Tallinn University of Technology

Faculty of Information Technology

Department of Computer Science

TUT Center for Digital Forensics
and Cyber Security

R^3AD

An architecture to include UAVs in national Airspace

Master Thesis

ITC70LT

Author:	Florian Gasteiger
Student Code:	144706IVCM
Supervisor:	Olaf Maennel, Ph.D

Tallinn 2016

Copyright Declaration

I hereby declare that I am the sole author of this thesis. The work is original and has not been submitted for any degree or diploma at any other University.
I further declare that material obtained from other sources has been duly acknowledged in the thesis.

....
(Date) (Florian Gasteiger)

Abstract

Drones are increasingly present in national airspace. Commercial and industrial use of this technology is rising, but proper regulations, a solid registration process and the possibility to fly beyond the line-of-sight continue to lag. This thesis introduces an architecture that helps to overcome these issues. Most significantly, this architecture proposes a shift from using WiFi to using mobile broadband networks and the Internet to facilitate communication between the drone and control station. A central server serves as a hub between the aircraft and the pilot. This central component provides security and authentication features such as VPN tunnels and public-key cryptography. It furthermore allows the owner of the server to grant and deny communication from certain pilots to specific Unmanned Aerial Vehicles. Due to concerns over trust and privacy, it is recommended that the central server be controlled by a governmental agency. In case of emergency, this authority can also disconnect the current pilot and take control of the aircraft. Detailed instructions and considerations on critical components and protocols that are necessary for an actual implementation are provided.

Kokkuvõte

Droonid jagavad teiste õhusõidukitega üha rohkem kontrollitavat õhuruumi. Droonitehnoloogia kommerts- ja tööstuslik kasutus on tõusuteel, kuid regulatsioonid, usaldusväärne registreerimisprotseduur ja võimalus lennata vaateulatusest väljaspool on jätkuvalt arengus maas. Käesolevas magistritöös tutvustame arhitektuuri, mis pakub nimetatud probleemidele lahenduse. Selle ühe olulisema komponendina liigutakse wifi kommunikatsioonitehnoloogialt mobiilioperaatorite pakutavate lahenduste ning Interneti peale drooni ja juhtimistöökoha vahel. Õhusõiduki ja piloodi vahel kasutatakse serverit, mis pakub turvalise kommunikatsiooni ja autentimisteenust, milles rakendatakse VPN tunnelid ja avaliku võtmega krüptograafiat. Lisaks võimaldab selline lahendus serveri omanikul reguleerida konkreetsete pilootide ligipääsu konkreetsetele droonidele. Lähtudes usaldusväärsuse ja privaatsuse tagamise vajadusest pakutakse välja, et sellist serveriplatvormi peaks pakkuma riiklik teenistus. Hädaolukordades saab selline teenistus õhusõiduki vajadusel üle võtta. Töös sisalduvad kirjeldatud lahenduse detailne analüüs koos juhiste- ning protokollidega, mis on vajalikud konkreetseks rakenduseks.

Contents

List of Figures

List of Tables

List of Abbreviations and Terms

AES	Advanced Encryption Standard	22
bps	bits per second	34
CBC	Cipher Block Chaining	60
CHAP	Challenge-Handshake Authentication Protocol	63
CSD	Circuit-switched Data	35
EASA	European Aviation Safety Agency	25
EDGE	Enhanced Data Rates for GSM Evolution	20
ENISA	European Network and Information Security Agency	59
ESP	Encapsulating Security Payload	62
EU	European Union	25
FAA	Federal Aviation Administration	24
FPV	First Person View	23
FTP	File Transfer Protocol	54
GCM	Galois/Counter Mode	60
GPRS	General Packet Radio Service	20
GPS	Global Positioning System	19
GRE	Generic Routing Encapsulation	63
GSM	Groupe Spécial Mobile	20
HD	High Definition	23
HSCSD	High-speed Circuit-switched Data	35

HSPA	High-Speed Packet Access	35
HSPA+	Evolved High Speed Packet Access	35
ICCID	integrated circuit card identifier	36
IMSI	international mobile subscriber identity	36
IMU	Inertial Motion Unit	22
IPSec	Internet Protocol Security	62
ISP	Internet Service Provider	32
ITU	International Telecommunications Union	36
LAI	Location Area Identity	36
LTE	Long-Term Evolution	20
L2TP	Layer 2 Tunneling Protocol	63
MAC	Message Authentication Code	62
MAV	Micro Aerial Vehicle	22
MCU	Micro controller Unit	22
NAT	Network Address Translation	23
PIN	Personal Identification Number	36
PKI	Public Key Infrastructure	47
PPP	Point-to-Point Protocol	63
PPTP	Point-to-Point Tunneling Protocol	63
PUK	Personal Unblocking Code	36
R^3**AD**	Registration Regulation and Range extension Architecture for Drones	29
SBC	Single Board Computer	32
SDU	University of Southern Denmark	27
SIM	Subscriber identification module	36
SSH	Secure Shell	61

SSL	Secure Socket Layer	64
TCP	Transmission Control Protocol	45
TLS	Transport Layer Security	62
UAS	Unmanned Aerial System	16
UAV	Unmanned Aerial Vehicle	12
UDP	User Datagram Protocol	45
UMTS	Universal Mobile Telecommunications System	20
US	United States	18
USB	Universal Serial Bus	55
VPN	Virtual Private Network	44

Chapter 1

Introduction

As with many other new technologies, drones were invented primarily for use in the military sector. Unmanned Aerial Vehicles (UAVs) became well-known to the public due to their questionable role in the US war against terror in Iraq. Already in 2005, when Hurricane Katrina hit the southern US, a request was made to allow unarmed Reaper drones (which were until then exclusively used in combat situations) to fly over the affected area to search for survivors. Its infrared camera with digitally enhanced zoom has the capability of identifying the heat signature of a human body from an altitude of ten thousand feet, making the aircraft an ideal search-and-rescue tool. [1] That was one of the first steps in including UAVs in civil airspace and using them for non-military missions. Since then, different kinds of drones were developed to specifically fit their intended purpose.

Despite the numerous purposes for drones today, (see chapter 2) they are still not a part of our everyday lives. One reason for that is the bad reputation of drones and the subsequent fear of them. Another reason is something far more technical, which is addressed by the following thesis. Civil drones are, with few exceptions, solely operated within the line-of-sight. This has some political and jurisdictional underpinnings but is mostly caused by the use of short range communication protocols between the drone and its control station. WiFi or Bluetooth limit the drone's operation range to a few hundred meters; yet they are still the default communication protocols for most low-cost drones. The majority of professional producers use direct link protocols to extend the flight range to up to 100km. But this capability comes with the price of needing additional range-extending stations in order to receive the signal. A different method is to utilize autopilots such as Pixhawk [2] for drones. This allows the UAVs owner to program static flight-routes into their aircraft in order to execute their mission. However, an autopilot is naturally not the ideal choice if one needs to receive a live video feed or be able to navigate the drone manually.

The following thesis introduces an architecture that allows control of the UAVs beyond the line-of-sight, and additionally provides an environment where third-party authorities have the possibility to engage with and monitor the aircraft. Within the last year, many states have discussed stricter regulations and compulsory registration processes for UAVs. Drones are increasingly present in national airspace; however proper regulations are not yet in place and registration processes are poorly built, making UAVs more threatening than productive. This work will not focus on recommending regulations but rather provides an architecture specifically developed to give federal agencies a method to introduce and enforce future limitations on drones. Strong authentication methods and encrypted communication moreover distinguish this architecture from current drone control mechanisms. It is therefore more appropriately-suited for use in commercial operations.
To overcome the limited communication range, this architecture introduces a way to control drones over the Internet using mobile broadband networks. The communication is no longer just between pilot and aircraft, but instead a third-party server is established that now manages

connection requests between the drone and the client. This new central authority also has the capacity to control who is allowed to fly drones, administer a database of drones and their owners and, if necessary, disconnect the pilot from the vehicle and take control of it. The central authority is certainly a very powerful entity; and therefore it seems logical that a federal agency takes over management of such a server.

From a cybersecurity perspective, a central authority that is in control of all communication is certainly questionable. Especially with the prevailing fear of state surveillance, the introduction of this architecture will most likely lead to significant privacy discussions and might be politically and culturally unacceptable for many democratic countries. However the purpose of this thesis is not to solve political issues but instead solely focuses on the technical aspects of secure drone communication. Licenses for drone pilots are almost nowhere required. This leads to an increasing number of aircrafts in the air, controlled by amateurs. The impact of this lack of organization can currently be followed in the news almost weekly. Incidents like the one in April 2016 where a drone crashed into a civil aircraft [3] are no longer a rarity and show very clearly the threat unlicensed drone pilots and their vehicles are for the national airspace.

Furthermore, todays drones are so weak in providing proper security for their communication (e.g. unprotected WiFi) that the usage for commercial purposes is almost reckless. The proposed central authority, however, can provide secure protocols that would make it more difficult for malicious actors to intercept drone communication. The central server within this new architecture would certainly be the main target of any attack and should be protected at the same level as other critical infrastructure systems. Even though this work does not answer the question if a central drone server is too dangerous for a real-world implementation, it provides a technical solution of how states can enforce stricter and better regulations for UAVs and therefore secure the national airspace more efficiently. This thesis moreover addresses specific protocols and proposes an architecture that is suitable for a resource-restricted system like a UAV. Challenges like broadband performance and driver issues are discussed alongside cryptography algorithms and protocols that are fitting for such systems.

The following chapter 2 gives a broader overview of intended purposes of civil drones as well as their current vulnerabilities. It gives a brief synopsis of current regulation and registration standards. Subsequently, chapter 3 will explain in detail the proposed architecture and its individual components. Chapter 4 shows the specific steps that need to be done to implement the architecture using the example of a Parrot AR.Drone2. It also gives detailed explanations of security protocols and describes necessary considerations for securely implementing VPN tunnels and public-key cryptography. Finally, chapter 5 summarizes the project and shows both the impact the new architecture has and the challenges that still need to be solved in a future work.

Chapter 2

Literature Review

2.1 Fields of application of civil drones

Despite the growing technological capabilities of Unmanned Aerial System (UAS), general society and the media are still primarily focused on evaluating the use of drones in war situations. The industrial sector, however, has taken up the task of exploring a variety of potential roles for drones.

Already in 2009, FedEx founder Fred Smith spoke about his desire to replace the company's fleet of package-delivery aircrafts with drones. [4] Then, in 2013, the online retailer Amazon started developing their service "Amazon Prime Air," [5] which will be launched in 2016, where "Octocopter" drones will deliver packages directly to the customers' doorsteps. The Australian startup Flirtey is also exploring drone delivery: *"We're interested in working with companies in time-sensitive last mile logistics; including online retail, fast food, letters and parcels, urgent medical delivery, etc."* [6] Flirtey is one of the first companies to develop and implement a functional concept of integrating drones into the civil airspace; it is now selling this service to interested clients.

The purpose of drones, however, extends far beyond just delivering goods: Drones can be very useful in situations where humans are unable to help, or when it is dangerous for them to do so. Rescue and safety missions are therefore an area in which UAVs have the ability to save lives, which is an objectively beneficial use for this technology. The following examples are unfortunately still mostly academic and are not yet implemented in the real world. However they do demonstrate the potential of drones and their capacity to support human rescue teams. Li et al. explain in their paper: “Drone-Assisted Public Safety Wireless Broadband Network” [7] how drones can be used to extend the Public Safety Network wireless coverage in the wake of a disaster to support the First Responder Network Authority in the United States. That approach would entail drones building up a multi-hop device-to-device communication scheme for the purpose of extending the wireless network coverage over regions where land-based relay can’t be deployed. Similarly, Noriki Uchida et al. presented, in 2014, a paper that proposes Autonomous Flight Wireless nodes for resilient networks consisting of Delay Tolerant Networks and Never Die Networks in order to communicate with isolated areas. [8] Both approaches would attempt to use a network of UAVs and their integrated communication protocols to reestablish communication networks within isolated areas or regions that had lost traditional communication during a catastrophe.

Andreas Birk et al. [9] have devised an alternative method where the drone’s digital video stream is used to create photo maps in real-time. These maps can then be used to guide rescue teams to the victims of catastrophes. In addition to Birk, Mahdi Asadpour et al. from the ETH in Zurich [10] have analyzed the capacity of drones’ video streams and cameras resolutions. Their purpose was to overcome the problem of drones having insufficient time to map an area well enough for a rescue team to come in. In their paper they came to the conclusion that a new amendment for the 802.11 standard is needed to assure both high-speed and reliable communication between UAVs.

A third big customer of drones can be found in the public sector: Law enforcement has begun using drones for a variety of purposes. The first time they were used was when a SWAT team from Texas launched a drone in order to get a last minute aerial sweep before they stormed the house of a suspect. [11] Ever since then, especially in the United States (US), law enforcement uses UAVs in order to surreptitiously get a fast, clear picture of a hidden property or area which they wish to enter. The advantages of using drones in these scenarios are: their ability to stay hidden, avoiding life threatening situations for officers, and the possibility to dodge national surveillance laws or the need for search warrants. [12]

The uses for UAVs, however, are far more than these examples: Farmers use them to monitor their fields [13], the movie industry utilizes them as mobile cameras [14]; Photographers make use of the drones integrated or attachable HD-camera to take aerial images or simple pictures from interesting angles. Some people have even hired drones to capture wedding photos. [15] The more flexible and mainstream the aircrafts get, the more uses there are and will be found for them in the future.

2.2 Vulnerabilities

Neither general society nor industry doubt the usefulness of drones, but still few realize how vulnerable most drones are. The following section will highlight some of the most crucial and common vulnerabilities today's drones have and discuss the potential threat they can become once a malicious actor exploits these vulnerabilities and hijacks the drone.

Similar to military drones, a lot of semiprofessional or professional drones are also equipped with a positioning system such as the Global Positioning System (GPS) or GLONASS. Whereas the US military can use their own (encrypted) version of GPS, the civil sector is only able to use the insecure and unencrypted version. Especially when thinking of letting drones fly autonomously in the future in order to execute some mission, this insecure technology leads to a huge security problem. GPS spoofing is not a completely new invention; Andrew Kerns (and others) showed that he can manipulate the GPS data of UAVs or even cause them to crash with a GPS spoofer that can be built for a few hundred dollars.[16]

Another popular attack vector is to exploit the, for many drones unsecured, or sometimes only barely secured, communication channel between the aircraft and the control station. Civil drones almost exclusively use protocols like WiFi, Bluetooth or ZigBee for their video and data stream. Each of these protocols, however, has been proven insecure. Hence it is not a big surprise that there are existing projects that exploit these precise vulnerabilities. Skyjack [17], for example, is an upgraded drone that cracks the WiFi hotspot of other drones in range, disconnects the true owner and remotely controls the cracked drones. Maldrone [18] is known to be the first backdoor into drones and is capable of killing the autopilot of Parrot's AR.Drones and therefore take over the drone completely. A third example is Skynet. [19] The concept of Skynet is to create a side channel to send command and control (C&C), which does not utilize the Internet, for shielding the botmaster's interaction with the botnet. For that, it uses drones to compromise networks and devices and enlists the victims in the botnet.

Depending on their intended purpose, hijacking a drone can have crucial impacts. It can range from packages not getting delivered to crashing UAVs into highly populated areas. Another problem that comes to mind by analyzing currently used communication protocols is the limited range of all these protocols. There exist a variety of approaches on how to secure WiFi or secure open ports, [20] [21] but in order to make drones more useful to the industry, the range drones can fly must also be increased.

One possibility to improve the range and simultaneously the security would be to replace the vulnerable WiFi protocol with a mobile communication protocol provided by the Universal Mobile Telecommunications System (UMTS) or Groupe Spécial Mobile (GSM). The below introduced architecture also takes this step. The security features of 3G and 4G protocols are certainly more difficult to overcome than unsecured WiFi but that doesn't mean that mobile communication can not be tampered with.
During the Black Hat conference in 2011 David Perez and Jose Pico explained how they managed to take full control over a victims data communication. The standard requirement that General Packet Radio Service (GPRS) and Enhanced Data Rates for GSM Evolution (EDGE) have to support the GEA0 encryption algorithm and the absence of mutual authentication makes 2G devices very vulnerable to their attack. Due to the fallback machanisms implemented in most UMTS devices to use 2G if 3G is not available the attack can even be extended to these 3G devices. [22]
In 2012 researchers from the Norwegian University of Science and Technology reported a deficiency in the specifications of the Authentication and Key Agreement protocols of many protocols administrated by the 3rd Generation Partnership Program (3GPP) such as UMTS or Long-Term Evolution (LTE). The flaw can be used by malicious actors to violate entity authentication properties and thereby impersonate other users. [23]

2.3 Incidents

Regardless of the above-mentioned technical flaws, drones are also lacking sufficient regulation and understanding, as illustrated by the following examples. An incident [24] that occurred in January 2015 where a small drone crashed into the grounds of the White House indicates two things: 1) Drones are a technology that can be used to easily access protected areas, and 2) drones are a technology that are not yet considered dangerous/critical yet. The ability to access restricted areas in combination with its high-resolution camera makes UAVs a perfect tool for surveillance and all the privacy violations that come with it. The lack of awareness as well as non-existent regulations make it very difficult to litigate against privacy violations by drones at present.

Another incident [25] in which a drone was used to carry drugs and mobile phones in and out of a prison in South Carolina (US) shows even more clearly how drones can be utilized for criminal tasks and how unaware society, including trained security staff, is of that danger.

Depending on their intention, there are a variety of actors prone to misuse drones for their own purposes. Hacktivists can also, for example, hijack drones to deliver their message or, as mentioned above, common criminals may use UAVs to deliver goods in otherwise difficult to access areas. The threat of thieves that simply want to capture the UAV or the package it's carrying in order to gain financial benefit is naturally always present and becomes more severe the more drones are in circulation.

Presently, UAV-manufacturing companies alone have the responsibility to secure their drones to a reasonable level; and since more security requires a lot more money, they will most likely resist implementing it. The need for national and/or local airspace regulations as well as security policies for UAV is long overdue and should be introduced as soon as possible. These improvements in legislation will not only better secure the airspace, but also promote the development of new projects in a secure environment.

2.4 Related Work

Section 2.2 provided an overview of vulnerabilities of drones the industry has to solve for the purpose of helping to uncover the full potential of UAVs and use them in a meaningful way for business.
The architecture described in this thesis also provides security features that mitigate the risk of drones getting hijacked or manipulated. Despite solely securing communication this work focuses on three additional issues: The extension of flight range, a proper registration procedure and a solution to enforce national regulations.
This section will review previously completed work by other entities in order to overcome these issues and show where they lack.

2.4.1 Range Extension

The range problem of UAVs is certainly one of the problems that drastically limits field application of drones. However, because it is a purely technical problem, it is an issue that industry can solve itself and is not dependent on governments or other external influences.
A solution that is becoming very popular and can be found in many professionally-used UAVs is the use of the MAV-Protocol [26] in combination with the Pixhawk Autopilot and the open source project QGroundControl. The Micro Aerial Vehicle (MAV) communication protocol was first developed by Lorenz Meier in 2009 and serves as a communication backbone for the MCU/IMU communication as well as the ground link communication. The protocol provides AES-256 encryption and with

proper range extension stations one can follow a drone up to 100 km. The whole three-pronged system is, however, only built to receive information regarding the UAV's current location and eventually its video stream. The drone's flight path can currently not be manipulated and is solely provided by the pre-programmed autopilot. Furthermore, the additionally needed large and expensive range extenders make the project ideal for companies such as the Estonian company Airborne Mechatronics [27], which uses this installation to inspect power lines with fixed winged UAVs. But implementing this approach is still rather unmanageable for a business that wants to operate a multiplicity of UAVs over a long distance at the same time.

A different approach that certainly solves the range problem is the use of the Internet as communication platform between drone and ground station. Leif Auke explains in his blog how he used two mobile phones to control a drone over 3G/4G. [28]
Auke developed an Android app which allowed a sender phone and a receiver phone to act as a remote control. With his communication software he enabled the phones to communicate peer to peer which allowed him to bypass Network Address Translation (NAT) restrictions. Because of this, he did not even need a server in between.
Auke's project is hardly more than a first experiment, but regardless, it does demonstrate how easily the range problem of WiFi or Bluetooth can be solved without expensive tools but instead by utilizing a network that is already well-established.

The Chinese company Skydrone [29] has also realized the potential of using a 4G network for drone communication and developed Sky Drone FPV. Sky Drone is supposedly the world's first low-cost, low-latency, digital full-HD consumer FPV solution that utilizes cellular networks. By using existing 3G or 4G/LTE networks, the Sky Drone FPV systems provide one with virtually unlimited range. The only requirement is cell tower coverage. Currently they only focus on transmitting a video stream through the LTE network and don't provide any way of controlling the actual drone. Their system also lacks encryption, therefore allowing everyone to view the video stream of the drone.

2.4.2 Regulation

In the US the authority that is in charge for all aerial vehicles is the Federal Aviation Administration (FAA). The current regulations contain rules like [30]:

- Fly below 400 feet and remain clear of surrounding obstacles
- Keep the aircraft within visual line of sight at all times
- Remain well clear of and do not interfere with manned aircraft operations
- Don't fly within 5 miles of an airport unless you contact the airport and control tower before flying
- Don't fly near people or stadiums
- Don't fly an aircraft that weighs more than 55 lbs
- Don't be careless or reckless with your unmanned aircraft – you could be fined for endangering people or other aircraft

Even though these rules were written to protect US citizens and to safely integrate UAVs into the airspace, they make commercial use of drones nearly impossible. If a delivery-by-drone company is not allowed to deliver to people because the regulations state that they can not fly near people, then their whole business concept is rendered moot. However these strict rules exist for a reason. As long as the FAA has no mean to enforce and control their regulations, their only option is to ban any drone - human contact across the board.

Besides the FAA in the US, the European Aviation Safety Agency (EASA) sets out three categories for drones [31] under the Advanced Notice of Proposed Amendment. The proposal was developed to allow new industry to grow while at the same time protecting people and goods. Drone regulations in the European Union (EU) are currently still subject to the individual countries' regulations. For example drones in Germany must not weigh more than 25 kg, whereas in Britain every UAV above 20 kg is subject to the same regulations as manned aircrafts. The EASA proposal tries to put an end to these individual regulations and create a framework that is valid for all EU member states.
The lowest category in the EASA proposal would cover low-energy aircraft, such as model planes, and would not require any license. Such drones would need to be flown within the line of sight, away from areas such as airports and nature reserves and up to an altitude of 150 meters. The "Specific operation" category (medium risk) requires authorization by National Aviation Authorities. A risk assessment performed by the operator must also be carried out since the aircraft shares airspace with other vehicles. A manual of operations should list the risk mitigation measures.
In the higher risk category, UAV and pilot would need to fulfill requirements similar to manned aircraft regulations. That would include proper licenses and trainings as well as maintenance of the aircraft.[32]

The goal of this thesis is not to come up with new or better regulations for integrating UAVs into national airspace. However regulations are certainly a important factor for commercial drone usage. The below introduced architecture does not depend on specific regulations but gives the proper authorities an instrument to enforce specific regulations. The architecture also comes with built-in features to check necessary licenses, which would be instrumental for some categories in the EASA proposal.

2.4.3 Registration

A third major problem with today's UAVs is the lack of authentication. In many cases, it is not easy to discover who is operating the drone and whether this drone belongs to the person flying it.

Since Dec. 21th of 2015 every owner of a Unmanned Aerial System in the US is now required by law to register it with the FAA [33].

The FAA provides an online registration platform for private owners who need to provide their credit card information, email and physical address. From there, one receives an identifying tag which must be attached to the drone. For commercial uses or aircrafts over 55lbs, the FAA needs even more detailed information about the vehicle, the owner and the usage of the UAV as well as how well it fulfills existing aircraft registration.

The US system, however, is certainly not a perfect solution. Physical license plates might show that someone actually owns the drone and also registered in the proper way. Unfortunately it does not reveal the actual pilot of the vehicle at any specific time. Section 2.2 showed how easy it is for malicious actors to hijack a drone and use it for their own purposes. A simple number plate will always hold the owner responsible and does not provide authentication which is badly needed for unmanned vehicles.

Outside the US, a number of other countries have developed a similar registration process. This development proves that most governments are not merely accepting the presence UAVs in the airspace, but also understand the need for a registration process. This process allows them to have a better overview of the vehicles in the air as well as the possibility to track the actual pilot of the UAVs in case of misbehavior.

The University of Southern Denmark (SDU) in Odense, Denmark, in cooperation with the Danish Transport and Construction Agency, realized that UAVs must be safe and accountable to the public if the economic benefits of drones are to be maximized. The SDU is currently working on two models that will allow the authorities to both identify and monitor drone activity online. The first approach is a small GPS that sends its position via the mobile network to a server when the drone is airborne; the second approach would include sending direct radio communication to a scanner on the ground. [34]
Both approaches are clearly an improvement to the physical license plate the US implemented since they not only show the owner of a specific vehicle but more importantly give the authorities the possibility to localize every drone in their airspace. That capability makes it far easier to spot suspicious behavior or locate vehicles in areas where they are not permitted entrance.

However neither of the models currently allow governments to interfere with suspicious UAVs. The authorities are able to spot odd behavior or airspace violation but they do not currently have a simple method to take over the drone or even force it to land. Recent developments like a drone that shoots a net on a misbehaving other UAV in order to catch it and carry it away might work in certain situations. [35] The response time and overall efficiency of such a solution however can not be compared with a method that would allow law enforcement to entirely take over the aircraft.
Another problem is that there is still no way to require the pilot to authenticate him or herself at the drone in order to control it. Therefore the responsible authorities might identify a hacked drone but still would not be able to identify the hacker.

2.5 Scope

The following work is primarily theoretical. Chapter 3 describes the overall architecture. Country or region specific aspects such as mobile network coverage, environmental or social issues are thereby out of scope. The work is to be considered as an initial framework which possible clients can adopt as they like and also modify to their specific needs. Chapter 4 describes a specific implementation of the architecture. However this is not the only way this concept can be implemented. While Estonia might consider to use its existing national ID cards for authentication other bigger countries like Germany need to consider scalability issues like multiple servers to handle the work load and reduce latencies. These state-specific issues are out of scope for this work. The main focus of chapter 4 is thereby more on analyzing suitable protocols and encryption schemes as well as discussing additional general challenges of an implementation than providing a solution for one specific purpose.

Furthermore this work does not consider political, legal or jurisdictional issues. Section 3.6 mentions basic trust and privacy considerations for the central authority but that does not replace specific security and trust analysis. And finally, this thesis does not cover modifications drone manufacturers need to make to integrate their aircrafts into the architecture. Chapter 4 describes necessary modifications for a specific drone in order to establish a connection with the central authority. Hardware and software modifications that are necessary to handle the additional cryptographic protocols or actually control the drone in the new environment are not part of this thesis.

Chapter 3

Architecture

3.1 Overview

The chapter above illuminated the problems and security concerns accompanied with allowing drones to be included in national airspace. Unlike conventional aircrafts or cars, it is not a simple task to determine who is actually controlling the UAV and therefore responsible for its actions. While planes can be forced to land, with the help of the military, there are no best practices or rules on how to intercept potentially misbehaving UAVs. Additionally, the potential of UAVs is still rather limited for commercial purposes due to the common usage of short range communication protocols. The following work describes an architecture that is capable of solving these problems. The Registration Regulation and Range extension Architecture for Drones (R^3AD) is easily scalable; even a rapid increase of UAVs in the near future can be handled without difficulty. Therefore this solution is superior to any currently existing systems.

Figure 3.1 shows the common architecture that is found in most Unmanned Aerial Systems. At startup, the aircraft establishes some sort of hotspot and the pilot then connects to it through the interface on his tablet or the control center. The most drastic departure from the current system in the new architecture is, as one can see in Figure 3.2, the introduction of a centralised server. The purpose of that server is to act as an intermediary link between the drone and its control station. The communication that was a direct link between drone and control station will now run via the server to which both sides have to connect to first. Obviously the owner of the server has a lot of power over the actual communication since all the traffic goes there first. Therefore Section 3.3 explains in detail the responsibilities of the server and proposes that the administration of it be put under governmental authority. One of the reasons, however, for introducing this third party is to be able to authenticate the pilot of the aircraft and also be able to restrict access to a certain UAV only to specific users. The architecture also recommends introducing the capability of a third party to entirely take over the UAV in emergency situations.

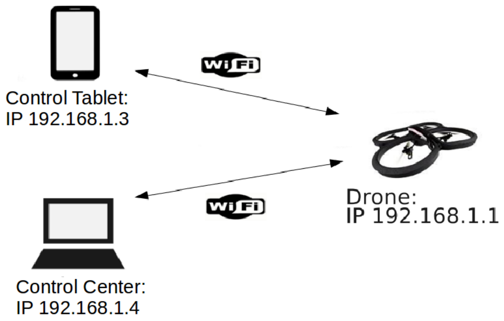

Figure 3.1. Common Architecture; Drone establishes WiFi hotspot to which clients connect to

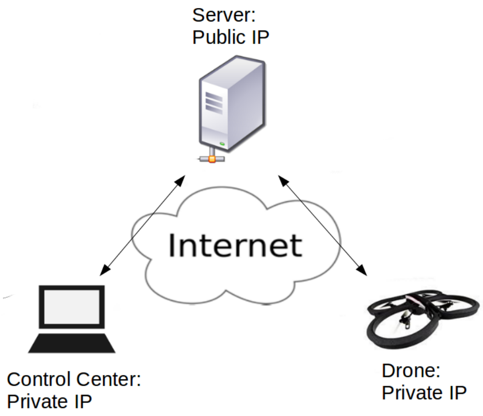

Figure 3.2. Registration Regulation and Range extension Architecture for Drones

By introducing the communication over a central server for drone control and navigation, conventional short range protocols are obviously no longer feasible. R^3AD suggests therefore to send all traffic over the Internet. Connecting drones over the network will also help to address many other challenges such as safety and security concerns, real time communications, potential interference with manned aircraft and supporting capabilities like beyond line of sight flight, once they are approved by the authorities. Mobile broadband makes a perfect transport layer for the control and/or video stream of UAVs because it is widely available, has a well-maintained infrastructure and cornerstones such as coverage and speed are constantly improving.

The subsequent sections describe in detail the components that are necessary and how they connect to each other in order to establish a stable and secure connection. The solution presented here should be considered a generic solution and therefore not merely suitable for one specific kind of UAS nor only one intended purpose. Section 3.4 describes in more detail how the actual communication will be established. By sending all communication over the Internet, a variety of security issues certainly need to be addressed. As mentioned above, the server needs to have the capacity to authenticate and even restrict access to a UAV; section 3.5 will expand upon these challenges. In order to develop R^3AD and to get a better understanding of the number of difficulties both the implementation and the architecture itself bring with it, chapter 4 explains subsequently to this chapter crucial problems and considerations based on one specific drone. Furthermore an analyzes of a number of different communication protocols based on their encryption and security features will also be done in the following chapter.

3.2 Aircraft Components

Almost all commercial UAVs are built around a main board with a micro controller as processor. Attached to the board are additional UART or communication interfaces. Operating systems are, in many cases, a minimalistic Linux kernel or another kind of busy box in which the stabilization, navigation and communication processes run. Hence a drone is in many ways not much more than an Single Board Computer (SBC) like the Raspberry Pi. Unfortunately like many SBCs, it requires additional software and hardware components in order to establish a connection to a mobile broadband network.

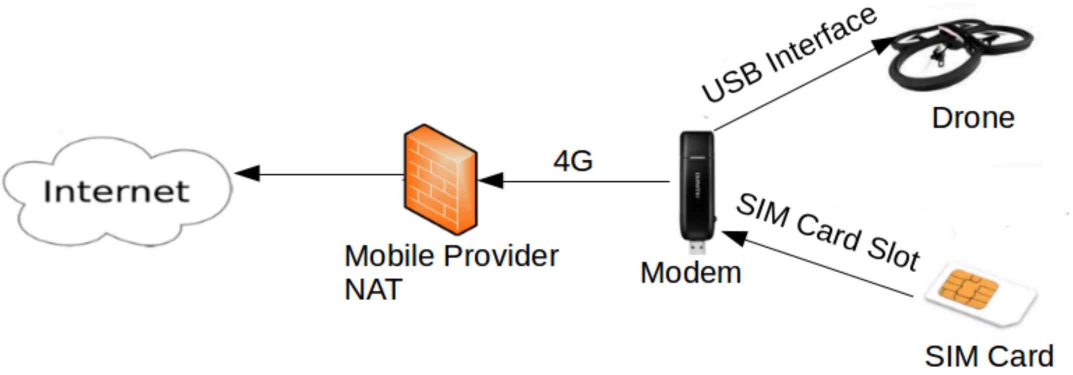

Figure 3.3. Drone Components

Figure 3.3 presents the necessary components. As shown, an LTE modem, attached to the drone, provides the required interface to connect to the Internet. A SIM card, which can be acquired by a local Internet Service Provider (ISP), contributes to the composition by providing the credentials necessary to connect to that ISP's broadband network. The subsections below describe further considerations for the individual components.

3.2.1 Drone

Since the architecture is developed to not just support one specific type of UAV but to be a generic framework which can integrate all kinds of UASs, the requirements on the vehicle need to be as few as possible. This design serves primarily to integrate UAVs in national airspace. As already mentioned in subsection 2.4.2, even the proposal of the EASA shows one category of UAVs that can be flown without further registration or licenses. A basic survey on UAVs in this category also shows that the majority of them would not fulfill the requirements that are necessary to be integrated in this architecture.

The targeted drones need to be able to handle a stable connection to a mobile network, which includes the management of a internal or external modem. Additionally section 3.4 introduces cryptographic communication protocols that require a certain processing power, which many micro UAVs can not provide. As the owner of the aircraft needs to be able to store cryptographic keys on the drone it is also necessary that the drone provides either a user interface that allows these operations or a open port which runs telnet or ssh to get direct access to the file system. Chapter 4 uses successfully the AR.Drone2 from the French manufacturer Parrot. The AR.Drone runs on a 1GHz ARMv7 micro controller. Similar aircrafts like the Solo drone from 3DR or DJI's Phantom drones, which are also rather popular for commercial uses have even more powerful hardware and can therefore easily be used in this architecture.

3.2.2 Modem

A modem is needed in order to establish a connection to a cellular network. In general, a modem is simply a network hardware device that modulates one or more carrier wave signals to encode digital information for transmission, and demodulates signals to decode the transmitted information. The goal is to produce a signal that can be transmitted easily and decoded to reproduce the original digital data. [36] Broadly speaking, modems can be used with many means of transmitting analog signals, from light emitting diodes (LED) to radio. In this scenario we need a modem that turns the digital signals from the drone into modulated electrical signals for transmission over the cellular network.

Modems are generally classified by either their symbol rate, which is measured in baud and denotes the number of times per second the modem sends a new signal, or by the amount of data they can send in a given unit of time, usually expressed in bits per second (bps). This available information capacity is commonly defined as the bandwidth of a device.

In order to determine what class of modem the architecture needs, one must first take a look at the necessary amount of data the modem has to be able to send and receive. Almost all UAVs are equipped with a camera to send live pictures back to the client. In the example of the AR.Drone2, that is used in chapter 4, the drone sends back a live video stream recorded by its 720p HD front camera (Table 4.1). To calculate the required bandwidth that is needed to receive a continuous video stream one can do the following:

720p resolution: $1280 * 720 pixel = 921600 pixel/picture$
One pixel needs $1B$ of storage $\rightarrow 115.2kB/f$
H.264 encoding [37] is the most commonly used standard for video compression and saves roughly 80% of storage space
$\rightarrow 115200B/f * 0.2 = 23kB/f$
$\rightarrow 23kB/f * 30f/s = 690kB/s = 5.5Mbps$ bandwidth

In addition to the video feed, the drone also sends telemetry data such as GPS coordinates or current speed to the control station. This kind of data, however, does not require a huge amount of bandwidth. It is safe to say that the modem needs to be able to handle a minimum bandwidth of about 6 Mbps for most common drones.

GSM [38], is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second generation (2G) digital cellular networks used by mobile phones, first deployed in Finland in July 1991. As of 2014 it has become the default global standard for mobile communications - with over 90% market share, operating in over 219 countries and territories. Even though GSM is the most widespread cellular standard in the world, it unfortunately does not meet the requirements for streaming HD video. As Table 3.1 shows, GSM networks can provide a maximum upload bandwidth of 177 Kbps (far lower than the required 6 Mbps.)

Generation	Technology	Max. Downlink	Max. Uplink
2G	CSD	9.6Kbps	9.6Kbps
	HSCSD	43Kbps	14Kbps
2.5G	GPRS	64Kbps	42Kbps
2.75G	EDGE	236Kbps	177Kbps
3G	UMTS	14.7Mbps	5.76Mbps
3.5G	HSPA	22Mbps	5.76Mbps
3.75G	HSPA+	42Mbps	11.5Mbps
4G	LTE	300Mbps	75Mbps

Table 3.1. Mobile Communication Protocols [39][40]

The systems developed later on, UMTS(3G) and LTE(4G), are not part of the GSM standard and therefore have much less coverage. As the figure shows, even third generation (UMTS) networks can barely fulfill the requirements. The latest UMTS release, HSPA+, can theoretically provide peak data rates up to 42Mbps in the downlink and up to 11.5 Mbps in the uplink. These rates would be fast enough to reliably communicate with the UAV but alas, they are reached very rarely.
Even though high speed mobile network coverage relies heavily on the ISP's network and the environment the UAV is in, the aircraft nevertheless needs to be able to handle these networks. Therefore it is necessary

that the UAV be equipped with a modem that supports 4G networks.

3.2.3 SIM Card

As with every mobile communication based on the GSM standard, a Subscriber identification module (SIM) card is necessary to connect to the mobile network. A Subscriber identification module is used to authenticate and identify subscribers on mobile devices. It is an integrated circuit chip that is intended to securely store the international mobile subscriber identity (IMSI) number and its related key. Additionally, a SIM contains the integrated circuit card identifier (ICCID) - a unique serial number, information related to the local network, a list of services the user has access to and a Personal Identification Number (PIN) and Personal Unblocking Code (PUK). [41]

Whereas the IMSI number is purely used to establish communication to the network, the ICCID is defined by the telecommunication standardization sector of the ITU in recommendation E.118 [42] as primary account number. Since SIM cards are generally bought by a local ISP, and the user usually has a contract with the ISP where he/she also has to provide personal information, the ICCID is suitable to map a person to a SIM card.

As the SIM card is (in this architecture) attached to a UAV, it is possible to identify the owner of a drone by looking at the ICCID. Similar to the drone registration procedure in the US that was mentioned in subsection 2.4.3, drone users now have to buy a SIM card instead of the FAA's license plate, in order to make their drone identifiable for the local authorities. In contrast to the US' system, a SIM card contains additional location information. Every location area of a mobile network has its own Location Area Identity (LAI). If a mobile device is in range of a Broadcast transmitter it stores its LAI on the SIM card and sends it back to the mobile provider. [40] The LAI is primarily used by the mobile provider for routing purposes; but it can also be used to locate devices, which gives authorities like the FAA or the EASA increased capabilities to identify if a UAV is flying in a restricted area, and if so, which one.

Enforcing registration with a SIM card leads to a rather crucial problem. Rural areas or regions with low/no network coverage will not be able to register at the mobile network and can therefore not be flown with this architecture. To still allow UAVs in these regions a hybrid system can be considered. Besides the primary mobile Internet, interface drones can still have a WiFi interface. In case of connection loss or an excessively weak signal, all data can be sent over the WiFi channel. Once the mobile connection is recreated, the drone switches back to this channel. This solution, however, is very demanding of the actual drone manufacturers. Both the drone's firmware and the control application need to be able to handle a dual-interface solution. The following work will not discuss a hybrid solution in further detail, but it is a possible solution to implement R^3AD in areas with general low network coverage.

3.2.4 Mobile Internet Service Provider

This architecture, unlike a conventional drone architecture, does not generate its own communication network but instead uses the Internet as a platform. The clear benefit of the proposed shift in architecture is the increase of potential distance between the drone and its control station. This advantage, however, comes with the negative aspects of relying on a external network. Still, Internet access is readily available in most populated areas of the world. Furthermore, nationwide mobile broadband coverage such as 4G, which has already explained crucial for the system to work, is gradually improving. Since complete mobile high speed coverage worldwide is, at least in the near future, not manageable, this architecture certainly offers less consistent availability than the original one. Therefore it must be emphasized at this point that the architecture introduced here is not suitable for isolated or poorly developed areas. It is, however, a very useful solution when it comes to regulating and enforcing these regulations near busy and commercially used airspace in areas with strong existing infrastructure.

Whereas a hybrid solution as mentioned above should only be considered for regions with general low network coverage, local connection problems can be overcome with already built-in solutions. Most modern drones already provide features which either force them to return to their home base or circle around the area until the connection to the pilot can be reestablished. These features can easily be adopted for R^3AD and provide a solid solution to momentary and local connection issues.

In addition to purchasing a drone the owner of the UAV needs to have a contract with a national Internet Service Providers, which will both provide a SIM card for the aircraft as well as either mobile or wired Internet access for the control station. For the aircraft's mobile connection, an ISP can (depending on the ISP) provide three options:

1. Public IPv4 address
   Public addresses have the benefit that both inbound and outbound connections can easily be established, which makes proper communication with the drone rather easy. However, the pool of public IPv4 addresses is almost depleted and many ISPs charge a lot extra for handing out a public address. Establishing a architecture based on these rare and expensive addresses is therefore unfeasible and would also diminish a possible scalability.

2. IPv6 address

   Other than IPv4, IPv6 has a much bigger address space (2^{128} instead of 2^{32}) which actually allows to assign every single device its own IPv6 address. Similar to a public IPv4 address this would solve major connectivity problems and possibly due to its uniqueness also replace the ICCID as identifier. On the down side IPv6 is still in its initial phase and many systems as well as some ISPs do not fully support IPv6 in their networks yet. Although usage of IPv6 addresses in this architecture does not lower its capabilities or complicate the system in any way, the proposed system does not rely on IPv6 due to its lack of current support.

3. Private IPv4 address

   Private IPv4 addresses are the standard for every private client of an ISP. The big disadvantage of using IPv4 for this architecture is that inbound traffic will normally get blocked, which makes it very complicated to establish a connection to the drone. Generally an ISP uses NAT to save public IP addresses. The dynamically allocated IP address the UAV gets is thereby part of the ISP's private network and therefore not reachable from the outside. The advantage of this one-to-many NAT is clearly the scalability. It is furthermore the default solution for almost all major ISPs which means it is considered both a well-established and rather inexpensive solution. The rest of the thesis will therefore be based on the drone being assigned a private IP address that only allows outbound traffic.

3.3 Central Server

The introduction of a central server as an intermediary link is certainly one of the most major changes this architecture proposes and adds an additional layer of complexity as compared to the original architecture. The reason to bring this additional component into play is, if nothing else, to provide a platform for proper registration of UAV owners and certified pilots.

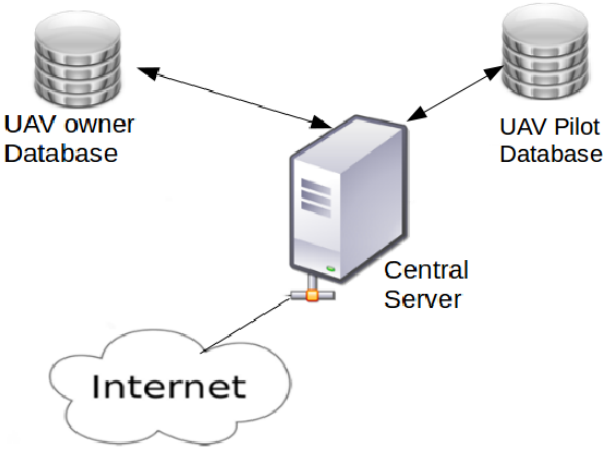

Figure 3.4. Central Server

The central authority is an organization that manages communication requests (3.4) between aircraft and pilot. Whoever is in control of the server not only has access to the two linked databases as shown in Figure 3.4, but also has the potential to take over specific aircrafts or disconnect the current pilot. The architecture is designed to have a government department control the central server. This thesis is based on the assumption that UASs will and should become useful tools for private industry. And while drones are currently still primarily a toy for hobby pilots, recent developments show that the commercial sector is strongly interested in the technology; UAVs will soon become a part of our everyday life. Given this shift towards use of UAVs it is imperative that the government gets involved and form a platform to enforce restrictions and regulations. Airspace is today solely regulated by national laws. It is precisely clear for conventional aircraft pilots where they are allowed to fly and what customs they have to abide to, whereas drone movement is either not at all or very sparsely regulated. Even though the actual regulations are up to the individual national jurisdictions, this thesis suggests to treat UASs more like the serious aircrafts they are, and no longer as toys. For this reason, as well as to more meaningfully manage the corresponding databases the central server of this architecture should be part of the country's transportation ministry. Depending on the political system, members of the law enforcement sector may need to have access to the server as well in order to interfere/take over an aircraft in emergency situations.

3.3.1 UAV Owner Database

One of the two databases that are proposed in this architecture stores information regarding ownership of individual drones. Similar to current system in the US, this architecture suggests that owners of a UAV need to register with their nationally responsible agency. After buying a UAV and purchasing a SIM card for it from a local ISP, it will then be necessary to submit the corresponding ICCID (see subsection 3.2.3) to the authorities. This system is akin to the well-established process of mobile phone contracts for which the ICCID is used as an account number. This architecture proposes using the ICCID as well to associate a drone to its owner. If and what additional data the specific agencies want to store is primarily up to their legislation and does not affect the performance of the general architecture. It would make sense, however, to store at least as much information as necessary to reliably map a person to a specific ICCID. Further information about the type of drone might be useful depending on the regulations of individual countries on

UAV categories.

The owner database, exactly like the “license plate” system in the US (subsection 2.4.3), only provides information about ownership. It can therefore be compared to license plates for automobiles. License plates, both physical and digital, only solve ownership issues. As with cars, the actual driver or pilot cannot be definitively determined.

However, a database like that is still very useful in determining the true owner of a stolen or lost UAVs. This will also give the authorities a primary contact for questions about a specific aircraft.

3.3.2 UAV Pilot Database

The second database the server has access to contains information about registered pilots. Currently some countries are debating whether or not a driver’s license for drones is necessary. This thesis strongly suggests the introduction of some sort of license for possible drone pilots. While a license to drive a car or fly a conventional aircraft is, in almost every country, compulsory, Unmanned Aerial Vehicle can be flown by everyone without necessarily having knowledge about airspace customs, no-fly zones, etc. Drones are a rather new technology, and lawmaking is certainly a slow process, but in the face of drastically increasing numbers of drones and the steadily increasing numbers of accidents and threats, it is a necessity to introduce licenses sooner rather than later. The specific requirements a UAV pilot needs to have are something the proper authorities of each country will need to decide and are therefore out of the scope of this thesis.

For this architecture, what is important is only that every possible pilot must register with the owner of the “UAV Pilot Database”; the central server needs to have access to that information as well. Whether a pilot owns his or her own UAV is therefore not relevant. Every person can be part of the “Owner Database”, the “Pilot Database”, or both.

3.4 Communication

The previous sections explained the individual components of the architecture. Now it is time to connect them in order to establish a secure and stable communication channel. As mentioned in section 2.2 malicious actors often exploit known vulnerabilities in the underlying communication protocol. Vulnerabilities in mobile communication systems like UMTS were also identified. Other than with the original architecture that relied on short range protocol, the hacker is no longer physically limited to a few hundred meters around the aircraft but can intercept the drones communication channels from all around the world. Strong encryption of both video and control channel can therefore no longer be an optional feature. It must be considered an essential characteristic of this new architecture. Besides the security issues, the architecture has one other major issue that makes establishing a initial communication more challenging. Other than the central server, both the UAV and eventually also the client's control station are protected by the ISP's NAT. That means that a connection request can only be started from the drone to the server or the client to the server but not the other way round.

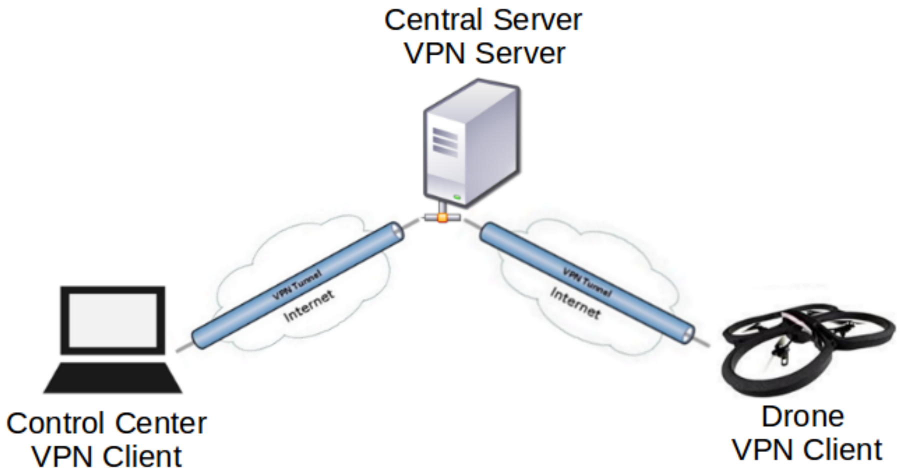

Figure 3.5. VPN Tunnels

3.4.1 Virtual Private Network

Figure 3.5 shows that this architecture relies on a Virtual Private Network [43]. As the name already suggests a Virtual Private Network (VPN) is a private network within a bigger public network. For this architecture the public network is the Internet and the private network contains all active clients as well as the UAVs. The central authority is also part of the private network and serves as VPN server. The advantages of a VPN are:

- Security:
  A VPN protects data while it is sent through a public network. It also provides encryption that ensures data confidentiality.
- Scalability:
  A Virtual Private Network is capable to grow without replacing the already existing VPN technology altogether
- Reliability:
  Clients can connect to the VPN any time; quality of connection stays the same regardless of the number of clients

VPNs come in two different types: Site-to-Site and Remote-access. [44] A site-to-site VPN connection connects two parts of a private network or two private networks. A routed VPN connection across the Internet logically operates as a dedicated Wide Area Network (WAN) link. Site-to-Site VPNs are mostly used by corporations that need to connect physically distant departments.

The other scenario would be for this architecture to use remote-access VPN. A remote access VPN connection is made by a single user, who connects to a private network from a remote location. The VPN server provides access to the resources of the network to which the VPN server is connected. The packets sent across the VPN connection originate at the VPN client. In order to establish a connection the VPN client needs to authenticate himself to the VPN server and, for mutual authentication, the VPN server authenticates itself to the client. In a remote access VPN connection over the Internet the client first initiates a dial-up connection to its local ISP. Using this physically established connection the client then initiates a VPN connection to the server. Once a connection is established the client can access the resources of the server's private network.

Most VPNs rely on tunneling to create a private network that reaches across the Internet. Tunnel protocols add a layer of security by encapsulating the entire package in a new package before it gets transported over the Internet. That encapsulated packet protects the contents by using encryption and ensures that the packet moves within a virtual tunnel. The original transport protocol does not get changed by this procedure.

3.4.2 Transport Layer Protocol

While tunneling protocols are located in the link layer (e.g. PPTP) or the Internet layer (e.g. IPSec) of the Internet Protocol Suite there is still need for a transport layer protocol. The major services a transport layer protocol provides are reliability, flow control, and multiplexing. The two most common used protocols for these purposes are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). [45]

- Transmission Control Protocol:
  TCP is used for connection-oriented transmissions. TCP provides flow control and data checksums. Ordered delivery and reliable transport features make TCP the best known and one of the most

used protocols in the Internet. These features however come with the price of a rather large header (20-60 bytes)

- User Datagram Protocol:
  UDP on the other hand is a connectionless protocol. With a header size of only 8 bytes, UDP does not guarantee reliability.Features like flow control and package ordering are not supported. Since UDP doesn't check for transmission errors, it is therefore faster than TCP.

For this architecture the recommended protocol is UDP. UAVs operate in an environment with potentially high package loss. Due to highly limited processing power as well as low power capacity on the drone, communication needs to be kept as simple as possible. Reliability and packet ordering functions are not really necessary and can even, in case of connection loss, result in overloading the drones processing power. Video streams are already commonly sent over UDP to ensure quality over reliability. It is highly recommended to also send the aircrafts AT-commands and telemetry data over UDP.

3.5 Authentication

Besides secure communication that the VPN provides for this architecture, proper authentication is also essential. Without authentication authorization is not possible and authorization is highly necessary in order to determine both licensed pilots in general and pilots for a certain UAV specifically.

Here, authentication is based on public key cryptography. Public key cryptography relies on integer factorization, discrete logarithm or elliptic curve relationships and is with current technology considered to be highly secure as long as the private key is kept secret and the environment is set up correctly. While public key cryptography is widely used for encryption and digital signatures, it serves in this architecture as authentication tool. The authority that holds the public key sets up a challenge. As long as the private key is kept secret it can be assumed that only the owner of the private key can solve this challenge and hence proves his identity. A major problem with this solution is the confidence that a particular public key is authentic. The usual approach to mitigate the risk of tampered keys is to use a Public Key Infrastructure (PKI) [46]. By creating a "web of trust" [47], with the help of certificate authorities, the Public Key Infrastructure substitutes individual endorsements of the link between user and public key. This solution might mitigate the problem but can not be considered to be highly secure. In contrast to PKI this architecture recommends that keys be exchanged only in person. The owner/pilot of a UAV can hand over his public key while he registers with the authorities. At the same time the owner can receive the server's public key, which then must be stored on the aircraft. Ideally, to ensure the integrity of the key, the registration procedure requires a personal visit by the proper authority and can not be done via the Internet.

This architecture has two different nodes that require authentication.

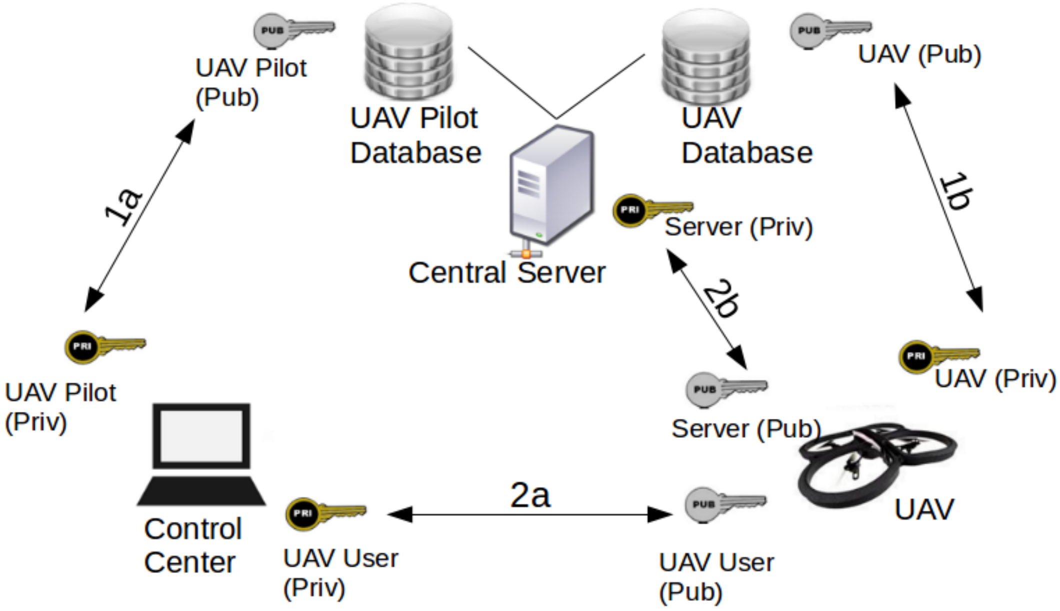

Figure 3.6. Authentication steps

1. Authentication at the central server: (Figure 3.6 link 1a and 1b) Both the drone and the clients need to authenticate at the server. The client or pilot needs to authenticate in order to verify if he/she has the necessary requirements to pilot a UAV in general. The associated credentials are stored in the “Pilot Database”. The client, as well as the aircraft, needs to be authenticated the the central server. Only if the UAV is properly registered and its data deposited in the “UAV Database” can the drone be considered a proper aircraft that fulfills all the specific terms regulations may demand. The authentication of pilot and aircraft is the preferred mechanism for a state authority to grant access to both the user and the aircraft, provided that required licenses, maintenance, or additional specifications are fulfilled.

   The authentication for client and UAV does not have to be an additional layer on top of the communication establishment. In fact, common VPNs already require authentication in order to connect to the private network. Therefore the authentication for these two instances can be accomplished together with the establishment of the remote access VPN communication.

2. Authentication at the aircraft: (Figure 3.6 link 2a and 2b)
   Once aircraft and client are verified by the central authority and the VPN tunnels are established, the pilot needs to authenticate at this specific aircraft in order to control it (link 2a). Other than determining whether a client fulfills the requirements to control a aircraft, in general it is not the central authority's responsibility which specific UAV a pilot has access to. The individual UAV owner holds this responsibility. Therefore the owner of a UAS needs to have access to the drone's file system and be able to store the public keys of all clients he/she wants to be able to control the aircraft. Even though it is not compulsory for the architecture to work it is recommended that the owner of the central server (aka the national authority) hand over its own key to the UAV owner (e.g. during the UAV registration process) and request the owner store the key on the aircraft. Thus, the central authority has a valid access key to the drone and can take it over in cases of emergency. To ensure that the authority's key is stored on the UAV, the server can simply try to start a control request to every connected UAV and disconnect the aircraft that denied the request.

3.6 Security and Trust considerations

Like many other IT systems R^3AD has three main requirements.

- Provide availability for the clients to connect to the central server
- Ensure the integrity of the transmitted data
- Provide data confidentiality

The central server is likely to be the main target of attacks. All communication between pilot and aircraft goes through it and depends on it. Therefore the availability and integrity of the server is the major concern in this architecture. Figure 3.7 can in no way be seen as a complete model of possible threats towards the central server but it is a basic model that helps to better understand two basic risks.

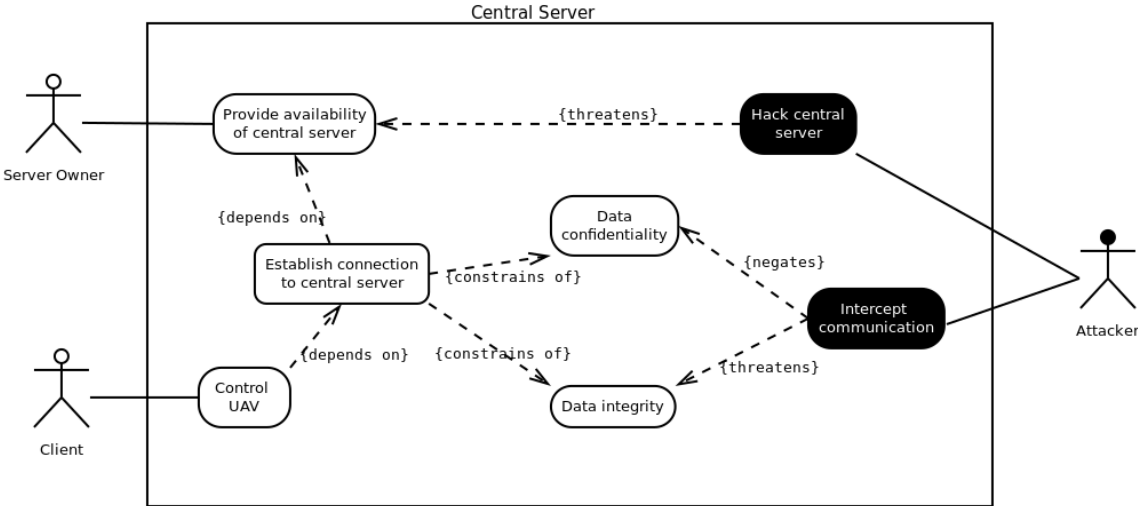

Figure 3.7. Risk model

By intercepting the data/video stream between the server and an end device an attacker not only negates the confidentiality of the transmitted data but also threatens its integrity. This can have multiple impacts:

Intercepting the data stream gives the attacker the ability to know where the aircraft is located and where it moves to. Eventually the attacker might even be able to take over the aircraft itself. Intercepting the video stream even leads to more serious problems. The attacker can receive the drone's video data and use it for his or her own purposes. This would lead to major privacy issues and therefore needs to be considered a very serious threat.
To mitigate this threat R^3AD uses strong encryption provided by the VPN; together with the two-layered authentication process, the risk of intercepted communication channels is mitigated. Even if an attacker succeeds in intercepting traffic, the data is encrypted and therefore it becomes unusable.

Secondly, the figure above shows an attack on the central server itself with the goal to damaging its availability. The availability of the central server is the main concern for all communication within this architecture. However even if an attacker can bring the server offline the connected aircraft would not necessarily crash. Even in current drone-communication architectures, connection issues are rather frequent. To overcome these issues, most modern UAVs have built-in functions that allow them to either return to their home station, land securely, or circle in the air until communication is restored. However a successful attack may have further consequences than halting all communication traffic. The databases explained in section 3.3 contain personal information of pilots and UAV owners. A security breach can therefore lead to a leak of personal data of numerous clients. Additionally the server holds keys to establish a connection to every aircraft within the system. An attacker in possession of these keys would have to ability to take over all connected drones. A more comprehensive security analysis for this threat is beyond the scope of this thesis; but there are examples of existing systems that handle similar security threats. Providers, such as Google, which offers cloud based services for millions of users, manage similar systems and work to ensure the data integrity and confidentiality for their customers.

Besides a potential attack against the architecture another issue needs to be addressed. By putting the central server into governmental control, the authorities have a tool that could allow them to receive video streams of every aircraft that is part of the system. Together with the feature of taking over a vehicle (intended for emergency situations) R^3AD gives a state a more precise and powerful surveillance tool than any satellite tracking system ever could. The question of how much trust citizens have in their governments and on what legal grounds the government could intercept a video stream or take over an entire drone goes beyond the scope of this work, but does require an in-depth analysis. It is necessary to mention this issue because it is a very basic consideration that could render the whole architecture unfeasible, especially given the misuse of surveillance power as made evident by the Edward Snowden leaks. On the other hand, most people trust their governments to at least somewhat protect their interests. Furthermore, a government entity (over a private sector company) is the most logical option for enforcing identification and air space regulations.

Chapter 4

Proof of Concept

Chapter 3 introduced the general architecture and its components. The following sections describe a basic implementation of this architecture. This chapter works to analyze the specific issues that might occur and to identify protocols that can handle the previous explained security considerations. This proof of concept only serves to show that a connection between UAV and server can be established and what steps are necessary to do so. Specific scripts to disconnect individual users or verify licenses will thereby not be addressed. Hence this implementation is neither a complete nor the best solution. Depending on the individual area of operation and the environment, different protocols and hardware might be more appropriate.

4.1 Components

4.1.1 Parrot AR.Drone2

To demonstrate the architecture's real world applicability, modifications will be made on a Parrot AR.Drone2 [48]. The AR.Drone is a beginner drone, rather than a commercially used aircraft. However, due to its mostly open-source code and its rudimentary but robust architecture, it is often used for academic or research purposes. The restricted resources of the Parrot drone also makes it a suitable drone for this project. By demonstrating a working communication on a low-level recreational drone, we ensure that industrial and commercial drones will be able to handle the recommended protocols.

Processor	1GHz ARMv7 Processor rev 2 (v7l)
Digital Signal Processor	800MHz video DSP TMS320DMC64x
Memory	1Gbit DDR2 RAM at 200MHz
OS	Linux version 2.6.32.9 (BusyBox)
Communication	USB 2.0 high speed for extensions Wi-Fi b,g,n
Total weight	380g / 420g (with indoor hull)
Flight time	12min
Max. Payload	250g
Sensors	3 axis accelerometer +/- 50mg precision 3 axis gyroscope 2000°/second precision Pressure sensor +/- 10 Pa 40kHz Ultrasonic sensors for ground altitude measurement Liquid Repellant (Hydrophobic) to avoid aqueous infiltration VGA CMOS Video Camera: 60fps QVGA (320*240) HD CMOS Video Camera: 30fps, 720p (1280*720)

Table 4.1. AR.Drone2 Specifications [48]

Table 4.1 shows the most important specification details of the AR.Drone2. In normal use, the drone, at start-up, creates an unprotected WiFi hotspot to which the clients are able to connect. Designated mobile applications enable easy-to-use control over the drone and are even capable of receiving the video stream of the drone's HD camera. A quick port scan reveals several open ports on the AR.Drone. They allow administrative Telnet access to the Linux system via port 23 and on port 21 the possibility for file up-/downloads via the File Transfer Protocol (FTP).

Further analysis shows that the drone uses UDP on port 5556 for receiving AT-Commands as well as on port 5554 to send telemetry data such as position or speed back to the client. The video stream that is sent back to the client device, however, uses TCP on port 5555. Fortunately, by default, the AR.Drone2 comes with a Universal Serial Bus (USB) port which makes it easier to connect an external modem. This is much better than, for example, the 3DR-Solo drone where one would first have to solder the port. It was not Parrot's intention for the port to be used to connect an external modem, but was rather meant to allow the addition of external storage devices to record large amounts of video data on the drone.

4.1.2 Huawei LTE Modem

The work done in this thesis uses a LTE modem from the Chinese company Huawei. The E3372 USB modem is capable of 4G mobile communication standards but can also, in case of low coverage, use 3G and 2G technology. Table 4.2 lists the most important specifications of the USB modem. What is pivotal for this implementation, besides small size and low weight, is that the E3372 supports Linux OS and does not need additional drivers to function.

Dimensions	Height: 88mm Width: 28mm Depth: 11.5mm Weight < 35g
Model	E3372
Form	USB Stick
Communication System	FDD: DD800/900/1800/2100/2600 UMTS: 900/2100 GSM:850/900/1800/1900
Speed	LTE FDD : Cat4 DL:150Mbps/UL:50Mbps @20M BW UMTS: DCHSPA+:42/5.76Mbps;21M/5.76Mbps;14M/5.76M HSUPA:7.2M/5.76M 2G: EDGE packet data service of up to 236.8kbps
microSD card slot	Yes
External Antenna Interface	TS-5*2
Receive Diversity	Yes
Operation System	WindowsXP,Vista,Windows7/8,MAC, Linux

Table 4.2. Huawei E3372 Specifications [49]

Another feature of this specific USB modem is that it comes with a microSD card slot that allows the user to store additional keys for authentication and encryption features, as described in section 3.4. IPv6, support for an external antenna for better reception and various built-in security features like a firewall or port forwarding settings make the modem a well-suited option for this implementation.

Even though the modem does not need external driver software, it is not recognized as such by the AR.Drone2. As already mentioned, Parrot did not include a USB Ethernet driver in its firmware, which thus requires modification of the drone's Linux kernel to make the modem operational. Appendix A gives detailed instructions on the additional kernel modules for the drone and how to compile and install them.

Once the modules are installed, the modem gets mounted and by inserting a SIM card, that can be bought by a local ISP, the drone should be able to connect to the Internet. In order to allow the modem to automatically establish a connection to the ISP's mobile network it is recommended to disable the authentication via PIN in the modem's settings. This can most easily be done by connecting the modem to a desktop PC and configuring it through the graphical user interface. Otherwise it is also possible to write a startup script for the modem that initializes the connection and provides the necessary PIN code.

4.1.3 Raspberry Pi Server

The central server of the architecture is represented by a Raspberry Pi 2. The Raspberry Pi [50] is based on a 900 MHz ARM Cortex-A7 processor. It is cost-efficent, minimizes energy consumption, and is well-supported in software; thus making it a solid server for this implementation. For further use, the Raspberry was equipped with a Debian ARM distribution for which installation guides [51] can easily be found on the Internet. In order to function as the central server, the SBC needs to have access to the Internet with a static IP address, which can also be purchased from a local ISP. The specific steps on how to set up the Raspberry Pi as well as the gateway to the Internet [52], [53] are well documented on the Internet and therefore not further discussed. The Raspberry Pi now has access to the Debian package repository which makes the installation of a database management system straightforward. Besides several paid editions of MySQL, the Oracle Corporation [54] also offers an open-source version. MySQL is one of the most commonly used database management systems and due to its high compatibility with other software, it is also a good system for this implementation.

After the successful installation of MySQL, the architecture requires the creation of two databases. One that contains information about the pilots and the other information about the drone. The pilot side of the architecture will not be discussed further. However an entry in the UAV owner database can be created. The entry contains at least the ICCID, which can be commonly found on the SIM card itself and a password hashed with a SHA-2 algorithm. These fields will be used for authentication as described later in section 4.3

4.2 Communication Protocols

Once the individual components are set up, the next step is to establish communication between them. The drone being protected by the ISP's NAT, which only allows outbound traffic, makes it more challenging to test the the communication. Using a tool like "Netcat" [55], however, helps to debug problems and test if the drone can in fact reach the server and further shows that communication between the devices is in general possible.

To secure the communication and to establish a real two-way communication channel between UAV and server additional consideration need to be made. Hence the following provides an overview of current protocols and encryption schemes and analyzes whether or not they can be used in the Registration Regulation and Range extension Architecture for Drones.

4.2.1 Encryption Algorithms and Operation Modes

R^3AD requires both symmetric and asymmetric cryptography. While asymmetric cryptography is used for the authentication features, symmetric cryptography is required to ensure data confidentiality between the devices. Since a detailed explanation of every encryption algorithm and their operation modes is beyond the scope of this work, the following will briefly introduce crytography methods their associated challenges.

Asymmetric cryptography

Asymmetric cryptography requires the generation of two keys. It is widely used for authentication purposes and to securely transmit a key for symmetric cryptography. The security of the cipher relies on the challenge of nearly impossible to solve mathematical problems. Algorithms of the RSA family [56] use factorization of large numbers as a challenge and the Diffie-Hellman [57] problem is based on discrete logarithms. Without going into too much detail on the mathematical principles, current recommendations of European Network and Information Security Agency (ENISA) [58] suggest the use of keys with 3072bit. Based on modern available computing power, this key size ensures security for the next 5-10 years.

Symmetric cryptography

While stream ciphers are in general faster and more simple in their architecture, they are not compatible with a VPN solution and are therefore no longer included in the following analysis. Data confidentiality will be assured through a block cipher. Block ciphers are essentially a keyed pseudo-random permutation on a block of data with a fixed length. A block cipher is not an encryption scheme but a component that goes into making a scheme [59]; this is mostly done by a mode of operation. The security of an encrypted block can technically never be proven, but it has mathematical assumptions similar to the factorization problem of large moduli. According to ENISA, this means that with current technology the minimum key size used to encrypt a block of data is 128bit. On this basis, only a few block ciphers can be considered secure for today and the near future. Only the AES and Camellia block cipher can fulfill these requirements and be considered secure. Both the Camellia and the AES cipher have a block length of 128bit and support key sizes of 128bit, 192bit and 256bit, which guarantees their security also within the near future. Both ciphers have theoretically a vulnerability to algebraic attacks, however these attacks have not yet been successfully executed. [58]

To ensure confidentiality or authenticity of a block cipher, a variety of modes of operation is available. The mode of operation describes how the underlying cipher can be repeatedly applied to data that expands the size of one single block.

- The Cipher Block Chaining (CBC) mode [60] is one of the most commonly used solutions. Other than, for example, the Electronic Codebook (ECB) mode, CBC does not encrypt every block separately but rather each block of plaintext is XORed with the ciphertext of the previous block. The dependency on the previous block, however, makes the CBC mode unsuitable for R^3AD. In an environment with potential high packet loss, a cipher chain is not an appropriate solution.

- Another mode of operation is the counter (CTR) mode. [61] The counter mode basically turns a block cipher into a stream cipher. It generates the next key stream block by encrypting successive values of a "counter". The CTR mode has the advantage that the encryption/decryption does not rely on the previous block and can therefore also be parallelized.

- Whereas most basic modes of operation such as CBC or CTR are only IND-CPA secure, authenticated encryption modes are IND-CCA, IND-CPA and IND-CVA secure. The Galois/Counter Mode (GCM) for example provides data confidentiality, integrity and authenticity. [62] Due to it's performance and efficiency, GCM is one of the most widely adopted modes of operation. GCM combines the basic CTR mode with the Galois mode of authentication, thus resulting in a higher throughput than CBC mode that can also be easily pipelined or parallelized.

4.2.2 Cryptographic Protocols

Kerberos

Kerberos [63] is primarily an authentication service that allows a client to authenticate his or herself to multiple services. The system uses an trusted authentication server that grants tickets to the clients. These tickets allow them to prove their identity to one another. Kerberos is primarily based on symmetric key primitives. In its current version(5), it supports AES for the first time. A security analysis by Boldyreva and Kumar [64] came to the conclusion that Kerberos can be considered secure as long as a secure MAC algorithm for the checksum is used and the underlying primitives meet standard notion of security. However Kerberos can just be used for authentication services. It does not provide data confidentiality which makes it rather unsuitable for this implementation.

SSH

Secure Shell (SSH) was originally designed as a replacement for insecure shell protocols like Telnet. [65] Today it is one of the most commonly used protocols for providing a secure channel between two networked computers. SSH provides confidentiality and integrity of a message. The transport layer of SSH handles key-exchange based on Diffie-Hellman and host authentication by combining this with a signature. For client authentication, password or public-key cryptography can be used. Besides its most common use for login to a remote machine, SSH also supports tunneling. Unlike SSH-1, the current version 2 is currently considered secure. Information leaked by Edward Snowden in 2014, however, suggested that the NSA may be able to decrypt some SSH traffic. Moreover, SSH has another more practical flaw that makes it a less suitable option for R^3AD. SSH often freezes when the connection is interrupted. In a environment like this where one has to reckon with temporary connection loss, this issue is decisive.

TLS

The Transport Layer Security (TLS) protocol aims to provide a confidential channel between two entities. [66] It is mostly used for securing traffic between an unauthenticated web browser and an authenticated web site. However due to its ease of use and high availability, it is implemented in a wide variety of libraries for various purposes. In addition to confidentiality, TLS also provides both authentication via public-key cryptography and data integrity via Message Authentication Code (MAC). The two choices in TLSv1 and TLSv1.1 (MAC-then-Encode-then-Encrypt and MAC-then-Encrypt) have been proven insecure. The current version TLSv1.2 now supports authenticated encryption, which is the only form TLS currently can be considered secure with.

IPSec

Unlike SSH and TLS, Internet Protocol Security (IPSec) provides security on the network layer, which renders re-engineering of applications unnecessary. [67] IPSec is primarily used for creating VPN tunnels; when used in tunneling mode, it provides security for the entire IP packet through encapsulation. Every IPSec implementation requires a Security Policy Database. The entries in this database define processing rules for certain types of traffic. The entries point to at least one Security Association, which contains (amongst other information) cryptographic keys and initialization vectors. To initialize and distribute this information securely, larger implementations of IPSec often rely on the Internet Key Exchange (IKE) Protocol. [68] IPSec comes in two protocols, Authentication Header (AH) and Encapsulating Security Payload (ESP), however only ESP provides data confidentiality through symmetric key encryption. Due to a number of successful attacks against the encryption-only mode of ESP, only the authenticated encryption mode provides integrity protection and can be considered secure.

4.2.3 VPN Tunnel Protocols

R^3AD requires the usage of a VPN to establish secure two-way communication. While 3.4 explained the purpose and features of a VPN generally, this subsection focuses on the specific protocols that can be used to create a VPN.

PPTP:

Point-to-Point Tunneling Protocol (PPTP) is one of the oldest methods for implementing a VPN and was published in RFC2637 [69] in July 1999. PPTP establishes a TCP connection to port 1723, which is then used to initiate a Generic Routing Encapsulation (GRE) tunnel to the same port. The tunnel is thereafter used to carry encapsulated packets from the layer 2 Point-to-Point Protocol (PPP). PPTP is the default tunnel implementation for all Microsoft systems but is not limited to it. PPTP also supports Linux, Mac and Android operating systems. Even though authentication and encryption features are not described in the original specifications, most current Microsoft Windows versions contain authentication and encryption in their PPTP stack. The most commonly used authentication protocol used with PPTP is CHAP. The issue is that PPTP requires a TCP connection. As established in section 3.4.2, R^3AD requires a UDP solution; therefore PPTP is not a suitable method for this implementation. Moreover, both Microsoft versions of the Challenge-Handshake Authentication Protocol (CHAP) have been proven vulnerable. Even Microsoft itself recommended discontinuing use of PPTP in favor of upgrading to a more secure VPN tunnel method, such as Layer 2 Tunneling Protocol (L2TP).

L2TP:

The Layer 2 Tunneling Protocol is published under RFC2661 [70] and has its origins in two other tunnel protocols; PPTP and Cisco's Layer 2 Forwarding Protocol (L2F). The latest version, (L2TPv3), is specified in RFC3931 [71] and provides additional security features that were not provided in the original version. Unlike PPTP, L2TP relies on UDP to establish and maintain the tunnel. Once a tunnel is created (and properly set-up), L2TP allows the user to bypass NAT restrictions and establish a stable bidirectional communication. The L2TP protocol itself does not contain any encryption or authentication features but relies instead on encryption protocols that it passes within the tunnel. That is why L2TP is mostly implemented in combination with IPSec. Its high support throughout different operating systems and strong security through the use of IPSec, makes it a generally solid method to establish a Virtual Private Network. However, it is rather complex to configure. L2TP also requires specific UDP ports to function, which could be blocked by certain ISPs and therefore would prevent the NAT traversal and ultimately break the communication channel. In general, L2TP is not flexible enough for the proposed architecture; particularly because it is limited to only a few protocols to enable proper data encryption and authentication features.

OpenVPN:

OpenVPN contains an open-source implementation of the TLS and Secure Socket Layer (SSL) protocol and is written by James Yonan, published under the GNU General Public License as an open-source application to implement VPNs. OpenVPN can create tunnels via both TCP and UDP and does not require specific ports, which makes it easier to bypass NAT or firewall restrictions. For authentication, OpenVPN offers several solutions: pre-shared keys, certificate-based, and username/password-based authentication, which are all implemented by default into the application.

OpenVPN heavily depends on the OpenSSL library to provide encryption as well as authentication. The various protocols within OpenSSL and support of OpenVPN throughout different platforms make it a very flexible and powerful application. Its large amount of extensions also makes it possible to integrate third party scripts that can authenticate against databases like MySQL. Unlike L2TP or PPTP, OpenVPN is not naturally integrated into any operating system by default. It requires the installation of additional software as well as the generation of a configuration file for both client and server. [72] [73]

4.2.4 Recommendations

Based on the previous analysis and together with recommendations from ENISA, [58] only a few combinations of protocols and algorithms can be considered not only secure but also suitable for R^3AD and this specific implementation.

L2TP together with IPSec is a possible solution. However this is true only if IPSec is used in authenticated encryption mode. Suitable combinations are therefore:

IPSec-with-AES-CTR and
IPSec-with-CAMELLIA-CTR

Besides L2TP, OpenVPN is also recommended for a potential implementation. Since OpenVPN relies on OpenSSL, the cryptographic protocol is always TLS. As mentioned above, TLS can only be considered secure by using encrypted authentication such as GCM or CCM. However because the CCM mode is rather inefficient, the only other recommendation is:

TLS-with-AES-GCM-SHA-2

Since R^3AD is a system with very restricted resources it is also necessary to consider that AES-256 is up to 40% slower than AES-128. Even though 256bit provides better security, its use consumes far too many resources. The 128bit version consumes much fewer resources and is still considered secure for the near future. Depending on the actual resources the UAV can spare for encryption, a 128bit solution should be preferred over 256bit in the majority of cases.

4.3 Establish Communication

The above analysis of protocols and ciphers concluded that either L2TP-with-IPSec or OpenVPN-with-TLS would be a secure option for this implementation. However the following will only illustrate the implementation of one specific combination: OpenVPN-with-TLS. OpenVPN is, for this implementation, simpler to setup and requires less additional software than L2TP.

The choice of OpenVPN automatically leads to the use of TLS as cryptographic protocol and, to follow the previous recommendation, TLS will be implemented with AES-128 in Galois/Counter Mode. To actually install OpenVPN one can download the necessary packages from the standard Ubuntu/Debian repository, which makes it very easy for the Raspberry Pi server. Since the drone's firmware doesn't include repositories, OpenVPN should be downloaded from the official homepage [74] and then installed manually on the AR.Drone2. OpenVPN does provide an ARM version, various manuals and installation guides [75] on the Internet to make the installation on the drone a simpler process. OpenVPN relies on OpenSSL for all cryptographic protocols, so OpenSSL naturally needs to also be installed on both the drone and the server. OpenVPN is dependent on the packages lzo and pam. Similar to OpenVPN, OpenSSL, lzo and pam are also well supported on ARM architectures; together with thorough documentations [76][77][78] the installation should proceed without complications.

OpenVPN provides two modes: routed or bridged VPN. However routing is more efficient and provides a greater ability to selectively control access rights on a client-specific basis. Hence this implementation uses a routed VPN. The individual configuration files are provided in Appendix B.1 (server.conf) and Appendix B.2 (client.conf). Other than the recommended cipher (AES-128-GCM) also client-to-client communication and authentication via username/password is included in the files. For future real world implementations, the use of cryptographic tokens via PKCS#11 should be preferred over username/password authentication. Since this implementation only serves as a proof of concept for establishing a basic connection, advanced authentication methods are not included. Eventually, on the Raspberry Pi server, a static route (*route add -net 10.10.10.0 netmask 255.255.255.0 dev tun0*) has to be added and IPv4 forwarding needs to be enabled to allow a connection.

The php script provided in Appendix B.3 demonstrates a rudimentary plug-in that allows OpenVPN to access the previously created databases. It verifies the username/password combinations which are stored in the database for the clients' VPN connection requests.

Once the necessary keys are generated, combined with the configurations scripts, and copied to the right locations [75], it is then possible to establish a VPN connection. The modification of the drone's flight software and control applications are out of scope, therefore this work will not explain what additional changes need to be made in order to actually fly the drone within the new architecture. However it is verified that the server can actually ping the drone thus proving R^3AD viable.

Chapter 5

Summary

The previous chapters described the architecture in detail and provided detailed considerations for a specific implementation. This chapter gives a overview over the architecture and section 5.2 and 5.3 summarize the advantages of R^3AD and addresses challenges and issues that need to be solved in a future work.

5.1 Conclusion

Figure 5.1 shows once again a complete picture of the architecture introduced in chapter 3. The following will now demonstrate a step-by-step guide for a UAV-owner to fly his or her aircraft within R^3AD. The reader should keep in mind that most of the steps that require the generation of keys can either be automated with the registration process or embedded in the control center's firmware. For upgrading existing UAVs to be compatible with this architecture a certain knowledge of Linux operating systems and cryptography is required.

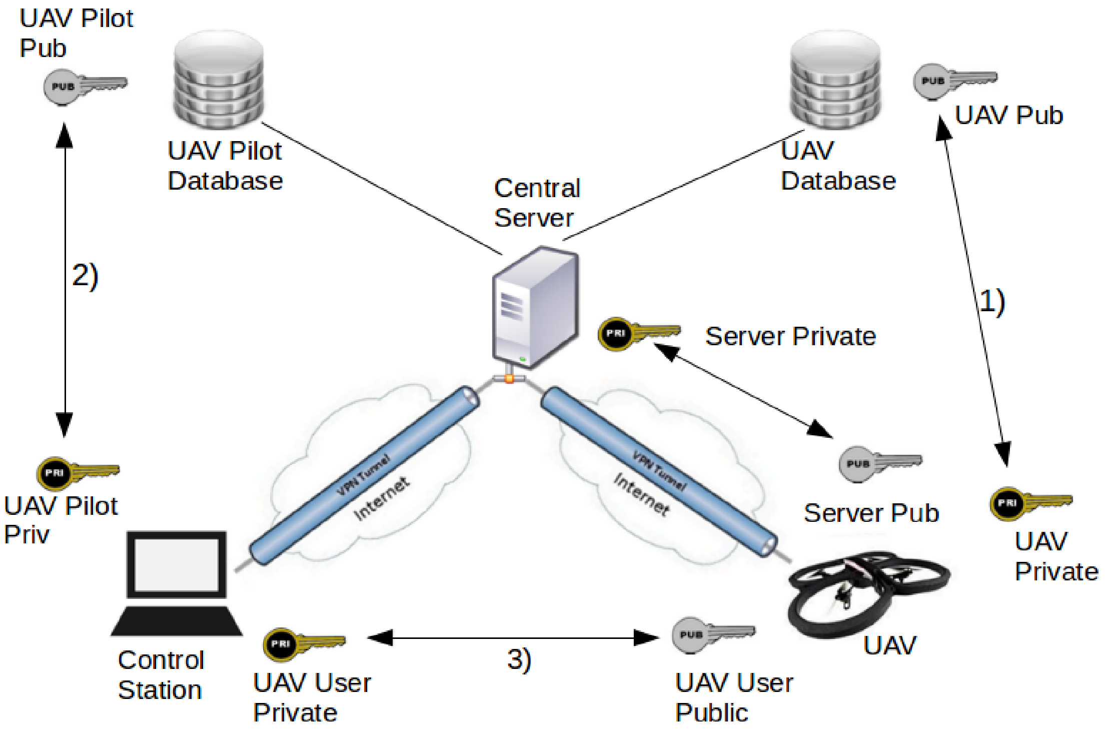

Figure 5.1. Final Architecture

1. Once a new drone is bought, the owner must purchase a SIM card for the drone. For UAVs that do not come with a integrated modem, a highspeed LTE modem is also necessary, which should then be connected to the drone in order to access a mobile broadband network (see section 4.1 for a possible example). Future UAVs might already be equipped with a modem which would simplify the initial configuration.

2. Once the components are purchased and the UAV is configured to connect to a mobile network, the next step is to register at the central authority. Registration is divided in two categories:

   - The owner must register as a pilot, which, depending on the local regulations, can require further training or licenses. However this step only needs only to be completed once, and is not required every time a new UAV is bought. For registration, the pilot creates a public-private key pair. The public key is combined with additional personal information (specified by the local authority) handed over to the authority and the private key stays with the owner.

- The second registration step is for the recently purchased aircraft. Every UAV must be registered. The owner then needs to create another private-public key pair and deliver the public key to the authorities. Together with the key, the SIM card's ICCID will be registered by the authorities, which will serve as the drone's identification number, as it does for mobile phones currently. In return the drone owner gets the public key of the central server. Depending on the actual regulations, further details about the drone might be necessary. The drone's private key and the server's public key both have to be stored on the aircraft itself, which can be done by a specific interface developed for this purpose or by accessing the drones file directory via FTP or SSH.

3. Besides the two key pairs that are required for registration at the authorities, the UAV owner also needs to create an additional key pair of which the public key is also stored on the drone, and the private key stays with the owner. This key pair is meant to authenticate a pilot at a specific drone. Besides his or her own public key, the owner can store additional keys on the drone from other individuals of their choosing, in order to permit other people to fly it.

4. The UAV starts up and establishes a connection to its mobile broadband provider. Then it contacts the central server to establish a VPN connection. The previously-created key pair (Figure 5.1 1)) is used to authenticate the aircraft and authorizes the drone to establish this connection. At this point, the central authority may then have the ability to check whether the owner has complied with maintenance guidelines or other restrictions.

5. In order for the pilot to control the drone he/she also needs to contact the central authority first. The pilot's private key (Figure 5.1 2)) authenticates him or her as a certified pilot and authorizes him or her to establish a VPN tunnel to the central server's private network.

6. Finally, once pilot and aircraft are part of the Virtual Private Network, the pilot can initiate a connection request to an aircraft (identifiable by its ICCID). Again the previously created key pair (Figure 5.1 3)) authorizes only permitted pilots to get access over a specific vehicle. The everyday access to the UAV will not take significantly longer than it does with current architectures.

5.2 Impact

R^3AD is designed to give national authorities better ability to enforce regulations and registration processes for UAVs. Enhancing the structure for registering and regulating drones will support the vast variety of purposes drones currently have and will have in the future. One of the major advantages of this architecture is that states can actually issue and demand licenses for drone pilots and at the same time have the ability to enforce these licenses. The authorities furthermore would have the ability to enforce registration for every aircraft and to prevent access to aircrafts that are not up to code. Where current communication protocols for drones lack in both range and security, R^3AD makes use of the existing mobile broadband network to extend the flight range and uses VPN tunnels to ensure a secure, encrypted communication channel between entities. The architecture is not designed to support only one specific protocol but, as shown in more detail in 4.2, leaves room for a variety of VPN tunnel, encryption and authentication protocols. This is intended to preserve the flexibility of the architecture in order to support the range of regulation requirements that different nations may come up with. Moreover, R^3AD can be easily expanded without any adverse affects to existing pilots and their aircrafts.

In summary, R^3AD is an architecture that gives the national authorities enough power to enforce all kinds regulation for UAVs. It also allows both private persons and the commercial sector to operate their UAVs in a secure environment and additionally extends the possible flight range of the aircrafts further than with any other current architecture.

5.3 Challenges

Even though the architecture has a lot of benefits and provides a solution for many current problems of UAVs, R^3AD is not yet ready for implementation.

One of the big advantages of R^3AD is also a significant challenge. The usage of mobile broadband network allows a theoretically unlimited flight range. However, as explained in subsection 3.2.2, it requires a stable high speed network connection. Currently only few countries can provide a nationwide LTE coverage; and even within these countries the limited coverage above a certain altitude limits the drone's freedom. Currently Intel and AT&T are cooperating to solve this issue in the US. [79] In order to implement this architecture nationwide, problems such as data roaming and fall-back to 3G or even 2G networks need to be analyzed first.

Some other minor issues need to be considered as well. R^3AD introduces the ICCID number as the primary identification number for drones. While the general usage of the number makes sense it also needs to be considered that the ICCID is solely stored on the SIM card. Unfortu-

nately these cards are very easily replaceable and therefore may not be ideal for verification of ownership. A solution like SIM lock used with mobile phones could be implemented. This would prevent misuse and therefore make the system more secure. Confidentiality is also essential for the public-private key distribution. As mentioned in section 3.5, the distribution of keys is a big security concern and may allow malicious actors to conduct man-in-the-middle attacks. Hence the only secure solution to this problem is an in-person key exchange; which can most easily be done during the registration process.
Whereas some drones have a firmware that is open-source and thereby easily customizable for this architecture, the majority of UAVs have closed source code. Therefore drone manufacturers need to be convinced to alter their drones' software to comply with R^3AD. At the same time, additional consideration about behavior in situations of communication or network loss need to be made and included in the firmware.

The biggest threat of this architecture is the almost unlimited power of the central authority. Putting this power in the hand of a national agency is certainly more reasonable than giving it to a private company. However without a proper security and trust analysis of the architecture as a whole and the central server in specific a implementation of R^3AD is irresponsible. Proper regulations on privacy issues caused by the UAVs cameras and laws on when the authority is allowed to intervene or take over a aircraft are a crucial step to make the architecture admissible. Additionally bigger states like Germany or the US need to consider introducing multiple servers to balance the workload and therefore find solutions to issues of responsibility and scalability.

This thesis lays a foundation to include UAVs more efficiently and securely in national airspace. A real world implementation however can only be conducted if policymakers, ISPs and manufacturers of UAV can work together cohesively.

Bibliography

[1] A. Robinson, "FAA authorizes Predators to seek survivors", US Air Force, Ed., Feb. 2006. [Online]. Available: http://archive.is/20120629064215/http://www.af.mil/news/story.asp?storyID=123024467 (visited on 12/03/2015).

[2] Pixhawk, *PX4 Autopilot*. [Online]. Available: http://www.pixhawk.org/start (visited on 02/18/2016).

[3] "Drone hits british airways plane as it prepares to land at heathrow", The Telegraph, Ed., Apr. 2016.

[4] A. Chris, "Fred Smith: FedEx wants UAVs", DIY Drones, Ed., Feb. 2009. [Online]. Available: http://diydrones.com/profiles/blogs/fred-smith-fedex-wants-uavs (visited on 12/18/2015).

[5] Amazon, "Amazon Prime Air", [Online]. Available: http://www.amazon.com/b?ie=UTF8&node=8037720011 (visited on 11/25/2015).

[6] Flirtey. [Online]. Available: http://flirtey.com/#about (visited on 11/25/2015).

[7] L. Xu, G. Dongning, Y. Huarui, and W. Guo, "Drone-assisted public safety wireless broadband network", *IEEE Wireless Communications and Networking Conference (WCNC)*, 2015.

[8] N. Uchida, M. Kimura, T. Ishida, Y. Shibata, and N. Shiratori, "Evaluation of wireless network communication by autonomous flight wireless nodes for resilient networks", pp. 180–185, 2014.

[9] A. Birk, B. Wiggerich, H. Blow, M. Pfingsthorn, and S. Schwertfeger, "Safety, security, and rescue missions with an unmanned aerial vehicle (UAV)", *Journal of Intelligent and Robotic Systems*, 2011.

[10] M. Asapdour, Domenico Giustiniano, and K. A. Hummel, "From ground to aerial communication: dissecting WLAN 802.11n for the drones", 2013.

[11] P. Finn, *Domestic use of aerial drones by law enforcement likely to prompt privacy debate*, The Washington Post, Jan. 2011.

[12] J. Vacek, N. Dakota, and S. A. Frazier, “Unmanned aerial vehicles: threat or asset to airborne law enforcement?”, *AIR BEAT*, 2009.

[13] *Uavs in the agriculture industry*, Farmingdrones.com, 2013. [Online]. Available: http://farmingdrones.com/about/ (visited on 04/20/2016).

[14] B. Child, *Hollywood applies to use drones on film productions*, The Guardian, Jun. 2014.

[15] *Couples taking wedding photography to new heights with drones*, CBC News, Aug. 2014. [Online]. Available: http://www.cbsnews.com/news/drone-for-hire-as-wedding-photographer/ (visited on 04/19/2016).

[16] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, “Unmanned aircraft capture and control via GPS Spoofing”, *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[17] S. Kamkar, *Skyjack*. [Online]. Available: http://samy.pl/skyjack/ (visited on 10/12/2015).

[18] R. Sasi, *Maldrone the first backdoor for drones*. [Online]. Available: http://garage4hackers.com/entry.php?b=3105 (visited on 12/03/2015).

[19] T. Reed, J. Geis, and S. Dietrich, “SkyNET: a 3G-enabled mobile attack drone and stealth botmaster”, *Usenix Workshop on Offensive Technologies*, vol. 5, 2011.

[20] J.-S. Pleban, R. Band, and R. Creutzburg, “Hacking drones - untersuchungen zur sicherheit der Parrot AR.Drone 2.0”, Fachhochschule Brandenburg, Ed., Mar. 2014.

[21] M. Peacock, “Detection and control of small civilian uavs”, Edith Cowan University, Ed., Jun. 2014.

[22] D. Perez and J. Pico, Eds., *A practical attack against GPRS/EDGE/UMTS/HSPA mobile data communications*, ser. Black Hat DC, Taddong, Jan. 2011.

[23] J.-K. Tsay and S. F. Mjølsnes, “A vulnerability in the umts and lte authentication and key agreement protocols.”, in *MMM-ACNS*, I. V. Kotenko and V. A. Skormin, Eds., ser. Lecture Notes in Computer Science, vol. 7531, Springer, 2012, pp. 65–76.

[24] A. Newcomb and A. Mullen, “Drone crashes at White House; its operator contacts Secret Service”, Los Angeles Times, Ed., Jan. 2015. [Online]. Available: http://www.latimes.com/nation/nationnow/la-na-drones-white-house-20150126-story.html (visited on 11/23/2015).

[25] C. Parsons, K. Hennessey, and K. Lee, "Drone carrying drugs and phones crashes outside South Carolina prison", abc News, Ed., Jul. 2014. [Online]. Available: http://abcnews.go.com/Technology/drone-carrying-drugs-phones-crashes-south-carolina-prison/story?id=24791025 (visited on 12/01/2015).

[26] L. Meier, "Model aircraft operations", 2009. [Online]. Available: https://www.faa.gov/uas/model_aircraft/ (visited on 02/25/2016).

[27] Airborne Mechatronics, 2016. [Online]. Available: http://airbornemechatronics.com/ (visited on 10/14/2015).

[28] L. Auke, "Using a 3g/4g internet connection and two mobile phones", DIY-DRONES, Ed., Nov. 2015. [Online]. Available: http://diydrones.com/profiles/blogs/using-a-3g-4g-internet-connection-and-two-mobile-phones (visited on 02/25/2016).

[29] Skylab Mobilesystems Ltd., *Sky drone FPV: the world's first digital HD consumer FPV solution*. [Online]. Available: http://www.skydrone.aero/index.php (visited on 02/23/2016).

[30] Federal Aviation Administration, "Model aircraft operations", Feb. 2016. [Online]. Available: https://www.faa.gov/uas/model_aircraft/ (visited on 02/25/2016).

[31] European Aviation Safety Agency, "Introduction of a regulatory framework for the operation of unmanned aircraft", Dec. 2015. [Online]. Available: http://easa.europa.eu/system/files/dfu/Introduction%20of%20a%20regulatory%20framework%20for%20the%20operation%20of%20unmanned%20aircraft.pdf (visited on 02/21/2016).

[32] S. Robins, "Drone regulations for europe", Oct. 2015. [Online]. Available: http://www.heliguy.com/blog/2015/10/08/drone-regulations-for-europe/ (visited on 02/02/2016).

[33] Federal Aviation Administration, "Unmanned aircraft systems (uas) registration", Dec. 2015. [Online]. Available: https://www.faa.gov/uas/registration/ (visited on 02/25/2016).

[34] B. Beach and S. K. Pedersen, "Number plates for drones will improve safety", University of Southern Denmark, Ed., Oct. 2015. [Online]. Available: https://www.faa.gov/uas/registration/ (visited on 02/25/2016).

[35] M. Goodrich, "Drone catcher: Robotic falcon can capture, retrieve renegade drones", Jan. 2016.

[36] J. R. Davey, "Modems", *Proceedings of the IEEE*, vol. 60, no. 11, pp. 1284–1292, Nov. 1972.

[37] AXIS Communication, *H.264 video compression standard*. [Online]. Available: http://www.axis.com/files/whitepaper/wp_h264_31669_en_0803_lo.pdf (visited on 03/02/2016).

[38] J. Pelkmans, *The GSM standard: Explaining a success story*, 3. Taylor & Francis, 2001, vol. 8, pp. 432–453.

[39] *Digital cellular telecommunications system (Phase 2+)*, vol. 13.1.0, TS148018, European Telecommunications Standards Institute, Apr. 2016.

[40] Technical Specifications Group, 3rd Generation Partnership Project (3GPP), 2015. [Online]. Available: http://www.3gpp.org/specifications/specifications (visited on 03/12/2016).

[41] P. Wu, W. Du, and H. Cui, "A design of sim card interface for general purpose", in *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions*, vol. 2, Jun. 2002, 1271–1274 vol.2.

[42] International Telecommunication Union, "Series e: overall network operation, telephone service, service operation and human factors", no. E.118, May 2006.

[43] A. G. Mason, *Cisco secure virtual private networks*. Cisco Press, 2002.

[44] Microsoft, *How VPN works*. [Online]. Available: https://technet.microsoft.com/en-us/library/cc779919%28v=ws.10%29.aspx (visited on 04/02/2016).

[45] D. E. Comer, *Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture*, 4th ed. Prentice Hall, 2000.

[46] *Public key infrastructure*, Microsoft, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/bb427432%28v=vs.85%29.aspx (visited on 01/14/2016).

[47] B. Schneier and N. Ferguson, *Practical Cryptography*. John Wiley & Sons, 2003.

[48] Parrot, *AR.Drone 2 - Specifications*. [Online]. Available: http://ardrone2.parrot.com/ar-drone-2/specifications/ (visited on 02/18/2016).

[49] Huawei, *E3372 - specification*. [Online]. Available: http://consumer.huawei.com/en/mobile-broadband/dongles/tech-specs/e3372.htm (visited on 02/15/2016).

[50] *Raspberry pi 2 model b*, Raspberry Pi Foundation, 2015. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ (visited on 01/14/2016).

[51] *Wiki - raspberry pi*, Debian. [Online]. Available: https://wiki.debian.org/RaspberryPi (visited on 01/22/2016).

[52] J. Morgan, *Tutorial: How to set up a raspberry pi web server*, 2012. [Online]. Available: https://www.jeremymorgan.com/tutorials/raspberry-pi/how-to-raspberry-pi-web-server/ (visited on 02/02/2016).

[53] D. Wilson, *Ultimate raspberry pi home server*, instructables. [Online]. Available: http://www.instructables.com/id/Ultimate-Pi-Based-Home-Server/ (visited on 02/02/2016).

[54] *MySQL - the worlds most popular open source database*, Oracle Corporation. [Online]. Available: https://www.mysql.com/ (visited on 02/03/2016).

[55] G. Giacobbi, *The gnu netcat project*, 2006. [Online]. Available: http://netcat.sourceforge.net/ (visited on 02/10/2016).

[56] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the Association for Computing Machinery*, vol. 21, no. 2, pp. 120–126, 1978.

[57] W. Diffie and M. E. Hellman, "New directions in cryptography", *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, 1976.

[58] *Algorithms, key sizes and parameters report*, version 1, European Network and Information Security Agency, Oct. 2013.

[59] H. C. Van Tilborg and S. Jajodia, *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2014.

[60] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code", *Journal of Computer and System Sciences*, vol. 61, no. 3, pp. 362–399, 2000.

[61] H. Lipmaa, P. Rogaway, and D. Wagner, "Ctr-mode encryption", in *First NIST Workshop on Modes of Operation*, 2000.

[62] D. McGrew and J. Viega, "The galois/counter mode of operation (gcm)", *Submission to NIST. http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf*, 2005.

[63] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer, "Kerberos authentication and authorization system", in *In Project Athena Technical Plan*, Citeseer, 1988.

[64] A. Boldyreva and V. Kumar, "Provable-security analysis of authenticated encryption in kerberos", in *Security and Privacy, 2007. SP'07. IEEE Symposium on*, IEEE, 2007, pp. 92–100.

[65] T. Ylonen and C. Lonvick, *The secure shell (ssh) protocol architecture*, RFC4251, Internet Engineering Task Force, 2006.

[66] T. Dierks, *The transport layer security (tls) protocol version 1.2*, RFC4246, Internet Engineering Task Force, 2008.

[67] N. Doraswamy and D. Harkins, *IPSec: The new security standard for the Internet, intranets, and virtual private networks.* Prentice Hall Professional, 2003.

[68] D. Harkins, D. Carrel, *et al.*, *The internet key exchange (IKE)*, RFC2409, Internet Engineering Task Force, 1998.

[69] *Point-to-point tunneling protocol (PPTP)*, RFC2637, Internet Engineering Task Force, Jul. 1999.

[70] *Layer two tunneling protocol "l2tp*
, RFC2661, Internet Engineering Task Force, Aug. 1999.

[71] *Layer two tunneling protocol - version 3 (L2TPv3)*, RFC3931, Internet Engineering Task Force, Mar. 2005.

[72] S. Riedel, *OpenVPN - kurz und gut.* O'Reilly, Aug. 2007, ISBN: 3897215292.

[73] M. Feilner, *OPENVPN: Building and integrating virtual private networks.* PACKT publishing, 2006.

[74] OpenVPN Technologies, Inc. [Online]. Available: https://openvpn.net/index.php/access-server/overview.html (visited on 04/12/2016).

[75] *Howto*, OpenVPN Technologies, Inc. [Online]. Available: https://openvpn.net/index.php/open-source/documentation/howto.html#quick (visited on 04/12/2016).

[76] OpenSSL Software Foundation. [Online]. Available: https://www.openssl.org/ (visited on 04/12/2016).

[77] *Lzo*, oberhumer.com GmbH. [Online]. Available: http://www.oberhumer.com/opensource/lzo/ (visited on 04/12/2016).

[78] *Linux-pam*, 2014. [Online]. Available: http://www.linux-pam.org/ (visited on 04/12/2016).

[79] "AT&T and Intel® to test drones on LTE network", 4GWE, Ed., Feb. 2016. [Online]. Available: http://www.mobilitytechzone.com/lte/news/2016/02/22/8319869.htm (visited on 02/25/2016).

Appendices

Appendix A

Installing Modem driver on AR.Drone

Using the *dmesg* command on the drone, and plugging in the modem into the USB port, the modem is mounted as a “Mass Storage Device“. The reason for that is that Parrot does not include USB modem drivers in the drone’s firmware. In order to get the modem running, drivers need to be compiled specifically for the drone’s architecture.

```
:~# dmesg

usb 1-1: New USB device found, idVendor=12d1, idProduct=1f01
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1: Product: HUAWEI_MOBILE
usb 1-1: Manufacturer: HUAWEI_MOBILE
usb 1-1: SerialNumber: 0123456789ABCDEF
scsi1 : SCSI emulation for USB Mass Storage devices
scsi 1:0:0:0: CD-ROM   HUAWEI   Mass Storage   2.31 PQ: 0 ANSI: 2
```

The first step of installing drivers on the drone is to figure out which drivers are needed for the modem. For that, the modem should be connected to a Linux desktop machine. By issuing the *lsusb* command on the Linux machine, one can see by looking at the product ID (`12d1:1f01`) that the Modem was mounted correctly.

```
:~# lsusb
```

```
Bus 001 Device 003: ID 12d1:14dc Huawei Technologies Co., Ltd.
Bus 001 Device 001: ID 1d6b:0002
```

To understand what drivers are necessary for the modem, the *usb-devices* command is helpful. It lists for the Huawei modem (and for many other USB modems) cdc_ether as the driver.

```
:~# usb-devices
T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=02 Dev#=  5 Spd=12  MxCh= 0
D:  Ver= 2.10 Cls=02(commc) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=12d1 ProdID=14dc Rev=01.02
S:  Manufacturer=HUAWEI_MOBILE
S:  Product=HUAWEI_MOBILE
C:  #Ifs= 3 Cfg#= 1 Atr=80 MxPwr=2mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=02(commc) Sub=06 Prot=00 Driver=
    cdc_ether
I:  If#= 1 Alt= 0 #EPs= 2 Cls=0a(data ) Sub=06 Prot=00 Driver=
    cdc_ether
I:  If#= 2 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50 Driver=usb
    -storage
```

modinfo cdc_ether gives additional information about the driver and even more importantly lists *usbnet* as a dependency. That means it will not be enough to copy the cdc_ether driver to the drone; all dependencies need to be available on the drone. *usbnet* again depends on *mii* in order to work. *mii* however has no further dependencies which means that these three modules are all that is needed for the modem to properly work on the drone.

```
:~# modinfo cdc_ether
filename:       /lib/modules/3.19.0-51-generic/kernel/drivers/net
    /usb/cdc_ether.ko
license:        GPL
description:    USB CDC Ethernet devices
author:         David Brownell
srcversion:     AE719599AF8786FFA964CD2
depends:        usbnet
intree:         Y
vermagic:       3.19.0-51-generic SMP mod_unload modversions 686
signer:         Magrathea: Glacier signing key
sig_key:        C4:6B:2A:77:A5:EC:50:06:D1:10:34:A8:8E:9E:42:9C
    :25:C1:86:BC
```

```
sig_hashalgo:   sha512
```

The next step is to figure out the kernel version the drone is using.

```
:~# uname -r
2.6.32.9-g980dab2
```

Now that all the necessary information has been compiled it is time to prepare a Linux machine with the necessary software. Since the drone is running on an ARM micro controller, a special compiler is needed. *ncurses-devel* is also needed later on for driver selection.

```
:~# sudo apt-get install gcc-arm-linux-gnueabi
:~# sudo apt-get install libncurses5-dev
```

The machine has now all the software it needs and the next step is to download and extract the Linux kernel which can be found on `https://www.kernel.org/pub/linux/kernel/`. It is important to download the same kernel version that runs on the drone.

```
:~# wget https://www.kernel.org/pub/linux/kernel/v2.6/linux
    -2.6.32.9.tar.bz2
:~# tar xjvf linux-2.6.32.9.tar.bz2
```

Open the Makefile in the root directory of the extracted kernel and edit the "EXTRAVERSION" to the exact version found on the drone (e.g. .9-g980dab2). Next the kernel.config file has to be downloaded from Parrots Developer homepage (`http://developer.parrot.com/`) and copied into the kernel directory.

```
# cp kernel.config linux2.6.32.9/.config
# make ARCH=arm menuconfig
```

In the menu config, the module for the *cdc_ ether* driver can be found and activated in `Device Driver -> Network Device support -> USB Network Adapters -> CDC Ethernet Support` Once the necessary modules are activated the kernel can be built.

```
:~# export ARCH=arm
:~# export CROSS_COMPILE=arm-linux-gnueabi-
:~# make
```

Once the process is finished the compiled drivers can be found in the `drivers/net` and `drivers/net/usb` directory of the Linux kernel. All the is left to do now is to copy the files to the drone by using its FTP link and activate the modules.

```
:~# telnet 192.168.1.1
:~$ cd /data/video
:~$ insmod cdc_ether.ko
```

Since the modem driver should be automatically loaded at startup, it is recommended to create a startup script.

```
:~$ cd /data/video
:~$ vi modem_start.sh
    mod_path="/data/video"
    insmod $mod_path/mii.ko
    insmod $mod_path/usbnet.ko
    insmod $mod_path/cdc-ether.ko
```

Let that script run during the drone's init phase. Therefore the following lines need to be added to `/etc/init.d/rcS`

```
if [ -f /data/video/modem_start.sh ];then
        /data/video/modem_start.sh
fi
```

Appendix B

OpenVPN Configuration

B.1 OpenVPN Server.conf

```
##################################################
# OpenVPN server config file for                 #
# use on the Raspberry Pi server                 #
#                                                #
##################################################

# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one.  You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
proto udp

# "dev tun" will create a routed IP tunnel,
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN interface.
dev tun

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key).  Each client
# and the server must have their own cert and
# key file.  The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys.  Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca droneserver.crt
cert central_server.crt
key central_server.key  # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
dh dh1024.pem

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.10.10.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.10.10.1.
server 10.10.10.0 255.255.255.0

# Maintain a record of client  virtual IP address
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt


# Suppose that you want to enable different
# firewall access policies for different groups
# of clients.  There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#     group, and firewall the TUN/TAP interface
#     for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#     modify the firewall in response to access
#     from different clients.  See man
#     page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# the TUN/TAP interface to the internet in
# order for this to work properly).
# CAVEAT: May break client's network config if
# client's local DHCP server packets get routed
# through the tunnel.  Solution: make sure
# client's local DHCP server is reachable via
# a more specific route than the default route
# of 0.0.0.0/0.0.0.0.
;push "redirect-gateway"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
client-to-client

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
```

```
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# AES-128-GCM as recommended in the thesis
cipher AES-128-GCM

# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
user nobody
group nobody

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
# or the other (but not both).
;log         openvpn.log
;log-append  openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
```

```
verb 6

# To enable authentication of a client via
# username/password specify the script
# necessary script
auth-user-pass-verify mysql_authent.php via-file


# An additional keypair is required
# to enable tls-authentication
tls-auth ta.key 0
```

B.2 OpenVPN Client.conf

```
##############################################
#   Client-side OpenVPN 2.0 config file      #
#   This config file can be used for the     #
#   Parrot AR.Drone2                         #
##############################################

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# In case of problems check the firewall
# settings
dev tun

# Connection to a UDP server
# Use the same setting as on the server
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote droneserver 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the drone server.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
user nobody
group nobody

# Try to preserve some state across restarts.
persist-key
```

```
persist-tun

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca droneserver.crt
cert parrot.crt
key parrot.key

# Verify server certificate by checking
# that the certicate has the nsCertType
# field set to "server".  This is an
# important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server".  The build-key-server
# script in the easy-rsa folder will do this.
;ns-cert-type server

# If a tls-auth key is used on the server
# then every client must also have the key.
tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
cipher AES-128-GCM

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo

# Set log file verbosity.
verb 6
```

```
#To enable username/password authentication
auth-user-pass

#TLS-authentication key for integrity
# verification
tls-auth ta.key 1
```

B.3 MySQL Authentication Script

```
#!/usr/bin/php

<?php
$HOST="localhost";
$USER="UAVServerRoot";
$PASS="uavpassword";
$DATABASE="uavowner";
$TABLE="drones";

// Read username / password from environment variables
//$username=getenv('username');
//$password=getenv('password');

// Read username / password from tmp file
$file=fopen("$argv[1]", "r") or exit(1);
while(!feof($file)) {
  $line=rtrim(fgets($file));
  if     (empty($username) && empty($password)) { $username="
      $line"; }
  elseif (isset($username) && empty($password)) { $password="
      $line"; }
}
fclose($file);

// Check database for username/password combination.
// Return 0 for success, 1 for fail.
mysql_connect($HOST,$USER,"$PASS");
@mysql_select_db($DATABASE) or die('Unable to select database: '
    . mysql_error());
$query=sprintf("SELECT 0=COUNT(*) AS result FROM $TABLE WHERE
    username='%s' AND password=SHA2('%s');",
        mysql_real_escape_string($username),
        mysql_real_escape_string($password));
$result=mysql_query($query);
mysql_close();
$bool=mysql_result($result,0,"result");
echo "$bool\n";
exit($bool);
?>
```