received, the command is relayed to the before opened system file (in this case */dev/ttyUSB3*), with the function *usb_tx()*. The response which is read (with *usb_rx()*) from the serial port (represented by the system file) is sent back to the flight control server, via the already established TLS connection. After sending the response back, the thread is again waiting for new incoming AT commands.

4.2.1.1.3 Drone ID

Since the main goal of this project is to propose a secured drone identification and tracking system, the most important parts of the application are gathering and sending location information data to the flight control server. As explained in Section 4.1.3, RSSI values from BLE beacons are used as location information (instead of e.g. GPS). Linux based operation systems (in this case Raspbian) are providing tools to (inter-) act with BT devices, which are explained in Section 4.2.1.3. The command *btmon* is monitoring all BT devices together with their RSSI values and other information about the device (see Section 4.2.1.3). Therefore, the Drone ID application block takes the output of *btmon* as an input. This software block is divided into two distinct threads.

The first thread (Thread-2 in Figure 4.7) is responsible for parsing the input data for a certain pattern (*parseInputData()*). The input data comes from the output of the *btmon* command. The parsing happens in an infinite loop. The pattern which has to be parsed by the application is explained in detail in the BLE tool section (Section 4.2.1.3). Shortly, it has to be parsed for the before specified BLE beacons (hardware addresses) and their corresponding RSSI value. This value is put into a list of certain data structs, which represent the RSSI values for each of the before specified BLE beacons. As explained in the BLE beacons' hardware implementation section (Section 4.1.3.1), there are 2 x 2 beacons specified within this project. In Listing 4.1, the BLE beacons are represented by their hardware addresses. In this implementation, the data struct for the RSSI values, is represented by a dimensional integer array, as depicted in the lower part of Listing 4.1.

Listing 4.1: BLE beacons and RSSI values struct

```
#define NR_PILLARS 2
#define NR_BEACONS_PER_PILLAR 2
char *BEACON_HW_ADDRESSES[NR_PILLARS][NR_BEACONS_PER_PILLAR] = {
        {"EC:2E:5D:7B:AE:48", "C7:1F:BF:43:DA:62"},
        {"E9:56:D5:DA:40:ED", "E8:D9:AE:DC:C1:10"}
};

struct Data
{
        int rssi[NR_PILLARS][NR_BEACONS_PER_PILLAR];
} data;
```

The second thread within this application block (Thread-3 in Figure 4.7), is waiting for new data within the data list which gets filled from the first thread of this application block. The waiting for new data is happening in an endless loop. If there is new data in the list, the latest data struct from the list is serialized according to the CBOR data serialization format, which is explained in detail in Section 2.2.1.2. The used CBOR library for this project is the *libcbor v0.5.0*, downloaded from [62]. In order to serialize the used data struct, the data has to be converted to a *cbor_item_t* according to the data type. As described in Section 4.1.3.1, RSSI values parsed from the *btmon* output can either be 0 or negative. As stated in [43], the minimum usable RSSI value is -88 dBm. Therefore an 8 bit integer is sufficient, as depicted in Table 4.1.

Type	Bytes	Output Range
negint8	1	-1 to -256
negint16	2	-1 to -65 536
negint32	4	-1 to -4 294 967 296
negint64	8	-1 to -18 446 744 073 709 551 616

Table 4.1: Ranges of unsigned integer (in CBOR)

The *libcbor* library provides positive and negative integers, but the value 0 is only held by the positive integer type. This has to be taken into account when converting the RSSI value into a *cbor_item_t*. That means if the value is 0, the function *cbor_build_uint8(...)* from the CBOR library has to be used and otherwise the function for creating a negative integer has to be used (*cbor_build_negint8(...)*). After converting the single values from integers to *cbor_item_t*, the items have to be pushed to the parent element within the CBOR structure. In Listing 4.2 a generic function for creating a CBOR root element out of the RSSI data struct is depicted (with an additional string containing other information e.g. ID).

Listing 4.2: Function to convert RSSI data struct to CBOR element

```
cbor_item_t *convert_data_to_cbor_item(struct Data data)
{
        cbor_item_t *root = cbor_new_definite_array(NR_OF_STRUCTMEMBERS);
        cbor_item_t *pillars = cbor_new_definite_array(NR_PILLARS);
        int i, j;
        for (i = 0; i < NR_PILLARS; i++)
        {
                cbor_item_t *pillar = cbor_new_definite_array(
                    NR_BEACONS_PER_PILLAR);
                for (j = 0; j < NR_BEACONS_PER_PILLAR; j++)
                {
                        cbor_array_push(pillar, data.rssi[i][j] == 0 ?
                            cbor_build_uint16(0) : cbor_build_negint16(abs(
                            data.rssi[i][j]) - 1));
                }
                cbor_array_push(pillars, pillar);
        }
        cbor_array_push(root, cbor_build_string(data.id));
        cbor_array_push(root, pillars);

        return root;
}
```

After creating a CBOR element, it has to be serialized with the function *cbor_serialize(...)* from *libcbor*. The serialized data is sent to the flight control server via the before established TLS connection (details are depicted in Section 4.2.1.2).

4.2.1.2 TLS

As explained in the corresponding design section (Section 3.4.1.2), the TLS layer is split between the host controller and the secure element. In Figure 3.12 the TLS host software architecture is depicted. In this section, the implementation of the TLS application interface together with the OPTIGA Trust X command library is explained. The command library provides the functionality to support certain steps of the TLS handshake. These support functions are implemented in the secure element and invoked by the host. The TLS application interface provides the TLS functionality to the application. The implemented IFX TLS application interface provides the basic functionality optimized for IoT applications. The IFX library consists of several C source files and the corresponding header files, depicted in Figure 4.8.

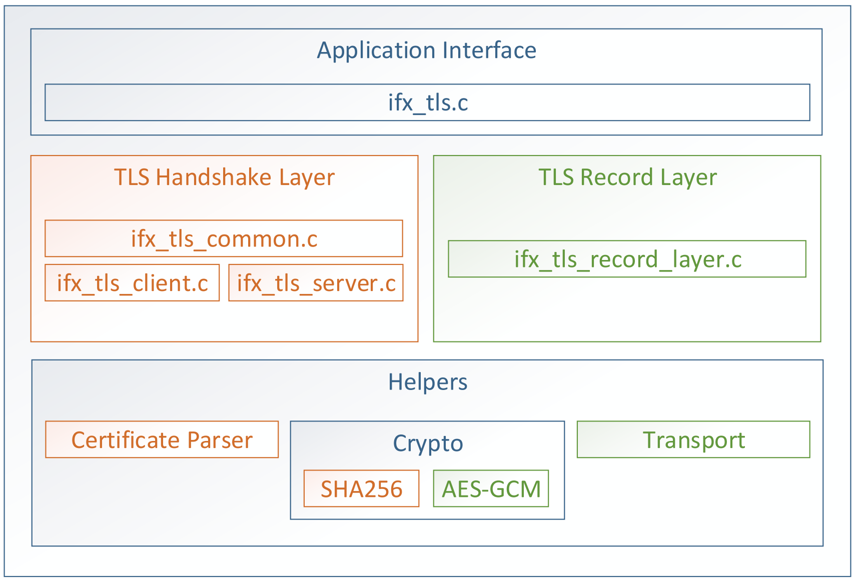


Figure 4.8: Infineon TLS library file architecture

The application interface is implemented in the file *ifx_tls.c*, where the main functions are *tls_init(...), tls_perform_handshake(...), ifx_tls_write(...) and ifx_tls_read(...)*. As previously depicted in Figure 2.7, the TLS layer can be separated into the TLS handshake layer and the TLS record layer. This separation is also represented by the file structure of the IFX TLS library. The TLS record layer is implemented in the file *ifx_tls_record_layer.c*. As the tasks in the TLS record layer are sending and receiving TLS packets, the most important functions are called *ifx_tls_record_layer_receive(...) and ifx_tls_record_layer_transmit(...)*. The TLS handshake procedure consists of several steps, where it has to be distinguished if the host is acting in client or server role. Therefore, the functionality for the TLS handshake layer is represented by three different files (plus corresponding header files), which are *ifx_tls_common.c, ifx_tls_client.c and ifx_tls_server.c*. In Addition, several helper functions for example for certificate parsing or certain cryptographic functions are necessary. The X.509 certificate parser functions as well as the hash algorithm SHA256 are used in the TLS handshake layer. In the TLS record layer, the AES and GCM files as well as the transport file, which contains helpers for the network sockets, are necessary. The following paragraph gives detailed information about the implementation of the TLS handshake procedure.

4.2.1.2.1 TLS Handshake Layer

In the corresponding design section in Figure 3.6 an overview of the TLS handshake is depicted. The green as well as the blue blocks from this figure are implemented in the before mentioned files *ifx_tls_common.c, ifx_tls_client.c and ifx_tls_server.c*.

The client file (*ifx_tls_client.c*) contains the functionality for sending the messages depicted on the left side of the picture, as well as the parsing for the server messages. Where the server file (*ifx_tls_server.c*) contains the functionality for sending the server messages (right side of Figure3.6) and the parsing for the messages sent from the client. Certain message are sent and parsed by the client and the server, for example the write *Certificate* message. Therefore the write and parse functions are implemented in the common file (*ifx_tls_common.c*). The TLS handshake procedure is implemented as a state machine according to [34].

4.2.1.2.2 ClientHello

The *ClientHello* message is the first message with the purpose of sending the security enhancement capabilities of the client to the server, which are:

- Highest TLS version
  - TLS 1.2
- Client random number
  - Must be random and unpredictable to prevent reuse of messages
  - Generated with TRNG by secure element
- Session ID
  - Always zero in this case, because session resumption is not used
- Supported cipher suites
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- Compression methods
  - No compression in this case, because compared to the certificates the size reduction is negligible
  - Compression will be removed in TLS 1.3 due to security reasons
- Extensions
  - Supported signature and hash algorithms (for authentication)
    SHA256, ECDSA
  - Supported elliptic curves (for ECDHE key exchange)
    NIST P256
  - Supported ECC point formats (for ECDHE key exchange)
    Uncompressed ECC point format

In Figure 4.9, the used cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 is explained.

The chosen cipher suite is one of the most used cipher suites in state-of-the-art IoT applications. The key exchange algorithm ECDHE is based on ECC, which needs according to NIST recommendations, a smaller key size in order to reach the same cryptographic strength [47]. Therefore it has a better performance, especially on processing power and storage limited IoT devices. The key size and the hashing algorithm varies in different

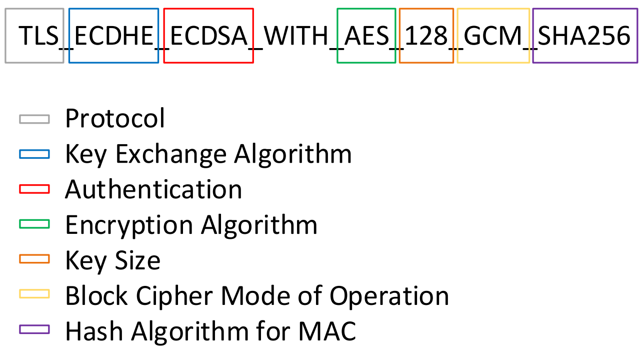


Figure 4.9: Cipher suite example

applications. For this project, only the hashing algorithm SHA256 is implemented, because the used HSM (OPTIGA Trust X) only supports this hashing algorithm (depicted in Table 2.9).

4.2.1.2.3 ServerHello

Before sending the *ServerHello* message, which is the response to the *ClientHello* message, the received message has to be parsed by the server. The security enhancement capabilities from the client are extracted by the server and if a set of acceptable security algorithms is found, the server sends the *ServerHello* message. Otherwise an alert message is sent and the TLS handshake procedure is aborted. The *ServerHello* message has a similar structure as the *ClientHello* message.

4.2.1.2.4 Server Certificate

The *Certificate* message contains the certificate chain. Within the IFX TLS library, this message composition is implemented in the *ifx_tls_common.c* file, because it is necessary for both parties of the TLS handshake (client and server). In this implementation, the certificate chain which sent within this message, is read from the secured certificate storage of the OPTIGA Trust X, which is explained in Section 2.2.3.3.

4.2.1.2.5 ServerKeyExchange

As specified in the selected cipher suite, ECDHE is used for key exchange in this implementation. The ECC key pair is generated by the server and transmits the public key in this message. The client will later use the servers public key to calculate the pre-master secret [34]. During the creation of this message, the server authentication is performed. This is done by hashing and signing the ECDHE public key. The hash and signature algorithms where previously specified in the *ClientHello* extensions, which are SHA256 and ECDSA in this implementation.

4.2.1.2.6 CertificateRequest

As described in the corresponding TLS design section, the *CertificateRequest* message is optional. The server sends this message, if he wants the client to authenticate itself against the server by sending *Certificate* message, which contains a certificate chain. As in the proposed drone identification system, the drone (client) has to perform an authentication against the flight control server, the transmission of the optional *CertificateRequest* message is implemented and used. With this message, the supported certificate types as well as the supported hash and signature algorithms are sent.

4.2.1.2.7 ServerHelloDone

At the end of the *ServerHello* and its associated messages, the *ServerHelloDone* message is sent. After sending this message, the server waits for a response from the client. This message is an empty handshake message. This step is required because the *CertificateRequest* message is optional.

4.2.1.2.8 Parse Received Server Messages

In this step, the client is parsing all messages received from the server up to now (from *ServerHello* to *ServerHelloDone*). All information received through the *ServerHello* message is checked for errors. If an error occurs, an alert message is sent and the TLS handshake procedure ends. Further, the server random number is saved, because it is needed in a later step within the TLS handshake procedure. It is necessary to copy the value, because all messages which are already parsed are discarded.

The parse certificate function is as the *Certificate* message, implemented in the file *ifx_tls_common.c*, because it used by both parties. It makes use of the helper functions from the file *ifx_tls_x509_parser.c*. The certificate chain is parsed, as explained in the corresponding design section (Section 3.2.2.2). The validity is checked certificate by certificate. In this implementation of the certificate parser, there is no memory allocated, except for the struct, which holds the address to the public key of the certificate and its length. For the first certificate in the chain, the public key is saved for a later step during the TLS handshake procedure. The last certificate in the chain has to be a certificate which is signed by a trusted root certificate, which is hold by the certificate storage [82]. For further steps during the TLS handshake, the public key of the root certificate is extracted.

In the next step, the *ServerKeyExchange* message is parsed and the Elliptic Curve Diffie-Hellman (ECDH) public key is extracted and stored. During parsing this message, the signature of the message is verified.

As in this implementation the *CertificateRequest* message is sent by the server, the client has to parse this message and check if it can serve the requested certificate types. Otherwise an alert message is sent and the TLS handshake procedure ends.

Since the *ServerHelloDone* message is an empty handshake message, the length of the message, which should be zero (excluding the header), is checked. Otherwise an alert message is sent.

4.2.1.2.9 Client Certificate

Triggered by the *CertificateRequest* message from the server, the client has to send a *Certificate* message to the server. As described in the analogous server message (Paragraph 4.2.1.2.4), the implementation is common for both roles. The sent *Certificate* is extracted from the certificate storage of the OPTIGA Trust X.

4.2.1.2.10 ClientKeyExchange

This message is used to set the pre-master secret [34]. The pre-master secret can either be a shared secret or a pre-shared secret. In this implementation the shared secret is used. The shared secret is generated with the HSM (OPTIGA Trust X). Since ECDHE is used as a key exchange algorithm, this is done by transmitting the Diffie-Hellman public value (generated by the HSM), which is a point on the elliptic curve. In this implementation, the function to write the *ClientKeyExchange* message, also contains the derivation of the bulk key which happens in additional two steps.

The master secret is generated with the KDF of the HSM with the pre-master secret and the random numbers from the client and the server as input. After that, the bulk key is generated with the KDF of the HSM, with the before generated master secret and again the random numbers as input. The bulk key is compared to the master secret, expanded in length, to be long enough to be split into a client write MAC key, a server write MAC key, a client write encryption key, a server write encryption key and an initialization vector for the AES-GCM algorithm [34].

4.2.1.2.11 CertificateVerify

"This message is used to provide explicit verification of a client certificate." [34] In this implementation the message contains a signature of the hash over all TLS handshake messages sent and received up to now.

4.2.1.2.12 Client ChangeCipherSpec

This message composition is implemented in the file *ifx_tls_common.c*, because it is used by the client and the server. With the *ChangeCipherSpec* message, which is part of the change cipher spec protocol (not of the handshake protocol), it is indicated, that from now on all messages are encrypted with the before negotiated cipher suite and keys [82]. As depicted in Figure 3.6, up to inclusively the *ChangeCipherSpec* message, the communication was plain and from now on all the message are encrypted (indicated with gray and blue bars on the picture border).

4.2.1.2.13 Client Finished

If the key exchange as well as the authentication process was successful, the encrypted *Finished* message is sent. The steps for creating the label, which is sent with the *Finished* message are:

1. Hash all sent and received messages
2. Append the label *"client finished"* to the hash
3. Use KDF to create label (with hash, label and master secret as input)

This function is implemented in the common C file, because the client and the server are sending the analogous *Finished* messages (with different label). This process ensures that all messages are linked together and that they can never be reused. In every TLS handshake new unpredictable random numbers (for client and server) as well as a random session key are generated.

4.2.1.2.14 Parse Received Client Messages

Parsing the client *Certificate* message, works analogous to the parser of the server *Certificate* message as described in the Section 4.2.1.2.8.

The function to parse the *ClientKeyExchange* message is implemented analogous to the implementation for writing the message. First, the shared secret (pre-master secret) is generated. Then the master secret and the bulk key are derived. After that step, the necessary keys for the encrypted communication are negotiated.

Further, the signature sent by the clients *CertificateVerify* message is validated. The validation happens with a function provided by the OPTIGA Trust X. If the validation fails, an alert message is sent and the TLS handshake procedure ends.

To verify the *Finished* message, the server calculates the finished value in the same way as the client does. This works by hashing all the sent and received messages, append the label *"client finished"* and use the KDF (with hash, label and master secret as input). The calculated value is compared to the received value (from the client *Finished* message). If the values do not match, an alert message is sent and the handshake ends.

4.2.1.2.15 Server ChangeCipherSpec

This message is analogous to the clients *ChangeCipherSpec* message. It indicates that from now on all messages are sent encrypted.

4.2.1.2.16 Server Finished

This message as well as the parsing on the client side work in the same way as the client *Finished* message. After successfully receiving and validating the server *Finished* message, the TLS handshake procedure is done and the application data can be sent encrypted.

The functions for writing and reading from the TLS secured communication channel are specified in the file *ifx_tls.c* and are called *ifx_tls_write(...)* and *ifx_tls_read(...)*.

4.2.1.2.17 TLS Record Layer

The before described application interface functions for reading and writing are calling functions from the TLS record layer, specified in the file *ifx_tls_record_layer.c*, which is using helper functions from the transport files.

As specified in the cipher suite, the encryption algorithm AES-GCM is used with a key size of 128 bit. The AES-GCM algorithm is explained in the corresponding design section (Section 3.2.2.1). The record layer functions for transmitting and receiving are using the helper files *aes.c* and *gcm.c*, which contain cryptographic functions. Each packet on the TLS record layer, consists of a header and the payload. The header contains of the protocol type, the TLS version (in this case TLS 1.2) and the length of the payload. Possible protocol types are:

- Handshake protocol
- Change cipher spec protocol
- Application protocol
- Alert protocol

4.2.1.3 BLE Tools

Since BLE RSSI values from several BLE beacons (detailed structure of BLE beacons are given in Section 4.1.3.1) are used to localize the drone within the demonstration setup, the C application hosted on the Raspberry needs to access these values. Linux based operation systems (in this case Raspbian) are providing tools to (inter-) act with BT devices. First, the operating system has to trigger a scanning procedure for BLE devices. This is done with *hcitool* command set. The detailed command set can be found in the Linux manual [51] or within the operating system with the command *man 1 hcitool*. The command, used to scan for BLE devices, is *hcitools lescan –duplicates* and displays the hardware addresses of the devices which are in the range of the BT module of the Raspberry Pi. The parameter *-duplicates* is necessary, because the BLE beacons are periodically sending advertising packets. This results in multiple packets from the same device, which otherwise would be filtered by the tool.

After triggering the operating system to perform a BLE scanning procedure, the received data packets have to be displayed. This is done with the Bluetooth monitor, which is started with the command *btmon*. The Bluetooth monitor is displaying all BT devices together with their signal strength values RSSI and other information about the device as depicted in Listing 4.3. These Linux commands are used for simplicity reasons for the demonstration setup. Otherwise the BT protocol stack has to be implemented.

Listing 4.3: Example for HCI Event from Bluetooth monitor

```
HCI Event: LE Meta Event (0x3e) plen 42 [hci0] 8.747653
      LE Advertising Report (0x02)
        Num reports: 1
        Event type: Non connectable undirected - ADV_NONCONN_IND (0x03)
        Address type: Random (0x01)
        Address: E8:D9:AE:DC:C1:10 (Static)
        Data length: 30
        Flags: 0x04
          BR/EDR Not Supported
        Company: Nordic Semiconductor ASA (89)
        Data: 0215011223344556677889 9aabbccddeeff001020304c3
        RSSI: -57 dBm (0xc7)
```

The most important information from Listing 4.3 extracted by the *parseInputData(...)* function from the C application, is the hardware address (*Address: E8:D9:AE:DC:C1:10 (Static)*, in this case) and the RSSI value which is -57 dBm in this example.

4.2.2 Flight Control Server Software

This section describes the software running on the flight control server, which is implemented on a Raspberry Pi 3. Details about the used hardware are depicted in the corresponding hardware implementation section (Section 4.1.2). The OS installed on the Raspberry Pi is the official Raspbian GNU/Linux 9 (stretch). Beside the OS, the main software parts are the application itself and the TLS implementation.

4.2.2.1 Application

Due to the fact that the TLS secured TCP/IP sockets are a platform independent network interface, it does not matter which programming language is chosen. That means it does not necessarily have to be C, which is used on the client (drone) side. The chosen programming language is Python (version 3.5.3). As depicted in the corresponding software design section (Section 3.4.2), the main parts of the application are the functionality for drone remote control, Profile switch and the drone identification itself.

Figure 4.10 depicts the structure of the flight control server application. The software blocks are the equivalents to the client (drone) software application, which are the blocks for Profile switch (AT relay), remote control (RC relay) and drone identification. Additionally, a GUI is implemented in order to provide the possibility for visualization and interaction with the user, which is also depicted in Figure 4.10.

In the file *main.py*, the functional separation between the threads is implemented. The setup for the GUI is implemented in *mainWindow.py*. The file *locating.py* holds helper

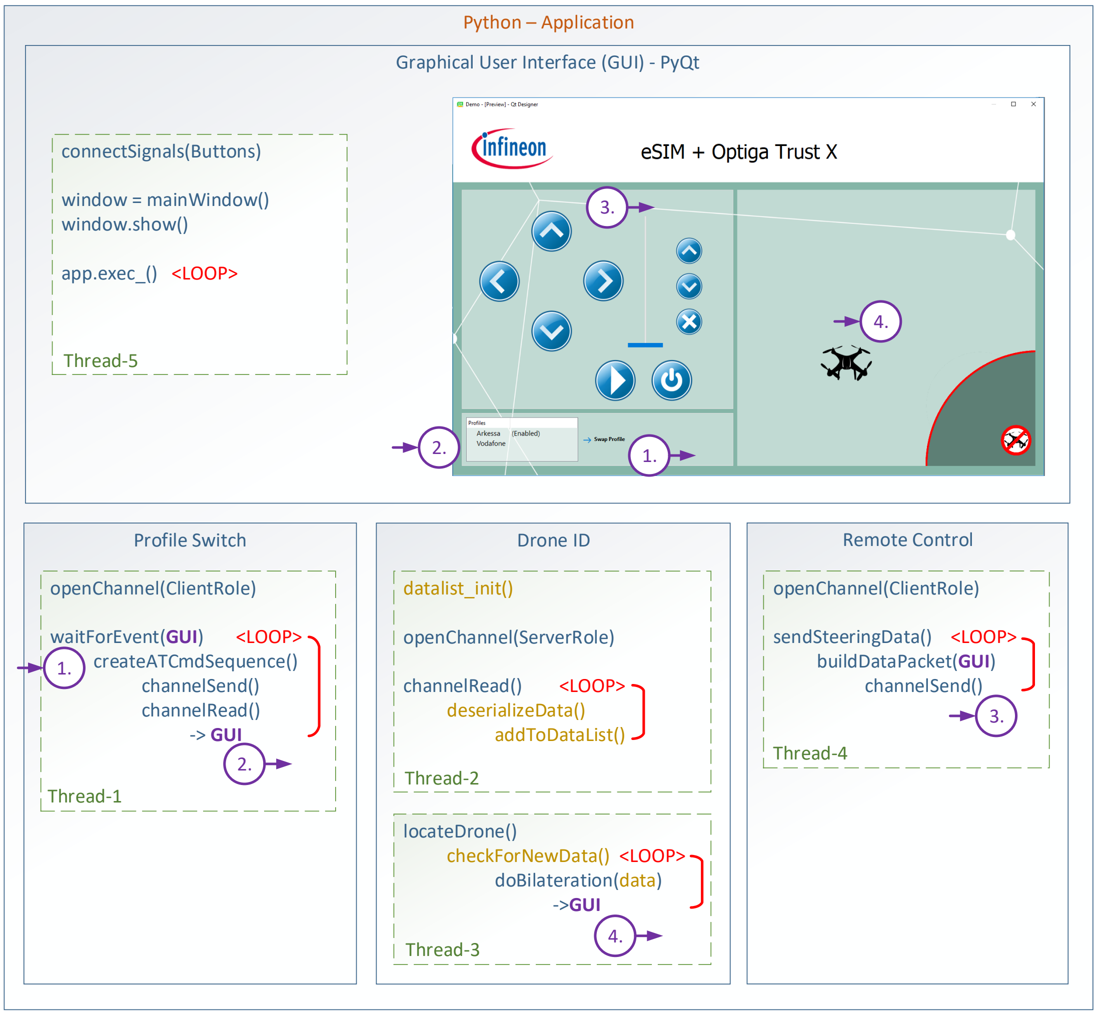


Figure 4.10: Flight control server software - application implementation

functions for the location estimation. In the file *steering.py* the 16-byte control data packet is created according to the input received from the GUI.

4.2.2.1.1 Graphical User Interface (GUI)

"Python has a huge number of GUI frameworks (or toolkits) available for it, from TkInter (traditionally bundled with Python, using Tk) to a number of other cross-platform solutions, as well as bindings to platform-specific (also known as "native") technologies". [63]

For this project, the common cross-platform framework *PyQt* is used, which works with Python 2 and 3. The GUI was created with the Qt Designer (version 5.9.1). Detailed information about how to create a GUI with the PyQt Designer can be found in [65]. To add the functionalities to the GUI elements, so called *signals* are used. This works by connecting the corresponding functions to the elements, which are called *Qt Widgets*. After connecting the GUI elements to the corresponding functions, the GUI is opened (*window.show()*) and event handling is started *app.exec_()*).

In Figure 4.11 the flight control server GUI is depicted. It is divided into three distinct areas, which are the remote control section (1), the eSIM section (2) and the drone position section (3). The remote control section provides the basic functionality to remote control the drone. The control cross allows to perform roll and pitch movements. The slider, together with the up and down button are changing the speed of the rotors. The *On/Off*-button is responsible for arming and disarming the rotors. The *Play*-button provides the possibility of automatic starting (up to a given height, tethered) and landing the drone. The second area allows the interaction with the eSIM (via the TLS secured channel), which is used for identifying the drone against a certain MNO to provide network services. After a successful connection setup to the drone, the available MNO Profiles are read out of the eSIM and displayed on the GUI. In addition, the enabled Profile is highlighted. If there is more than one Profile available at the eSIM, an MNO Profile change can be triggered by selecting the desired Profile and clicking on the *Swap Profile*-button. The area on the right side within the GUI is sketching a map, which is representing the table from the demo setup from Section 4.1.3. In the lower right corner, the no-fly zone is visualized. Details about the functionalities behind the GUI are described in the corresponding section below.

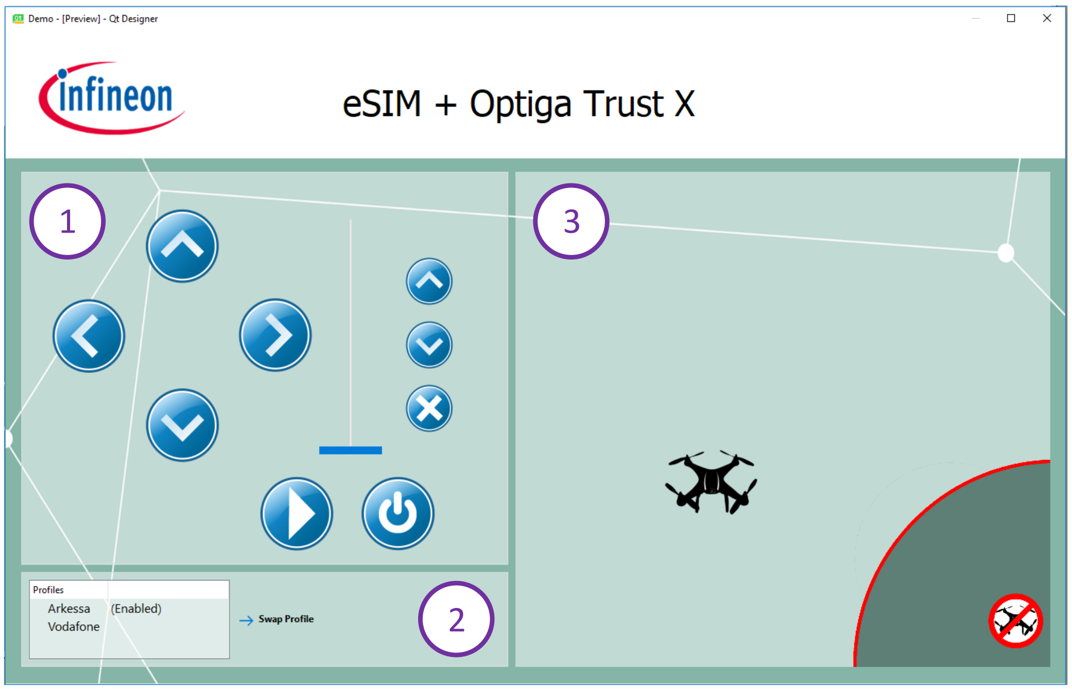


Figure 4.11: Flight control server GUI

4.2.2.1.2 Drone Remote Control

The functionality for the drone remote control is running in a distinct thread (Thread-4 in Figure 4.10). This thread is periodically sending (every 20 ms, but timing not critical) the 16 bytes long data packet according to Spektrum's DSMX protocol, to the drone's Raspberry Pi which is relaying the data packets to the flight controller. The data packet is sent to the flight control server via the before established TLS connection (details are depicted in Section 4.2.2.2). The 16-byte control data packet contains the control information for roll, pitch and yaw of the multicopter, as well as the speed and arming information for the rotors. Initially, the control data packet is filled with *Idle*-values, which keeps the drone in a neutral position with disarmed rotors. If a remote control event within the GUI happens, these idle values are modified. If for example the *Right*-button is pressed, the roll-value within the control data packet gets modified as long as the button is released.

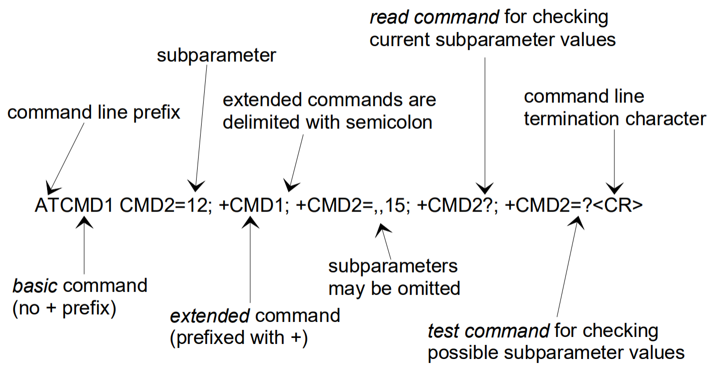
4.2.2.1.3 Profile Switch

This software block is responsible for triggering a Profile switch within the eSIM, which is used on the client (drone) for authentication against an MNO. In the proposed solution, the Profile switch is based on AT commands. These commands are generated on the server side and forwarded to the client, which is relaying them to the LTE module, as explained in Section 4.2.1.1.2. The AT command structure as well as the necessary AT command sequence for switching the MNO Profile on the eSIM are described in the following section.

4.2.2.1.4 AT Commands

AT commands are commands which are used to configure modems. AT is the abbreviation for *Attention*, and it is used as a compulsive prefix for every command. The standard is specified in the ITU-T recommendation V.250 [42]. The use of this standard, avoids the necessity of modem specific drivers. In general there are three groups of AT commands, the basic commands, the register commands and the extended commands. For some modems there is a fourth group of command, which is called proprietary or special command set. The basic command set provides basic interaction functionality such as dialing or answering a call. The register commands are used to set certain register values based on indices. The extended commands are separated into test, read, write and action commands [22].

In Figure 4.12 and Figure 4.13, the structure of basic and extended AT commands and its response are depicted. The main parts of the command line structure are the AT prefix, and the command itself, which is concatenated with a *+* for the extended case, and without it for the basic case. The commands are separated with a semicolon. For reading commands, a question mark is added and for test commands an equal sign followed by a question mark is used. Write commands are followed by an equal sign and parameters, which are separated with commas. Each command line ends with the termination character <*CR*>.



read command for checking current subparameter values
subparameter
command line prefix
extended commands are delimited with semicolon
command line termination character
ATCMD1 CMD2=12; +CMD1; +CMD2=,,15; +CMD2?; +CMD2=?<CR>
basic command (no + prefix)
extended command (prefixed with +)
subparameters may be omitted
test command for checking possible subparameter values

Figure 4.12: Basic structure of AT commands [22]

In Figure 4.13 the response structure is depicted. The final result code in case of a successful command execution should always be *OK*.

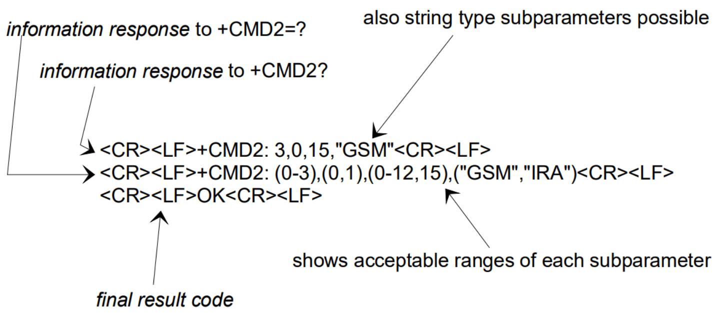

information response to +CMD2=?
also string type subparameters possible
information response to +CMD2?
<CR><LF>+CMD2: 3,0,15,"GSM"<CR><LF>
<CR><LF>+CMD2: (0-3),(0,1),(0-12,15),("GSM","IRA")<CR><LF>
<CR><LF>OK<CR><LF>
shows acceptable ranges of each subparameter
final result code

Figure 4.13: Basic structure of AT command response [22]

4.2.2.1.5 Command Sequence

In order to perform a local Profile switch on Avnet's eSIM, AT commands has to be sent to the eSIM to enable an already downloaded Profile. In this case a local Profile switch means, that the Profile switch is not executed by Avnet's Subscription Manager Secure Router (SM-SR). The following listing shows the procedure together with the AT commands which has to be followed according to Avnet's starter guide for the eUICC test pack purchased from [5] [6].

1. Open channel to modem (channel 1)
   AT+CSIM=10,"0070000001"
2. Select applet
   AT+CSIM=42,"01A4040010A0000000770307601100FE0000300001"
3. Send specific command to applet
   (a) Get eID
       AT+CSIM=10,"8116000012"
   (b) Audit (get current state)
       AT+CSIM=42,"01A4040010A0000000770307601100FE0000300001"
   (c) Enable Profile
       AT+CSIM=42,"8101000010A0000005591010FFFFFFFF890000****"
       **** indicates Issuer Security Domain Profile (ISDP) to select
4. Close channel to modem (channel 1)
   AT+CSIM=10,"0070800100"

4.2.2.1.6 Drone Identification and Tracking

The drone identification and tracking part of the application is separated into two threads (Thread-3 and Thread-4 in Figure 4.10). One thread is listening for new location information data on the before opened channel. If new location information data is received, the CBOR encoded data is de-serialized and saved to a list of BLE RSSI values, which are used for estimating the drone location. In the second thread of this drone identification and tracking software block, the estimated location of the drone is calculated based on the principle of bilateration, which is described in the subsequent section (Section 4.2.2.1.7). To smooth the noisy RSSI values, the average of the last five values is calculated. This results in values with less noise, but still a fair drone position update interval. The update interval is 0.5 s, because the advertising interval of the BLE beacons is 100 ms. In order to calculate the position of the drone, the BLE RSSI values have to be converted to distance values beforehand. The following equation (Equation 4.1) is used to convert the RSSI value to a distance value, based on a reference value (RSSI value on 1 m distance):

$$d = 10^{\frac{RSSI_{1m} - rssi}{10N}} \tag{4.1}$$

where d is the calculated distance; $RSSI_{1m}$ is the reference RSSI value in 1 m distance; $rssi$ is the RSSI value to convert; N represents the propagation exponent (2 for free space).

The identification of the drone against the flight control server is done with the certificate received during the TLS handshake procedure for the connection establishment, which is explained in the corresponding design section (Section 3.2.2.3).

4.2.2.1.7 BLE Bilateration

Bilateration is a geometric principle to calculate the position of a a point P, with the knowledge of two distances to the point P from two distinct stations. For example, GPS receivers are using the principle of trilateration, which works in the same way as bilateration but it uses three input distances. This results in one distinct point of intersection of the circles which can be drawn with the distances and the positions of the stations as center of the circle. That means, bilateration has a restriction compared to trilateration, but since the BLE beacons are located in the corners of the area of interest, there is only one possible point of intersection. Compared to the principle of lateration, the angulation principle (e.g. triangulation) takes the angles instead of distances as input parameter.

In order to avoid the computational expensive solving of three or more quadratic equations, the principle of Heron-bilateration is used [32]. Equations 4.2 and 4.3 depict the formulas for calculating the x and y position of the unknown point P, which is the drone position in this case (modified from [32]):

$$\begin{aligned} x &= x_{H2} - 2\frac{\sqrt{s(s-d_{H1})(s-d_{H2})(s-b_{12})}}{b_{12}} \\ \textit{with} \quad b_{12} &= |y_{H1} - y_{H2}| \quad \textit{and} \quad s = \frac{1}{2}(d_{H1} + d_{H2} + b_{12}) \end{aligned} \tag{4.2}$$

$$\begin{aligned} y &= y_{H2} + \sqrt{d_{H2}^2 - h_{12}^2} \\ \textit{with} \quad h_{12} &= x_{H2} - x \end{aligned} \tag{4.3}$$

Figure 4.14 depicts the parameters which are used in Equations 4.2 and 4.3 for calculating the Heron-bilateration [32]. The points *H1* and *H2* are the stations (beacon positions in this case) and the corresponding distances d_{H1} and d_{H2} are the distances from the stations to the unknown point P.

In addition to the implementation of the equations above, the input parameters (RSSI values) have to be checked for errors or infeasibility, such as the maximum possible values for the distances, which can be exceeded, because the BLE RSSI values are noisy.

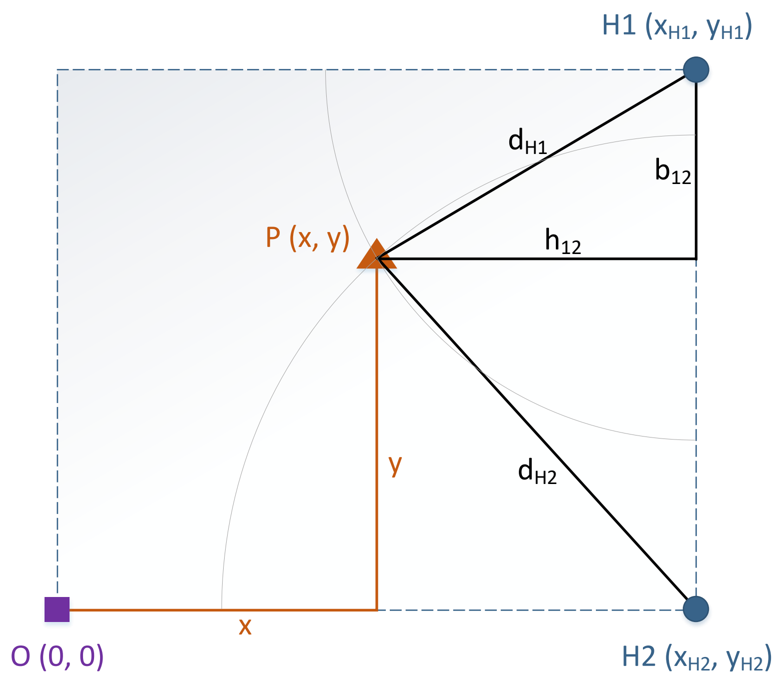


Figure 4.14: Principle of bilateration (modified from [32])

4.2.2.2 TLS

As depicted in the corresponding design section for the servers TLS (Section 3.4.2.2), the TLS block is fully implemented in software. The used software library is *OpenSSL*, an open-source software library for the TLS protocol. For the demonstration setup, the certificates and keys which are necessary for the secured connection establishment are stored locally on the flight control server. As the application on the flight control server is written in Python, a wrapper for the *OpenSSL* library is necessary. Python provides this wrapper with the *ssl* module. Detailed information about the *OpenSSL* library can be found on the official website [59].

4.2.3 BLE Beacon Software

For the BLE beacons software, an example project from the nRF5 SDK version 15.2.0 was used. The used project is a beacon transmitter sample application, which is setting the board into BLE advertising mode to broadcast BLE advertising packets in a given time interval with a given output power. The provided advertising intervals are between 100 ms and 10.24 s. In order to get a high timing resolution, the smallest interval (100 ms) was chosen. The values for the provided transmission power are between -40 dBm and +4 dBm. The highest value was chosen, because it results in the best RSSI - distance correlation. In order to work with the example project, the Integrated Development Environment (IDE) Segger Embedded Studio ARM V3.40 was used. To flash the software to the board, Segger J-Link (version v6.22) was used.

4.3 Evaluation

In this chapter, the Infineon TLS library is compared to the open-source library *OpenSSL*, based on the use case for IoT devices.

4.3.1 TLS Library Comparison

OpenSSL is a full-featured TLS library written in C [59]. The library supports several TLS and DTLS versions, a huge set of cipher suites and it also provides a cryptographic library [59]. In contrast to that, the Infineon TLS library, only supports TLS version 1.2 and one specific cipher suite. The *OpenSSL* library is suitable for a various set of applications and devices, where the Infineon TLS library is optimized for low processing power devices, especially in the IoT area. The main differences between those two libraries are depicted in Table 4.2.

	OpenSSL	IFX TLS
Language	C	C
Open-source	Yes	No
TLS versions	1.0, 1.1, 1.2, 1.3 (draft)	1.2
SSL versions (insecure)	2.0, 3.0	-
DTLS versions	1.0, 1.2	-
Lines of code	$\sim$ 546,200 [7]	$\sim$ 3500 + 6800*
Compiled code size (for Raspbian)	$\sim$ 460 kB	$\sim$ 64 kB

Table 4.2: TLS comparison

(*) The lines of code for the Infineon TLS library, is split into two parts. Approximately 3500 lines for the TLS library itself, plus approximately 6800 lines of code for the cryptographic helper functions. That means, compared to the *OpenSSL* library, the lines of code are roughly 50 times less. Especially for IoT devices, libraries with less compiled code size as well as fewer lines of code are more suitable. On one hand, the storage is mostly limited and on the other hand, some IoT devices need to get certified. That means all the software needs to be reviewed, which leads to a huge working overhead, if the lines of code are significantly more. Due the the storage limitation of IoT devices, another advantage of the Infineon TLS library can be taken into account. The X.509 certificate parser is implemented in a more efficient way, regarding memory consumption. In this implementation, the parser is not allocating any memory, except for the struct, which holds the address to the public key of the certificate and its length.

Since the *OpenSSL* library is backwards compatible to older TLS versions, downgrading attacks are possible and security issues which are already solved in newer versions can be exploited. For applications, where an interaction with a user is necessary, the version downgrading could be desirable, because it should be possible for users with older browser to interact with the server. This is not relevant for IoT applications, such as implemented in this project.

Chapter 5

Conclusion

The proposed drone identification and tracking system, which is introduced in this thesis, is based on standardized protocols and state-of-the-art technologies. Using standardized protocols for authentication (TLS) and data serialization (CBOR) instead of using proprietary protocols, increases acceptance changes in case of submitting the proposed system to an authority such as the FAA or the EASA. This would be the first step in order to convince authorities to adopt the proposed concept or even parts of the concept as a basis for new, upcoming regulations in the context of drone identification and authentication. TLS, which is an RFC standardized and approved by the IETF, provides the important properties authenticity, integrity and confidentiality. The authentication respectively identification of the drone (and further of the legal owner) is the most important part within this context. The TLS protocol is not only used for authentication, but also for securing the communication channel. During the connection establishment for the secured communication channel (where the authentication happens), keys and certificates are necessary. In order to securely store the necessary keys and certificates, the TLS layer of the communication is partitioned between the host (drone) and an HSM. The used HSM is the Infineon OPTIGA Trust X, which is Common Criteria (CC) certified, and provides a secured storage. Further, this design decision allows to relieve the host controller from processing power expensive, cryptographic operations.

For this system, the LTE network is selected as a physical link. In order to reach global, economical connectivity, an eSIM is used for the network authentication against an MNO. The economical aspect can be reached, due to the fact, that for eSIMs, over-the-air MNO changes are possible, which allows to select country specific MNOs, instead of using a global roaming tariff for the LTE connection. This is done with the introduction of Profiles and Remote SIM Provisioning (RSP), where each Profile is linked to an MNO. To change the MNO, a certain Profile has to be activated if it is already on the eSIM. Otherwise the Profile has to be downloaded first. Depending on the application, the consumer or the M2M approach can be chosen. Detailed information about Profiles are given in the eSIM section (Section 2.2.2.2). Contrarily, for traditional SIM cards, the SIM has to be physically replaced, in order to change the MNO. Due to this fact, traditional SIMs have to be placed on an accessible position in order to replace for an MNO change. eSIMs can be placed in every arbitrary position within the device because it does not have to be replaced. This aspect is especially in the IoT context very important, because those devices tend to be small in physical size.

Compared to other drone identification and tracking systems, the proposed system has several advantages. First of all, this system is a real identification and authentication system, not as DJI's AeroScope, which is a detection and tracking system. That means the AeroScope system (which is explained in Section 2.1.3.1) needs to detect a drone first, in order to track and identify it later on. Further, the detection range is limited to the range of the base station. That means every area which should be observed (no-fly zones) needs a base station. If more no-fly zones are within a given range, one base station can be used for surveillance, but since the range is limited to approximately 40 km [18], a huge amount of base stations would be necessary in order to reach a global coverage. This is not feasible, even tough the drone itself has to be modified in addition (all except DJI drones).

Within the newly proposed system, the drone is identifying as well as authenticating itself against a flight control server, which is driven by an official authority. Theoretically, it would be possible to have one single flight control server, which is used by every drone for authentication. In practice, this is not really feasible, because of several reasons. One reason would be the different regulations and laws of different countries and also the huge amount of drones, connecting to the server. As depicted in the certificate retrieval process (Figure 3.7), it would be a good practice to have several flight control servers per region. This is a similar approach as for DNS. That means for the proposed system, there is one global lookup service, which holds all certificates and domains from the regional authority CAs in a database. This global lookup service is used in order to get the corresponding domain of the regional authority, depending on the location. Further, each regional authority is driving one ore more flight control server. Possible parameters for the number of flight control servers could be the size of the region and the density of drone owners in this region. The chosen approach introduces a globally available system, with a standardized, state-of-the-art secured communication channel. The system is globally available, due to the fact that the chosen physical communication channel (LTE, later 5G) is widely spread, especially near civilization.

An advantage compared to Vodafone's RPS system is, that LTE is only used as a communication channel, not also for authentication. This means the physical channel could easily be replaced at any time, by any other communication channel, which is probably available in the future (e.g. 5G).

A disadvantage, which is a probably not possible to circumvent, if an active identification and authentication process happens, is that the drone system has to be modified. It has to be equipped with LTE, but this comes with other possible use cases, such as beyond the line-of-sight steering or transmitting high quality video streams. This is possible because LTE is capable of high transmission rates. Further, an additional software has to be implemented on the drone, but this could be standardized, in order to limit additional implementation work for the drone manufacturers. Another drawback which comes with additional system blocks (software and/or hardware) is, that it lowers the battery duration.

5.1 Outlook

For the future it would be advisable to propose the developed design and concept to aviation safety authorities, such as the FAA or the EASA. One good way to do so, would be to engage with an aviation authority via an RFI. If there is some legitimate interest shown from an authority, the proposed system should be improved in several ways. For example the application running on the drone, could be improved in the context of energy efficiency, which is a very important property for mobile applications. Further, the certificate retrieval process could be implemented as described in the design and concept. These steps were done manually in the current prototype. Additionally, the demonstration could be extended and changed in a way to operate it in realistic environments. That means to operate several drones (instead of only one) outside, and use GPS as a positioning system. On the flight control server side, the tracking of the drone could be implemented and stored in a database.

If, theoretically, a regulation comes up which uses a system similar to the proposed solution, there is one aspect which should be kept in mind. The commercial drone manufacturers could be forced to implement the system for every drone they are selling, but people can also built their own drone, without the identification system implemented. It theoretically could be forbidden by law to launch such drones, but it will never be possible to totally prevent it. Due to the fact that people need to have a certain amount of know-how to built their own drone, a regulation for commercial drone manufacturers would at least gather a big part of drone owners. Keeping that aspect in mind means, that also in future, radar systems cannot be totally replaced or omitted by such identification and authentication systems.

Acronyms

ADS-B	Automatic Dependent Surveillance - Broadcast
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BLE	Bluetooth Low Energy
BSON	Binary JSON
BT	Bluetooth
CA	Certificate Authority
CBC	Cipher Block Chaining
CBOR	Concise Binary Object Representation
CC	Common Criteria
CFB	Cipher Feedback
CI	Certificate Issuer
CNN	Convolutional Neural Network
COSE	CBOR Object Signing and Encryption
CSI	Camera Serial Interface
CTR	Counter
DES	Data Encryption Standard
DH	Diffie-Hellman Key Exchange
DIM	Drone Identity Module
DK	Deutsche Kreditwirtschaft
DMP	Digital Motion Processor
DNS	Domain Name System
DSA	Digital Signature Algorithm
DTLS	Datagram Transport Layer Security
EASA	European Aviation Safety Agency
ECASD	Embedded UICC Controlling Authority Security Domain
ECB	Electronic Codebook
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral Key Exchange
ECDSA	Elliptic Curve Digital Signature Algorithm
ESC	Electronic Speed Control
eSIM	Embedded SIM
ETSI	European Telecommunications Standards Institute
EU	European Union
eUICC	Embedded Universal Integrated Circuit Card

EUM	eUICC Manufacturer
FAA	Federal Aviation Administration
FIB	Focused-Ion Beam
FIPS	Federal Information Processing Standards
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GCM	Galois/Counter Mode
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input Output
GPS	Global Positioning System
GSMA	GSM Association
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HCI	Host Controller Interface
HSM	Hardware Security Module
IDE	Integrated Development Environment
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internet Protocol
ISDP	Issuer Security Domain Profile
ISD-R	Issuer Security Domain - Profile
ISD-P	Issuer Security Domain - Root
ISM	Industrial, Scientific and Medical
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
IV	Initialization Vector
I2C	Inter-Integrated Circuit
JSON	JavasScript Object Notation
KDF	Key Derivation Function
LBA	Luftfahrt-Bundesamt
LDS	Local Discovery Service
LiPo	Lithium-Polymer
LPA	Local Profile Assistant
LPD	Local Profile Download
LTE	Long Term Evolution
LUI	Local User Interface
MAC	Message Authentication Code
MNO	Mobile Network Operator
M2M	Machine to Machine
NFC	Near Field Communication
NAA	Network Access Application
NIST	National Institute of Standards and Technology
OFB	Output Feedback
OS	Operation System
OTA	Over-The-Air
OTDOA	Observed Time Difference Of Arrival

PAL	Platform Abstraction Layer
PCB	Printed Circuit Board
PER	Packed Encoding Rules
PKI	Public Key Infrastructure
PTZ	Pan-Tilt-Zoom
PWM	Pulse Width Modulation
RC	Radio Control
RFC	Request for Comments
RFI	Request for Information
RPS	Radio Positioning System
RSSI	Received Signal Strength Indicator
RSA	Rivest–Shamir–Adleman
RSP	Remote SIM Provisioning
SDK	Software Development Kit
SIM	Subscriber Identity Module
SMP	Security Manager Protocol
SM-DS	Subscription Manager Discovery Server
SM-DP+	Subscription Manager Data Preparation - enhanced
SM-SR	Subscription Manager Secure Router
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
TOA	Time Of Arrival
TLS	Transport Layer Security
TRNG	True Random Number Generator
UART	Universal Asynchronous Receiver Transmitter
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UICC	Universal Integrated Circuit Card
USB	Universal Serial Bus
XDR	External Data Representation
3GPP	3rd Generation Partnership Project

Bibliography

[1] Alice Evans. Heathrow airport: Drone sighting halts departures. `https://www.bbc.com/news/uk-46803713`. [Online; accessed 2019-01-10].

[2] L. Alqaydi, C. Y. Yeun, and E. Damiani. Security enhancements to TLS for improved national control. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 274–279, Dec 2017.

[3] ARM. *PrimeCell UART (PL011) Technical Reference Manual.* r1p4 edition, 2005.

[4] Avnet. Infineon. `https://www.avnet.com/shop/us/m/infineon/`. [Online; accessed 2019-01-07].

[5] Avnet. Welcome to the Avnet eUICC test pack. `https://www.avnet.com/wps/portal/silica/products/product-highlights/product-registration/avnet-euicc-test-pack/`. [Online; accessed 2019-01-02].

[6] Avnet. *Welcome to the Avnet eUICC test pack.* 0.85 edition, 2018.

[7] BlackDuck. OpenSSL. `https://www.openhub.net/p/openssl`. [Online; accessed 2019-01-08].

[8] Bluetooth SIG. *Bluetooth Specification Version 5.0.* 2016.

[9] Bluetooth SIG, Inc. Bluetooth technology. `https://www.bluetooth.com/bluetooth-technology/radio-versions`, 2018. [Online; accessed 2018-07-12].

[10] BMVI. Klare Regeln für Betrieb von Drohnen. `https://www.bmvi.de/SharedDocs/DE/Artikel/LF/151108-drohnen.html`, 2018. [Online; accessed 2018-09-20].

[11] Brendan Horan. *Practical Raspberry Pi.* Apress, 1 edition, 2013.

[12] BSON. BSON. `http://bsonspec.org/spec.html`. [Online; accessed 2018-07-10].

[13] Bormann C. and Hoffman P. Concise Binary Object Representation (CBOR). RFC 7049, RFC Editor, October 2013.

[14] CBOR. CBOR — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/wiki/CBOR`, 2018. [Online; accessed 2018-07-10].

[15] T. Chen, W. Huo, and Z. Liu. Design and Efficient FPGA Implementation of Ghash Core for AES-GCM. In *2010 International Conference on Computational Intelligence and Software Engineering*, pages 1–4, Dec 2010.

[16] COPTRZ. Real-time DJI Drone Detection System. `https://www.coptrz.com/dji-aeroscope/`. [Online; accessed 2018-09-28].

[17] T. Fischer C. Lesjak R. Matischek D. Houdeau, W. Maurer. Implementation of the architecture concept on test bench-level. *Power Semiconductor and Electronics Manufacturing 4.0*, 1.1, 2016.

[18] David Atkinson. Drone Detection and Tracking with Aeroscope. `https://www.heliguy.com/blog/2018/03/06/discussing-dji-aeroscope/`. [Online; accessed 2018-09-28].

[19] M. Dworkin. NIST Special Publication 800-38A, 2001 Edition: Recommendation for Block Cipher Modes of Operation, Methods and Techniques. December 2001.

[20] Dwight F. Hare Ellen H. Siegel. Hardware security module (HSM) chip card, 12 2004.

[21] ETSI. *Smart Cards; Machine to Machine UICC; Physical and logical characteristics (Release 9).* 2010.

[22] ETSI. *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; AT command set for User Equipment (UE).* 10.3.0 edition, 2011.

[23] FAA. FAA UAS Remote Identification Request for Information (RFI) - Data Exchange Strategies and Demonstrastions for UAS Remote Identification. Technical report, FAA, 2019.

[24] H. T. Friis. A Note on a Simple Transmission Formula. *Proceedings of the IRE*, 34(5):254–256, May 1946.

[25] GSM Association. *GSMA eUICC PKI Certificate Policy Version 1.1.* 2017.

[26] GSM Association. *RSP Architecture Version 2.2.* 2017.

[27] GSM Association. *RSP Technical Specification Version 2.2.* 2017.

[28] GSM Association. Embedded SIM. `https://www.gsma.com/aboutus/leadership/committees-and-groups/working-groups/sim-working-group/embedded-sim`, 2018. [Online; accessed 2018-07-30].

[29] S. Gueron and V. Krasnov. Speeding up Counter Mode in Software and Hardware. In *2014 11th International Conference on Information Technology: New Generations*, pages 338–340, April 2014.

[30] Hobbywing. XRotor Micro40A 4in1 BLHeli-S DShot600. `http://www.hobbywing.com/goods.php?id=588`. [Online; accessed 2018-11-14].

[31] Horizon Hobby, LLC. *Specification for Spektrum Remote Receiver Interfacing.* Rev a edition, 2016.

[32] C. Huang, L. Lee, C. C. Ho, L. Wu, and Z. Lai. Real-Time RFID Indoor Positioning System Based on Kalman-Filter Drift Removal and Heron-Bilateration Location Estimation. *IEEE Transactions on Instrumentation and Measurement*, 64(3):728–739, March 2015.

[33] C. Huitema and A. Doghri. Defining Faster Transfer Syntaxes for the OSI Presentation Protocol. *SIGCOMM Comput. Commun. Rev.*, 19(5):44–55, October 1989.

[34] IETF. *RFC5246 - The Transport Layer Security (TLS) Protocol Version 2.1.* tools.ietf.org, 2008.

[35] Infineon Technologies AG. *XMC4500 Microcontroller Series for Industrial Applications.* V1.5 edition, 2017.

[36] Infineon Technologies AG. *DPS422 Digital barometric pressure and temp sensor for portable and IOT devices.* V1.3 edition, 2018.

[37] Infineon Technologies AG. *OPTIGA Trust X Datasheet.* 2018.

[38] Infineon Technologies AG. *OPTIGA Trust X1 Solution Reference Manual.* 2018.

[39] InvenSense Inc. *MPU-9250 Product Specification.* 1.1 edition, 2016.

[40] ISO. ISO/IEC JTC 1/SC 17. `https://www.iso.org/committee/45144.html`. [Online; accessed 2018-10-15].

[41] ITU-T. ITU-T Recommendation X.690 : Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). Technical report, 2002.

[42] ITU-T. ITU-T Recommendation V.250: DATA COMMUNICATION OVER THE TELEPHONE NETWORK, Control procedures, Serial asynchronous automatic dialling and control. Technical report, 2003.

[43] B. Changseok J. Joonyoung, K. Dongoh. Distance Estimation of Smart Device using Bluetooth. In *The Eighth International Conference on Systems and Networks Communications*, pages 13–18, Oct 2013.

[44] M. Ji, J. Kim, J. Jeon, and Y. Cho. Analysis of positioning accuracy corresponding to the number of BLE beacons in indoor positioning system. In *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pages 92–95, July 2015.

[45] M. Jian, Z. Lu, and V. C. Chen. Drone detection and tracking based on phase-interferometric Doppler radar. In *2018 IEEE Radar Conference (RadarConf18)*, pages 1146–1149, April 2018.

[46] Johannes Ebert. Wireless Communication Networks and Protocols. Lecture Slides, TU Graz. [delivered 2018-03].

[47] Julie Olenski. ECC 101: What is ECC and why would I want to use it? `https://www.globalsign.com/en/blog/elliptic-curve-cryptography/`. [Online; accessed 2019-01-04].

[48] M. Koschuch, T. Fruhwirth, A. Glaser, S. Schmidt, and M. Hudler. Speaking in tongues practical evaluation of TLS cipher suites compatibility. In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, volume 01, pages 13–23, July 2015.

[49] K. Lee, S. Y. Lee, C. Seo, and K. Yim. TRNG (True Random Number Generator) Method Using Visible Spectrum for Secure Communication on 5G Network. *IEEE Access*, 6:12838–12847, 2018.

[50] Y. Li, X. Meng, S. Wang, and J. Wang. Weighted key enumeration for EM-based side-channel attacks. In *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, pages 749–752, May 2018.

[51] M. Krasnyansky, M. Holtmann. hcitool(1) - Linux man page. `https://linux.die.net/man/1/hcitool`. [Online; accessed 2018-11-20].

[52] Magdalena Nohrborg. LTE. `http://www.3gpp.org/technologies/keywords-acronyms/98-lte`. [Online; accessed 2018-10-19].

[53] A. Maratea, S. Gaglione, A. Angrisano, G. Salvi, and A. Nunziata. Non parametric and robust statistics for indoor distance estimation through BLE. In *2018 IEEE International Conference on Environmental Engineering (EE)*, pages 1–6, March 2018.

[54] Marcus Janke, Dr. Peter Laakmann. In *Attacks on Embedded Devices*, Embedded World Conference Nurenberg, 2016.

[55] Maxim Krasnyansky, Marcel Holtmann. hcitool (1) - Linux Man Pages. `https://www.systutorials.com/docs/linux/man/1-hcitool/`, 2018. [Online; accessed 2018-07-13].

[56] Microchip Technology, Inc. Generic Attribute Profile (GATT) Overview. `http://microchipdeveloper.com/wireless:ble-gatt-overview`, 2018. [Online; accessed 2018-07-13].

[57] Nordic Semiconductor. nRF5 SKD. `http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk%2Fdita%2Fsdk%2Fnrf5_sdk.html`. [Online; accessed 2018-11-22].

[58] Nordic Semiconductor. nRF51822. `https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822`. [Online; accessed 2018-11-22].

[59] OpenSSL Software Foundation. OpenSSL. `https://www.openssl.org/`. [Online; accessed 2019-01-02].

[60] M. Slanina P. Sedlacek and D. Kovac. An Overview of Indoor and Outdoor Positioning Technologies with Focus on their Precision . *Electro Revue*, 18(6), December 2016.

[61] J. Park, D. H. Kim, Y. S. Shin, and S. Lee. A comparison of convolutional object detectors for real-time drone tracking using a PTZ camera. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 696–699, Oct 2017.

[62] Pavel Kalvoda. libcbor. `http://libcbor.org/`. [Online; accessed 2018-11-20].

[63] Python Software Foundation. GUI Programming in Python. `https://wiki.python.org/moin/GuiProgramming`. [Online; accessed 2018-11-22].

[64] L. Qi et al. A Secure End-to-End Cloud Computing Solution for Emergency Management with UAVs. December 2018.

[65] Qt Company Ltd. Qt Designer Manual. `http://doc.qt.io/qt-5/qtdesigner-manual.html`. [Online; accessed 2018-11-22].

[66] S. Fries R. Falk. Advanced Device Authentication: Bringing Multi-Factor Authentication and Continuous Authentication to the Internet of Things. In *CYBER 2016, The First International Conference on Cyber-Technologies and Cyber-Systems*, pages 69–74, October 2016.

[67] Ranjan Parekh. *Principles of Multimedia.* Tata McGraw-Hill, 1 edition, 2006.

[68] Raspberry Pi Foundation. Camera Module. `https://www.raspberrypi.org/documentation/hardware/camera/`. [Online; accessed 2018-10-31].

[69] Raspberry Pi Foundation. Raspberry Pi Zero W (Wireless). `https://micro-pi.ru/raspberry-pi-zero-w-rpi0w-bcm2835/`. [Online; accessed 2018-11-16].

[70] Raspberry Pi Foundation. The Raspberry Pi Uarts. `https://www.raspberrypi.org/documentation/configuration/uart.md`. [Online; accessed 2018-11-16].

[71] SAMSUNG. eSIM Architecture. `https://developer.samsung.com/tech-insights/eSIM/esim-architecture",note="[Online; accessed2018-07-31]`, 2018.

[72] Cellars A. Schaad j. CBOR Object Signing and Encryption (COSE). RFC 8152, RFC Editor, July 2017.

[73] Sixfab. LTE-G-086 Cellular Miniature PCB Antenna. `https://sixfab.com/product/lte-g-086-cellular-miniature-pcb-antenna/`. [Online; accessed 2018-11-06].

[74] Sixfab. Quectel EC25 Mini PCle 4G/LTE Module. `https://sixfab.com/product/quectel-ec25-mini-pcle-4glte-module/`. [Online; accessed 2018-11-05].

[75] Sixfab. Raspberry Pi 3G-4G/LTE Base Shield V2. `https://sixfab.com/product/raspberry-pi-3g-4glte-base-shield-v2/`. [Online; accessed 2018-11-05].

[76] Sixfab. Raspberry Pi Iot Shields Sources. `https://github.com/sixfab/rpiShields`. [Online; accessed 2018-11-14].

[77] Steve McCaskill. Vodafone's 4G RPS to boost drone tracking. `https://www.techradar.com/news/vodafones-4g-rps-to-boost-drone-tracking`. [Online; accessed 2018-10-01].

[78] Audie Sumaray and S. Kami Makki. A Comparison of Data Serialization Formats for Optimal Efficiency on a Mobile Platform. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '12, pages 48:1–48:6, New York, NY, USA, 2012. ACM.

[79] Inc. Sun Microsystems. XDR: External Representation Standard. RFC 1014, RFC Editor, June 1987.

[80] B. Sung, K. Kim, and K. Shin. An AES-GCM authenticated encryption crypto-core for IoT security. In *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–3, Jan 2018.

[81] Sven Fischer. *Observed Time Difference Of Arrival (OTDOA) Positioning in 3GPP LTE.* Qualcomm, 1.0 edition, 2014.

[82] Thomas Fischer. *Design and Implementation of a Secure Personal Assistant Device with BLE and NFC*. TU Graz, 2016.

[83] Vodafone Group. Vodafone to protect the Skies with Trials of the Worlds first IoT Drone Tracking and Safety Technology. `https://www.vodafone.com/content/index/media/vodafone-group-releases/2018/iot-drone-tracking.html`. [Online; accessed 2018-10-01].

[84] Bing Zhou, Xianxiang Chen, Xinyu Hu, Ren Ren, Xiao Tan, Zhen Fang, and Shanhong Xia. A Bluetooth low energy approach for monitoring electrocardiography and respiration. In *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013)*, pages 130–134, Oct 2013.