UCLouvain

epl

École polytechnique de Louvain

Development of a fuzzer on the RF layer for the ADS-B technology

Author: **Julien Delbeke**
Supervisor: **Ramin Sadre**
Readers: **Claude Oestges, Vasileios Friligkos**
Academic year 2020–2021
Master [120] in Cybersecurity

Abstract

Automatic Dependent Surveillance - Broadcast is a technology of air traffic control used around the world. Its purpose it to broadcast real-time information about the on-board systems of the aircraft and provide a geographical positioning to Air Traffic Controllers and other aircrafts in its vicinity. The intent of this thesis is to develop a fuzzer on the RF layer dedicated to the ADS-B technology and its receiver devices. The process consists of a literal study about the technology to develop an encoder capable of broadcasting custom messages from a personal radio transmitter and providing the means to choose to values to be sent either specifically or randomly as well as the quantity of messages and their intervals in time. The results of multiple tests in this paper shows its ability to simulate broadcasted messages while choosing its content and intercepting them with an monitored radio receiver. The utility of this tool is to provide the means to test both the hardware and software of a receiver by monitoring its behaviors and finding its limits.

Acknowledgment

I would like to sincerely thank my family and friends for their support and especially my loving girlfriend, Natalia for her constant motivation throughout this project.

I would also like to thank my academic supervisor, Mr. Ramin Sadre for his guidance and advises.

A special thank you goes to EUROCONTROL and its team for my past internship and their help with resources for this paper.

Last by not least, I would like to thank Université Libre de Bruxelles (ULB), Université Catholique de Louvain (UCL), Université de Namur (UNamur), Royal Military Academy (RMA) and Haute-école Bruxelles-Brabant (HE2B) for organizing this master degree.

Glossary

- ADS-B : Automatic Dependent Surveillance - Broadcast
- EUROCONTROL : European Organisation for the Safety of Air Navigation
- FAA : Federal Aviation Administration (USA)
- Independent vs dependent surveillance : Refers to whether the surveillance position is determined by the surveillance system itself (independent) or reported by the aircraft (dependent).
- PSR : Primary Surveillance Radar
- SSR : Secondary Surveillance Radar
- SESAR : Single European Sky ATM Research
- GNSS : Global Navigation Satellite System
- PNT : Positioning Navigation Timing
- ELS : Elementary Surveillance
- EHS : Enhanced Surveillance
- SDR : Software-Defined Radios
- SNR : Signal to Noise Ratio
- PPM : Pulse Position Modulation
- ACAS : Airborne Collision Avoidance System
- ICAO : International Civil Aviation Organization

- UAT : Universal Access Transceiver
- CRC: Cyclic Redundancy Check
- CPR : Compact Position Reporting

Contents

Chapter 1

Introduction

For decades, the air traffic surveillance has been relying on radar technology dating back to the 1950's. Those ground radars face limitations with a slow update rate, ranging between 5 to 12 seconds, and are challenging to install in difficult terrains such as mountains. The Automatic Dependent Surveillance - Broadcast is a technology part of the transition to the next generation of surveillance. ADS-B allows for a faster update rate around 1 second and easier installation which brings better ground coverage in geographical regions not attainable before. The aircraft is broadcasting real-time data with satellite positioning allowing for more precision. This level of accuracy and reliability helps air traffic controllers (ATC) to reduce the minimum separation between aircrafts which enhances safety and efficiency by providing better cost effective routes. In addition, nearby aircrafts can directly receive and display information broadcasted by other aircrafts.
The way this technology has been developed as a clear text communication has enable radio amateurs to cheaply build ADS-B receivers and observe air traffic a few hundred kilometers away from their position. Semi commercial websites have been created to gather all the publicly available data to provide a complete live view of the traffic all over the world. The most popular one being FlightRadar24 with over 20 thousands connected receivers, more than 2 million users and an average of 180 thousands flights tracked per day [FlightRadar24, 2021].
The long time goal being the wide adoption of this communication mean, every flying aircrafts will be equipped with ADS-B capabilities. It is therefore useful to test the working state of a device receiving and decoding those messages. Furthermore, as the messages broadcasted are created based on real-time information from the on-board system of the aircrafts, it would be necessary to be able to recreate errors in a controlled environment that may have occurred and observe the outputs and behaviors of a receiver hardware and software.

1.1 Background and objectives

The objective of this project is develop a custom made fuzzer that can be able to evaluate the reliability of any device capable of receiving and decoding 1090MHz Extended Squitter messages with format ADS-B.
Such a tool would need to fully customize an ADS-B message with chosen values for each field or randomized inputs in order to test a wider range on possibilities. It would also need the ability to choose the number of messages broadcasted and the intervals between them to find the limits of the receiver. Lastly, the versatility and flexibility must be taken into consideration to facilitate its use by the user.
After an extensive research through scientific papers and projects, it appears there does not exist a program publicly available able to test ADS-B receivers hardware and software. Few projects exist on shared platforms such as GitHub but are limited and do not offer a complete range of formats and customization.
The subject of this paper was suggested by EUROCONTROL as part of their testing of communication means. It is expected that this project will be applied to professional equipment and further development carried on by their team.

1.2 Structure of the thesis

This paper is organized as a scientific research and development project. It starts with the current state of the art around the ADS-B communication technology. This is followed by a literature about the data link used for communications in the aeronautics industry, Mode-S. The three next chapters are the methodology, results and discussions about the developed software. Some snippets of code will be described here while the source code can be found on the shared platform GitHub at the following link : `https://github.com/Jucky9/ADSBFuzzer`.

Chapter 2

State of the art

2.1 Current air traffic communication

The common mean to create a situational picture of the airspace relies on the Primary Surveillance Radars (PSR) and the Secondary Surveillance Radars (SSR).

The first one is composed of rotating antennas that transmit radio waves which bounce back after reflecting on the aircraft. Those antennas most commonly rotates at 5 to 12 rpm which translates to the information being sent to the ATC every 5 to 12 seconds. This system limits lay on the maximum range which varies between 46km and 150km depending on the class of the radar and on the fact that reflected signal need more power and can lead to reflection coming from fixed objects. However, this can be mitigated using a moving target indication. Its advantages are that the radars does not rely on the on-board system of the aircraft to send information and can therefore detect aircrafts having failures to transmit or intruders [Skybrary, 2021a].

The SSR on the other hand uses a system based on request response where the interrogation is sent over the 1030MHz frequency and a replies on 1090MHz. The rotation speed of those secondary radars are approximately the same of for the previous ones while the maximum range varies between 250 and 400km. This system has different interrogation modes ranging from Mode A, C or S. Mode A being the identifying code only, Mode C enables the ATC to receive the aircraft altitude or flight level [Skybrary, 2021b] and lastly, Mode S permits data exchange with more information being sent and will be detailed in the following chapter.

These two types of radars are often used together to form what is called a integrated system complex (ISC) [Institute, 2019].

Figure 2.1: Secondary Surveillance Radar on top of a Primary Surveillance Radar [Azimut, 2021]

As of May 2021, PSR and SSR are still in use in the international aerospace but with the introduction of satelite navigation, the aircraft disposes of a much more precise positional knowledge than the ground radars can send to the ATC.

2.2 ADS-B versions

There currently exist 3 versions of ADS-B in use. The first version; called version 0, followed the regulations *RTCA DO-260 / ED-102* [ED-102, r 13] published by the European Commission later modified for version 1 and version 2 in the following regulations respectively *RTCA DO-260A* [DO-260A/ED-102A, l 10] and *ED-102A / RTCA DO-260B* [DO-260B, r 02].
Those changes mainly concern the quality and integrity of the information provided in the message. The first standardization required only one field with a value to indicate the level of accuracy of the position. Multiple data were added to the latest version to provide a better knowledge on the certainty of the positioning.

Among other changes, Event-Driven Messages were added to the recurrent messages being Target State and Status, Aircraft operational Message and Aircraft Status Message.

It is important to note that those 3 versions are interoperable regarding the

critical data transmission. This allows different regions to demand different requirements. Indeed, Europe and North America have established requirement for version 2 while Australian airspace requires a minimum of version 0 [SESAR, 2021a].

The next standard ED-102B/DO-260C, also called ADS-B version 3, has been approved in December 2020 for future development. According to SESAR, it will offer significant improvements in bandwidth and new data items [SESAR, 2021b].

2.3 ADS-B Out

The ICAO gives the following definition for *ADS-B Out* in its ADS-B Implementation and Operations Guidance Document : "An ADS-B system feature that enables the frequent broadcast of accurate aircraft position and vector data together with other information." [ASIA and OFFICE, r 04]
In others words, ADS-B Out describes the ability of an aircraft to transmit data over a frequency to other surrounding aircrafts and ground stations. This data will be transmitted over 1090ES data link corresponding to the frequency 1090MHz or 978MHz also called 978 UAT. The latter being deployed mainly in North America and limits its uses but allows for meteorological data to be transmitted as well.

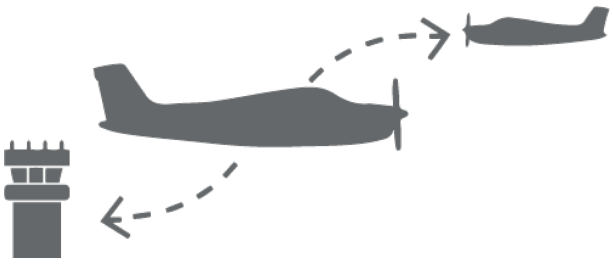

Figure 2.2: ADS-B Out
[Txtav, 2021]

2.4 ADS-B In

The same document as above gives the following definition regarding *ADS-B In* : "An ADS-B system feature that enables the display of real time ADS-B tracks on a situation display in the aircraft cockpit." [ASIA and OFFICE, r 04]
It is useful to add to this definition that not only other aircraft cockpits display those information. ATC and even amateur radios can receive and interpret those data.
Some smaller aircrafts can be equipped with ADS-B In without necessarily being

equipped with ADS-B Out. This can lead pilots to be overly confident in the output given by ADS-B In and to not take into consideration the presence of aircrafts without ADS-B Out capabilities.
Overall, it allows for a better situational awareness without relying on information received from the ATC alone.

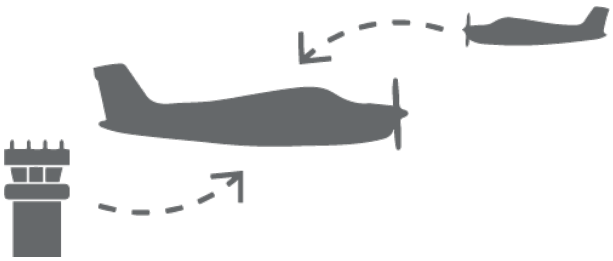

Figure 2.3: ADS-B In
[Txtav, 2021]

2.5 GNSS

The on-board ADS-B system relies on the data provide by the Global Navigation Satellite System (GNSS) that describes a constellation of satellites orbiting the earth. They provide positioning, navigation and timing (PNT) to a high level of precision. The positioning can be accurate to only a few meters [UNOOSA, 2021].

2.6 Around the world

In Europe, the European Commission mandated the Single European Sky ATM Research (SESAR) to develop an implementation plan for ADS-B across the continent. They decided that as of January 8th 2016, all new aircraft with a maximum takeoff weight of 5,700kg or more or a maximum cruise speed of 250 knots were obligated to be equipped with ADS-B Out capabilities. As for existing aircrafts, the deadline to meet the requirements was set for June 7th 2020. However, it was reported in 2018 in the ADS-B Implementation Plan that this deadline would not likely be respected by a non-negligible portion of the affected EU-based Airspace Users. The transition phase was then extended December 7 2020 with exemptions for older plane up to 2023 [SESAR, 2021a].

According to the latest data provided by EUROCONTROL from May 2021, 87.9% of aircrafts and 88.4% of flights operating in their managed area are equipped

with ADS-B v2.

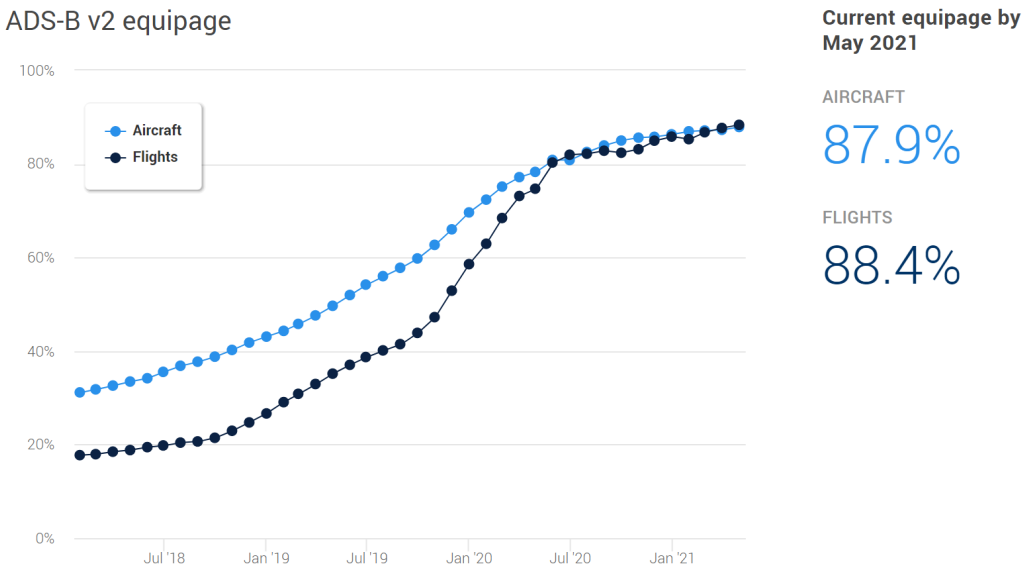

Figure 2.4: ADS-B v2 equipage
[EUROCONTROL, 2021]

In the United States, all aircrafts were obligated to be equipped by January 1st 2020.

2.7 Security

Since its implementation, ADS-B has been the subject of some security concerns as it is designed to broadcast real-time data of flights in clear text for anyone to intercept with a relatively simple and inexpensive setup such as the one that will be used in this project.

4 categories of attacks can be identified based on the scope they target. The first one being the confidentiality that is compromised with eavesdropping. Even though the technology is designed to be open without cryptography, real-time flight data can be considered as sensitive information and its collection may present a

security concern. The second is the integrity aspect of the communication that is threatened by message modification and message deletion. The modification of a message is called a Man-in-the-middle attack and include the interception of a message, modification of the content and forwarding the altered version to the original recipient. Next is the availability that can be affected by a jamming attack which is the transmission of a radio signal interfering in the communication and decreasing the Signal to Noise Ratio (SNR). This method is very effective and challenging to counter as one of the ways is to hop on different frequencies. The last one targets the lack of authentication in the ADS-B technology with message injections. The tool developed in this project could maliciously be used to perform this attack by broadcasting a created custom message on the main frequency of 1090MHz. It can be considered as a dangerous weakness and could disrupt pilots and cause risks of aircraft collision in extreme cases. However, such threats could be mitigated by ATC by cross referencing the data received with the data provided by PSR and SSR Mode A/C or identifying the origin of the message with the timing of reception and the claimed position of the aircraft. [Wu et al., 2020, H and S., 2020].

2.8 RF layer fuzzer

The concept of fuzzers is useful to automate the discovery of bugs, issues within the code and test input formatting constraints. The idea is to simulate malformed or semi malformed data inputs to a hardware or software and to monitor the results. It allows to try out a large number of scenarios too complicated or time consuming to do by hand. Different types of fuzzers exist. In the scope of this project, 3 types of fuzzers will be combined:

- Dumb fuzzer : this is the simplest one as it generates completely random inputs without concerns about the data format required. It is however inefficient and in some case useless if the size of the input is too big.
- Smart fuzzer : this implementation requires more time as the format has to be taking into consideration. This allows to set fixed values for certain field and random or within specific ranges for other fields. It is often more effective and is interesting for fuzzing a single field for example.
- Mutation-based fuzzer : the idea is to begin with a known valid input data that will be modified. This technique can provide a more accurate way of fuzzing specific fields but takes longer to implement.

The fuzzing on the radio frequency layer is more challenging than a regular fuzzer. Indeed, as Matt Knight explained in his lecture given at Black Hat, an RF

fuzzing project is protocol specific and it can be harder to evaluate what constitutes a crash or bugs. Furthermore, the environment is harder to recreate. It is important to take into consideration the whole testing setup as a layer 2 fuzzing implies a certain level of trust in the chipset used. "One can only see what one's radio tells you is happening" [Knight and Speers, 2018].
This fuzzer can be used to fuzz either hardware (antennas, SDR,..) or software decoding and interpreting the data. The fuzzer being implemented on the transmitter side, the monitoring of the receiver will need to be adjusted every time a new software or hardware is used. For the scope of this project, only one physical receiver and one decoding software will be tested and will serve as a proof of concept.

Chapter 3

Mode-S Structure

Mode-S is a Secondary Surveillance Radar technology integrated during the 1990s that allows for selective interrogation of the aircraft by the Air Traffic Controller (ATC). The standard Mode-S offers two different capabilities with elementary surveillance (ELS) and enhanced surveillance (EHS) that must be specified in the "capability" field. ELS equipped aircrafts must be able to transmit altitude, status, capability and identity. EHS equipped aircrafts must complies to the same requirements with additional downlink aircraft parameters such as selected altitude, speed, direction and roll and track angle. This technology mitigates the garbling present in SSR for Mode A/C and allows the ATC to interrogates multiple aircraft simultaneously [Schäfer et al., mber].

Furthermore, Mode-A and Mode-C allowed for only a limited number of unique identifier assigned to aircraft which is 4096. With the load of the air traffic increasing, Mode-S now provides a 24-bit field allocated for an unique identifier. A parity check field as well as a possibility for a more precised altitude (25-foot increment instead of 100-foot) were added.

ADS-B utilizes the 1090ES datalink and is structured as follows [Sun, y 30, Sun, 2021]:

Figure 3.1: 1090ES datalink

3.1 Preamble

The preamble is 8-microsecond long and composed of 4 pulse at intervals of 0µs, 1µs, 3.5µs and 4.5µs. It is fixed and can therefore be represented as a binary number such as "1010000101000000". The preamble serves as a synchronization to decode the data block following.

3.2 Downlink Format

The Downlink Format for Mode-S ADS-B messages is fixed and of value "17". It is represented by the first 5 bits of the Extended Squitter data block. Other downlink format values are used for other types of surveillance such as Mode-S request of altitude or identity as well as Airborne Collision Avoidance System (ACAS)

3.3 Capability

The following 3 bits of the data block are used to transmit the capability level of the transponder and its value can therefore vary between 0 and 7. As the Mode-S communication uses the interrogation technology, it allows the aircraft to inform the interrogator which data can be transmitted according to the capability of the aircrafts on-board system.

3.4 ICAO Address

This field is 24-bit long and is used to store the unique address given to each Mode-S transponder which allows the identification of an aircraft. The ICAO redistributes the 16 777 214 possibilities to local States of Registry that in turn assign those given addresses to their aircrafts. This address is often represented as a 6 characters long hexadecimal value. This unique address is also used to interrogates a specific aircraft by ATC.

3.5 ADS-B data block

Automatic Dependent Surveillance - Broadcast (ADS-B) is based on the on-board system of the aircraft that receives its geographical coordinates from another on-board system and will broadcast it periodically to every devices listening on a specific radio frequency. This system therefore provide ATC or other nearby aircrafts with a much more detailed package of information about the aircraft

situation compared to ground radars.
The technology is Automatic as it does not require input from anyone, Dependent as it still depends on the on-board system to provide the live aircraft data to be sent.
ADS-B uses the Mode-S Extended Squitter (1090MHz) datalink and requires Mode-S EHS capabilities in order to be implemented.

Type Code

The TC is the first part of the ADS-B data block with a length of 5 bits and determines to content of the rest of this block. Each TC is used to transmit different types of data following the table bellow:

Type Code	Description
1 - 4	Aircraft Identification
5 - 8	Surface Position
9 - 18	Airborne Position (with barometric altitude)
19	Airborne Velocities
20 - 22	Airborne Position (with GNSS altitude)
23 - 31	Reserved for other uses

Table 3.1: Type Codes correspondence

Those different types of ADS-B messages will be broadcasted by an airborne aircraft, according to Mode-S documentation written by EUROCONTROL, typically at the following rates:

- Airborne Position : 2 per second
- Airborne Velocity : 2 per second
- Aircraft Identification : 1 per 5 second

The standard states that you should not transmit more than 6.2 messages per second on average [Organization, 2014].

Aircraft Identification

The value of the Type Code ranging between 1 and 4 indicates an aircraft identification messages, also called ACID, that will be structured as follows:

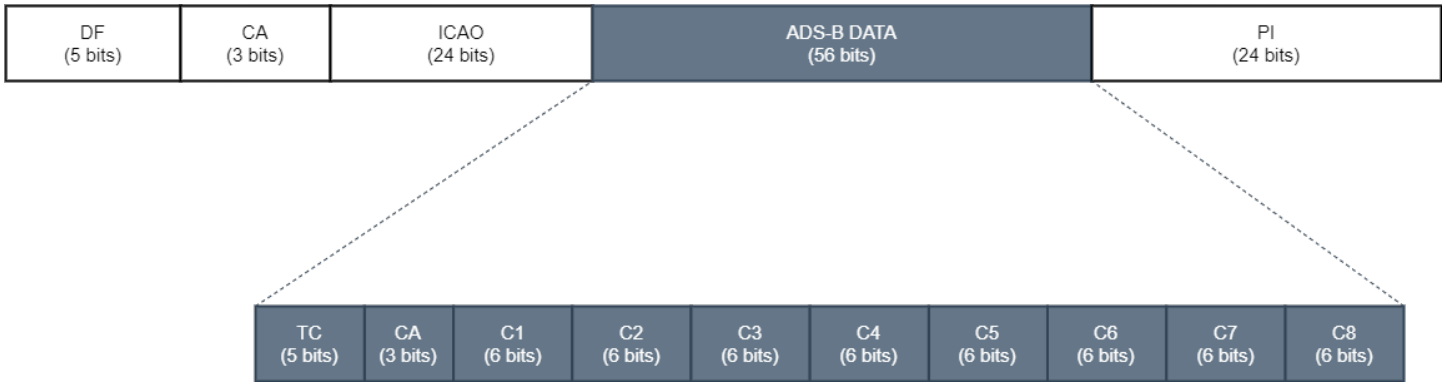

Figure 3.2: Aircraft identification message structure

Following the first 5 bits for the TC, the CA field long of 3 bits combined with the value of TC indicates the Aircraft Category to a relative precise level. The detailed table of Aircraft Category can be found in Appendix A.1. 8 different data fields of 6-bit long will compose the rest of the ADS-B block. Each field value is mapped to a character following this table:

Characters	Values
A - Z	1 - 26
" " (space)	32
0 - 9	38 - 57

Table 3.2: ACID mapping table

These values combined form the call-sign but is not unique to each aircraft and references the route used.

Surface Position

When the value of the TC is set between 5 and 8, the rest of the data fields are different to take into consideration that the aircraft is currently on the ground. As an example, in this case the altitude would not be a pertinent information to be sent. The ADS-B data block format will be as follows:

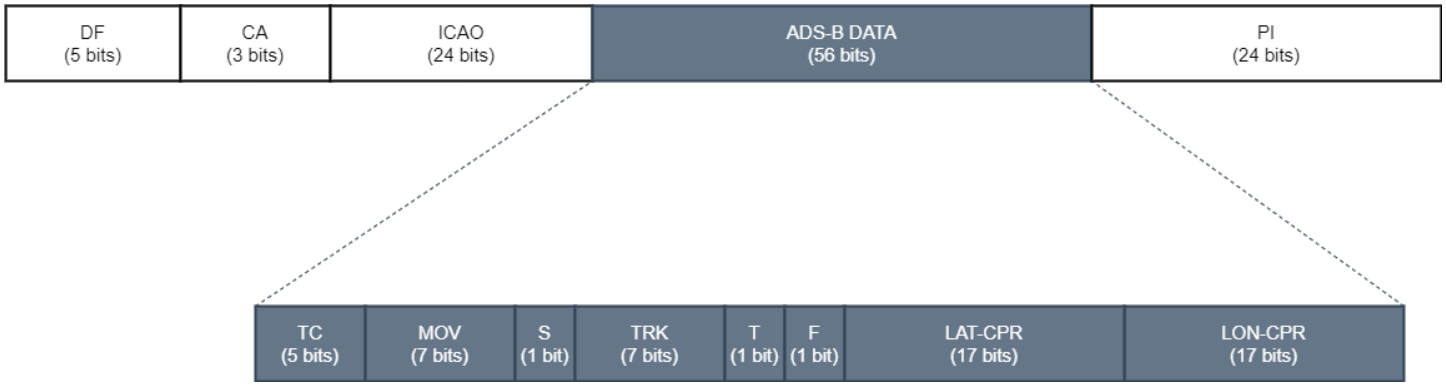

Figure 3.3: Surface position message structure

MOV field represent the ground speed of the aircraft but the value needs to be mapped to the table in Appendix A.2 and is non-linear to accommodate to lower and higher speed more precisely.

S stands for status and is composed of a single bit.

TRK field which is 7-bit long contains the ground track of the aircraft and indicates its heading angle in relation to the true north that is represented by a zero degree angle.

T is for Time and signals if the aircraft is synchronized to UTC time and is therefore a single bit long.

F, which is the same length as the previous field, indicates if the data frame is *even* or *odd*. Such specification is required for the translation of latitude and longitude. Indeed ADS-B uses the Compact Position Reporting (CPR) format to send coordinates. This format allows to encode latitude and longitude with 17-bit block each. This keeps the format of ADS-B to a 56-bit data block but a second frame is necessary to decode the exact coordinates. *0* indicates an even frame while *1* indicates an odd frame. The last two data blocks of the ADS-B payload, *LAT-CPR* and *LON-CPR* are therefore 17-bit long each.

Airborne Position

The following type of ADS-B message is the Airborne Position and start with a TC ranging from 9 to 18 if the altitude is given from the barometric devices and ranging from 20 to 22 if it comes from the on-board GNSS equipment. The rest of the data block follows this structure:

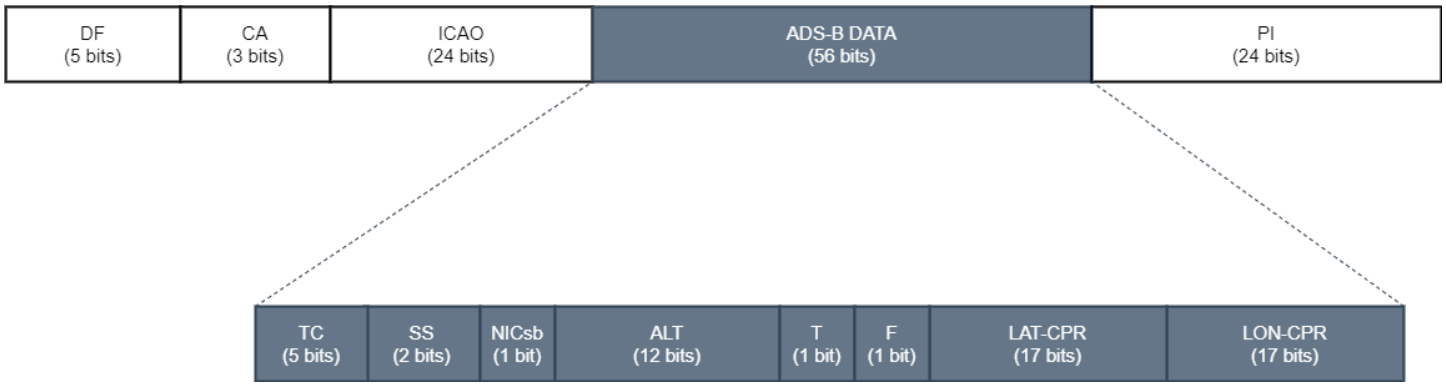

Figure 3.4: Airborne position message structure

SS stands for Surveillance Status with a 2-bit long value.

NICsb is a Navigation Integrity Category supplement bit.

ALT is a field of 12 bits allocated to encode the altitude. The altitude is encoded either by increments of 25 or 100 feet. This can be determined by the value of the *Q-bit* located at position 48 of the ADS-B message or position 8 of the altitude field. A value of 1 is for 25-feet increments while 0 is for 100-feet increments.

The last four parts of the ADS-B data block are the same as the Surface Position explained above with a Time and Frame flags followed by two 17-bit long blocks for the latitude and longitude in the CPR format.

Airborne Velocity

This type of ADS-B messages broadcast the airborne velocity of the aircraft and the corresponding TC is 19. The structure is slightly more complex as show here:

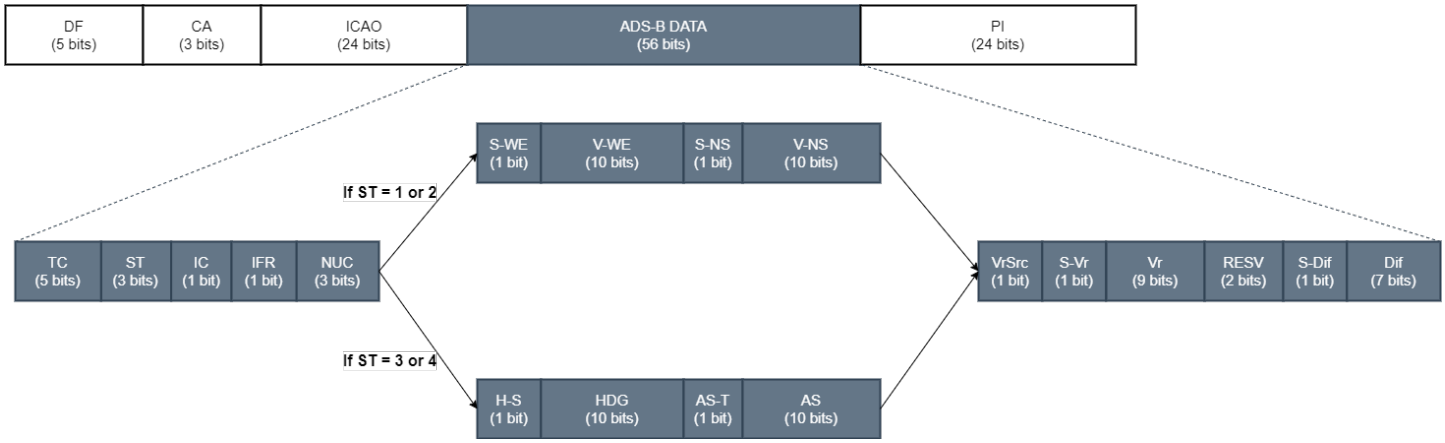

Figure 3.5: Airborne velocity message structure

It contains a larger number of fields than the previously cited message types.

The first five fields are the Type Code, Sub-type, Intent Change flag, Instrument Flight Rules flag and Navigation Uncertainty Category. The field **ST** designates the Sub Type of the airborne velocities type and is 3-bit long. The rest of the data block will vary depending on the value of this field. If the value is between 0 and 3, the 4 data blocks from bit 15 to bit 37 will be:

- S-EW : Direction sign (1 = East-West and 0 = West-East)
- V-EW : Velocity component
- S-NS : Direction sign (1 = North-South and 0 = South-North)
- V-NS : Velocity component

The variation between 0 and 3 reference a subsonic speed or supersonic speed respectively. If the value is between 4 and 7, the same 4 data blocks will be:

- H-s : Heading status (available or not)
- HDG : Magnetic heading
- AS-T : Airspeed Type (Indicated airspeed or true airspeed)
- AS : Airspeed

These information will be transmitted instead of the ground speed above when they are not available for various possible reasons.
The last 6 fields are respectively the Vertical rate Source which is either from the

GNSS or barometer, Sign flag for Vertical rate either up or down, the Vertical rate value, a reserved field, Sign flag indicating if the GNSS altitude is above or below the barometric altitude and the Difference value between the two.

Chapter 4

Methodology

4.1 Overview

The goal of this thesis being to develop a tool allowing to test a software or hardware listening to ADS-B communications, it was decided to use two computers with one transmitting and the other one receiving. The complete setup is a computer running Windows 10 with a virtual machine running Linux Ubuntu 20.10 on VMware Workstation to which is attached a transmitter device HackRF One and a Raspberry Pi 4B running Ubuntu 20.10 with the receiver device RTL-SDR connected to it. The first computer is running the developed code feeding the transmitter with chosen ADS-B messages to be broadcasted over a specific radio frequency to the receiver. This receiver will use a dedicated software and will be monitored to collect the outputs. The experiment layout can be seen in Figure 4.1 and a picture of the hardware in Figure 4.2:

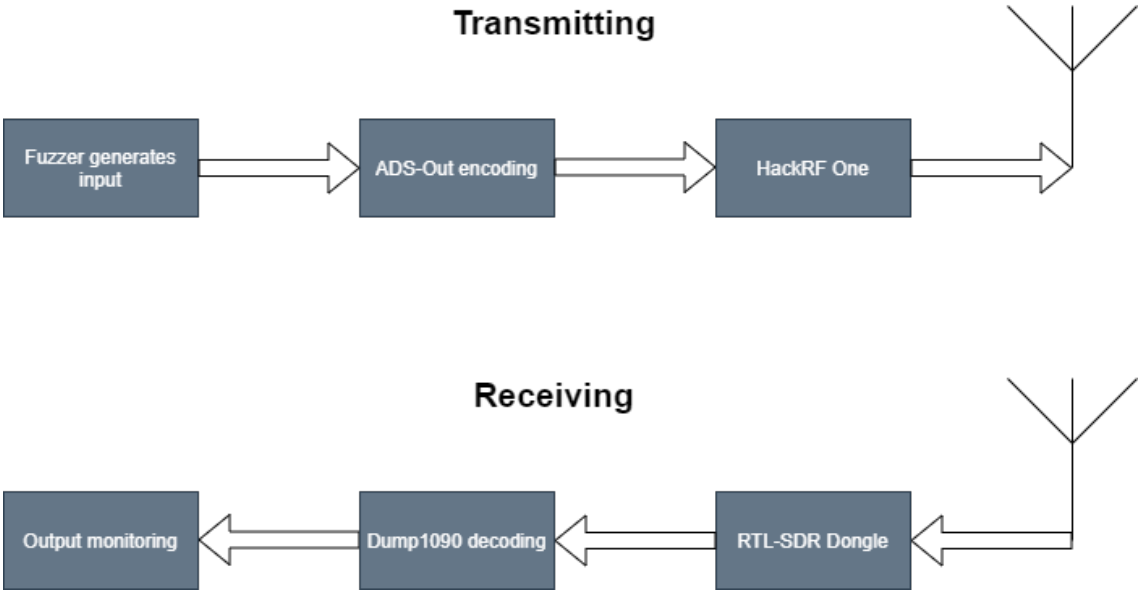

Figure 4.1: Experiment setup diagram

Figure 4.2: Experiment setup hardware

4.2 Transmitter

The transmitter used is a HackRF One developed by Michael Ossmann, founder of Great Scott Gadgets [M. Ossman, 2021] and generously landed by EUROCONTROL in the scope of this project. The device allows to receive and transmit over a wide range of frequencies spanning from 1MHz all the way to 6GHz and supports a sample rate from 2Msps to 20Msps (mega samples per second).
As stated previously, ADS-B is using the frequency of 1090MHz which is within the available range. However, as it is forbidden for anyone without proper authorization to emit on this frequency for security reasons, a lower and more common frequency will be used for this experience such as 433MHz. This frequency is often used by low power devices (LPD) such as car key fobs or amateur walkies-talkies.

Figure 4.3: HackRF One from Great Scott Gadgets

According to the official documentation of the device [Ossman, 2021a], at the frequency cited above, the HackRF One maximum TX power ranges between 5dBm and 15dBm. As this fuzzing experience takes place at close range, this output power is sufficient.
Regarding the RX power, the maximum recommended is -5dBm for this device.
The source code and documentation of the HackRF One is freely available on the official website or via their GitHub repository [Ossman, 2021b].

4.3 Receiver

The receiver used is a USB dongle with an standard antenna attached to it and branded by Nooelec [Nooelec, 2021]. They were originally produced to receive DVB-T TV (Digital Video Broadcasting Television) a few years ago and are today repurposed as software-defined radios (SDR).
It is a relatively cheap SDR that act as a plug and play device and can receive frequencies ranging from 25MHZ up to 1750MHz. The device was developed by Realtek and can be referred as RTL2832U after the name of the chip it is carrying. It has a maximum sample rate of 3.2Msps but may drop many frames at those numbers. The balanced rate is found at around 2.56Msps.

Figure 4.4: RTL-SDR from Nooelec

4.4 Fuzzer

The role of the fuzzer is to generate inputs to the transmitter as explained in details in Section 2.8. The tester using the fuzzer is still required to make choices regarding the type of fuzzer used or the specific fields that need testing. In this regard, a simple user interface was developed within the terminal. The generated inputs need to be encoded in order to be transmitted by the HackRF One. After making researches online for a free-to-use Mode-S ADS-B encoder, the open source project

ADS-B Out was selected [lyusupov, 2021]. This project can be redistributed and/or modified under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License.

ADS-B Out provides a starting point to the encoder but is not developed to incorporate the complete scope of the ADS-B technology neither serve as a fuzzer. Indeed, the program in its latest release can only encode one Type Code corresponding to the Airborne Position and does not allow for customization of all the fields. Following the basis of the code, an encoder for the other Type Codes as well as the features of the fuzzer were implemented. Other encoding functions were also modified to enhance performances. As the original project was developed using Python 2.7, the same programming language was used.

4.4.1 Encoder

A signal modulation needs to be performed on the data to be sent. For a Mode-S downlink signal, the type of modulation used is Pulse Position Modulation (PPM). The way it works for a binary value is that a "0" is represented with a 0.5µs flat signal followed by a 0.5µs pulse while "1" is represented the opposite way starting with a 0.5µs pulse followed by a 0.5µs flat signal [Sun, 2021]. It is similar to a manchester encoding scheme shown in Figure 4.5 where the data is located in the first half or the second half of the pulse.

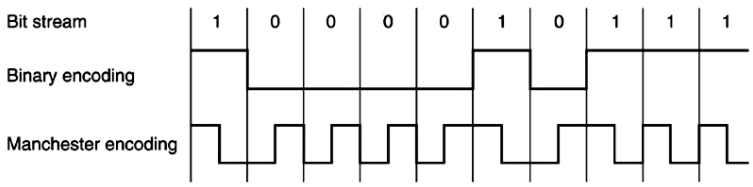

Figure 4.5: Manchester encoding

Below is the implemented function to perform the manchester encoding following the description given above.

```
1
2 def manchester_encode(byte):
3     """
4     Manchester encoding
5     Argument:
6         byte: byte to be encoded
7     Returns:
```

```
8           list of bits encoded
9       """
10      manchester_encoded = []
11
12      bin_list = list('{0:08b}'.format(byte))
13
14      for bit in bin_list:
15          if int(bit):
16              manchester_encoded.append(1)
17              manchester_encoded.append(0)
18          else:
19              manchester_encoded.append(0)
20              manchester_encoded.append(1)
21
22      return manchester_encoded
```

In digital telecommunication, algorithm for error detection during transmission are used. It allows to validate the data received and correct it if the scheme permits it or discard the message. There exist research papers on the implementation of error correction algorithms for ADS-B communications such as [Ma and Zhang, 2014] and [Ren et al., 2019] but the ICAO does not define any standard algorithm. In the case of ADS-B, the cyclic redundancy check (CRC) algorithm that uses binary polynomial arithmetic is implemented. Appended to the ADS-B signal is a 24-bit remainder computed based on the first 88 bits without the preamble and xored with a fixed generator, which consists of a binary modulo 2 division. The generator is a fixed 25-bit long value and of binary representation *1111111111110100000010010000000* and hexadecimal value *FFFA0480*.
As show in the Figure 4.6, 3 bytes of zeros are appended to the 88 bits before the computation. The receiver can verify the message at reception by xoring the complete message with the same generator. If the remainder of the computation is zeros, the message is validated. If not, the message contains errors.

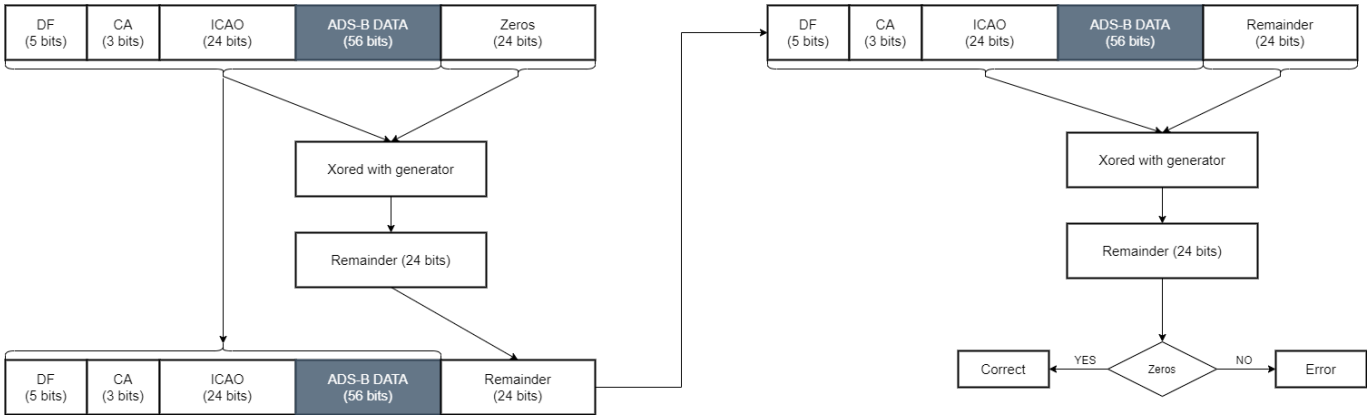

Figure 4.6: CRC scheme

Following is the python implementation of the algorithm:

```
def crc(data_bytes):
    """
    Cyclic Redundancy Check
    Compute remainder that allows to detect
    if an error occurred during transmission
    Argument:
        data_byte: 11 bytes message
    Returns:
        integer number representing the remainder
    """
    #Polynominal generator
    generator = "1111111111111010000001001"

    remainder_length=len(generator)-1

    bin_list=[]
    for byte in data_bytes:
      bin_list.extend(list('{0:08b}'.format(byte)))

    #Add zeros
    bin_list.extend(list('0' * remainder_length))

    for i in range(len(bin_list)-remainder_length):
        # Do modulo 2 multi if 1
        if bin_list[i] == '1':
            for j in range(len(generator)):
                # modulo 2 multi
```

```
28                      bin_list[i+j] = str((int(bin_list[i+j
                            ])+int(generator[j]))%2)
29
30      remainder = ''.join(bin_list[-24:])
31
32      return int(remainder,2)
```

The Compact Position Reporting (CPR) algorithm was taken from the project *ADS-Out* under the license of GNU General Public License 3.0. This algorithm allows to encode latitude and longitude with less bits then normally necessary. This is the standard defined for ADS-B encoding and is mandatory to fit the data into the payload. This is not a proprietary algorithm and is used in commercial ADS-B devices.
It would take 45 bits to encode a standard position while this technique can do it using 35 bits (17 for latitude, 17 for longitude and 1 for the odd or even frame flag). It can be decoded two different ways, either as a globally unambiguous position decoding where no position is originally known and both messages are required, or as a locally unambiguous position decoding where the decoder knows a previous reference position from older messages, therefore only one frame is needed.
This type of encoding works by dividing the world in multiple latitude zones and longitude zones on two different grids. The sender transmits two positions in relation to a specific zone for each respective grids. The receiver on the other side can use those two positions and find a corresponding match to determine the correct zone and the global position. After the initial position is decoded using a pair of even and odd frame received within 10 seconds, each frame following will be used to update the current position.

To generate the complete payload that will later be modulated, the following function was implemented. Each parameters can be modified to fulfil the goal of the fuzzer and obtain different output messages.

The diagram shown in Figure 4.7 describes the encoding of the type code for Airborne Velocity as the other ones use the same logic. Each field value being given either by the user or assigned randomly uses more bits than the necessary number required by the ADS-B standard. Therefore, each value needs to be bit shifted, masked and concatenated into bytes to build the payload to respect the format.
Each variable is shifted according to the number of bits it should contain in the documentation on the format of an ADS-B message. At the end, the CRC is calculated based on the 11 bytes above and appended to the payload. The function outputs a list of 14 bytes (112 bits) that is the formed ADS-B message.

A snippet of this function can also be found in the Appendix A.4.

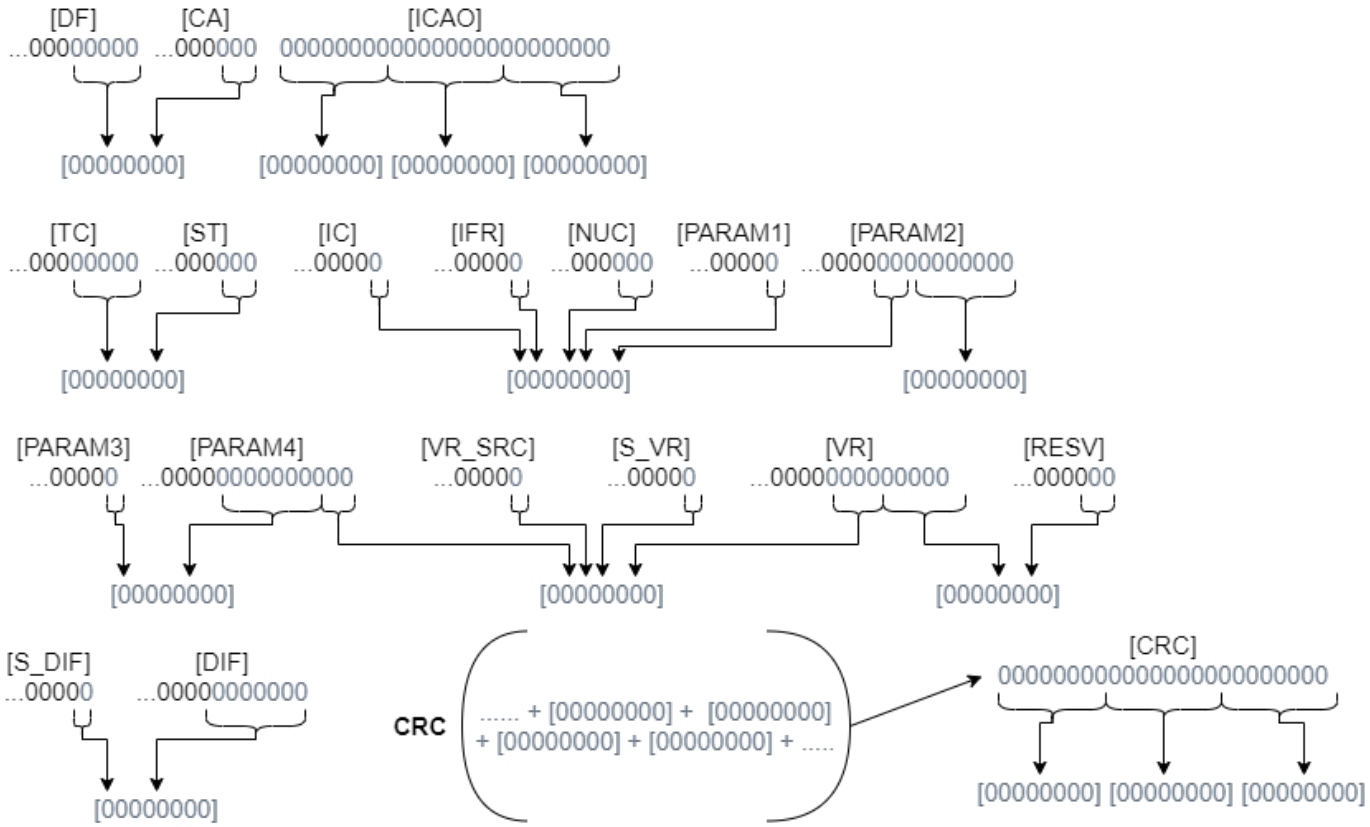

Figure 4.7: Airborne velocities encoding diagram

In radio frequency applications, quadrature signals, also called IQ signals, are used. In-Phase and Quadrature components defines two periodic signals differing in phase by 90 degrees. The HackRF One deals with an 8-bit unsigned representation for the IQ components. The program therefore creates an *.iq8s* file where the data is written. As each individual bit of data from the original message is represented by two bits (pulse and flat) after the manchester encoding and each bit is encoded as a signal with an 8-bit signed number for I and Q, the overall payload of 120 bits including the preamble is represented by 3840 bits. It is required by the device to align this input to a 256KB buffer. This is done using the following command:

```
$ dd if=Samples.iq8s of=Samples_256K.iq8s bs=512 seek=511
```

Where *bs* is the block size in bytes and *seek* is the number of block to skip before writing. The message being of length 3840 bits or 480 bytes, it will be padded with

32 bytes of zeros to obtain a block size of 512 bytes. A single file can therefore contain 512 blocks of ADS-B messages.

Using the official debian package available, *hackrf_transfer*, the previously created file can be interpreted and transmitted by the HackRF One. The documentation of the package can be found on its official page on the *debian.org* website [hac, 2021].
This package can be used with the command line the following way:

```
$ hackrf_transfer -t Samples_256K.iq8s -f 433000000 -s 2000000 -x 10
```

-*t* option is the input file to be transmitted.
-*f* option is the desired frequency in Hz.
-*s* option is the sampling rate in Hz.
-*x* option is the gain in dB.

The fuzzer developed can therefore transmit all four types of an ADS-B message with DF 17 over a chosen frequency through the transmitter device HackRF One.

The purpose of the program being of testing only, no strict input verification was implemented.

In order to facilitate the use of the program, it was decided to implement a argument-based execution combined with a configuration file. The configuration file can be used to set a value for each inputs and the user needs to indicate the path of this file at execution time. It is useful to recreate and save scenarios. The argument-based execution can be used in conjunction by adding any argument at execution time. The value of the argument referenced will prevail over the one found in the configuration file. This is done with the libraries *ArgParse* and *ConfigParse* of Python [arg, 2021, con, 2021]. To reference the path of the configuration file, the user can use the argument *-cfg* or *- -configuration*.
The user will also have the option to randomize any field by using the value *"?"*. The random value will be selected based on the length in bits of the specific field. For most of the fields, if no value was given either in the configuration file nor in the arguments, they will be randomized. Exceptions are made for some field with a default value to allow an easier use of the tool such as the *Downlink Format (DF)* set to *17* by default as it is the value for 1090ES messages. Parameters such as the transmission frequency, sample rate, output file or frame (even or odd) for geographical position related messages are set with a default value that can be modified.

One of the mandatory argument is the type of message the user wishes to transmit. As explained previously, the user can choose between Aircraft Identification,

Surface Position, Airborne Position and Airborne Velocity. This is done through the parameters *-t* or - *-type* with the possibles choices being respectively : *acid, sp, ap* and *av*. This value could also be specified in the configuration file.

When fuzzing a hardware or software, the time constant plays an important role. Therefore, the tool allows to chose the number of messages to be broadcasted and the interval at which they will be broadcasted. The user can choose the interval in seconds with the parameters *-i or - -interval* and the amount with *-nb or - -number*. If no value is given, it will default to 1 message and interval set to 0. When the number of messages is greater than one and the interval value is zero, the tool will transmit one message per file at a time. However, if the value is negative (i.e. *-1*) the messages will be broadcasted in a single file as a batch. As said previously, with the message size being of 512 bytes and the maximum file size of 256KB, the maximum number of messages to transmit as a batch in 512. The rest will be sent separately.

As an additional feature, the tool will keep a log of every messages broadcasted. It will write them in hexadecimal format as this is the most commonly way to display raw data in receiver softwares. A timestamp and the message type will also be logged each time to allow for an easier monitoring and comparison of the results obtained between the transmitter and the receiver. The parameter *-log* or - *-logfile* are used to indicate the path of the log file while the default value is *ADSBFuzzer.log*.

Chapter 5

Results

In order to produce a proof of concept, the working state of the multiple features of the program will be demonstrated in this chapter.
Firstly, a test will be conducted on the receiver alone tuned to the frequency 1090MHz listening for real passing aircrafts. This will prove the good working order of the device and the decoder. We will follow by capturing a valid message from the previous step, re-encode it using this fuzzer and broadcast it over a different frequency with the transmitter. This will prove the working state of the transmitter as well as the encoder. Afterwards, custom ADS-B messages of a given types will be broadcasted from the transmitter to the receiver to prove the working state of the customization feature of the software. Lastly, different amount of messages with different intervals will be broadcasted and a comparison will be performed between the log and the monitored receiver.

5.1 Receiving over 1090MHz

To prove the good working state of the hardware and software of the receiver, we will intercept a communication from a passing aircraft and compare it with publicly available live data. On May 10th 2021 at 15:50:28, the following ADS-B communication was intercepted with the RTL-SDR plugged into a Raspberry Pi 4B running Ubuntu 20.10 with the software *dump1090-fa* on frequency 1090MHz. The following command was used :

```
$ dump1090-fa --freq 1090000000
```

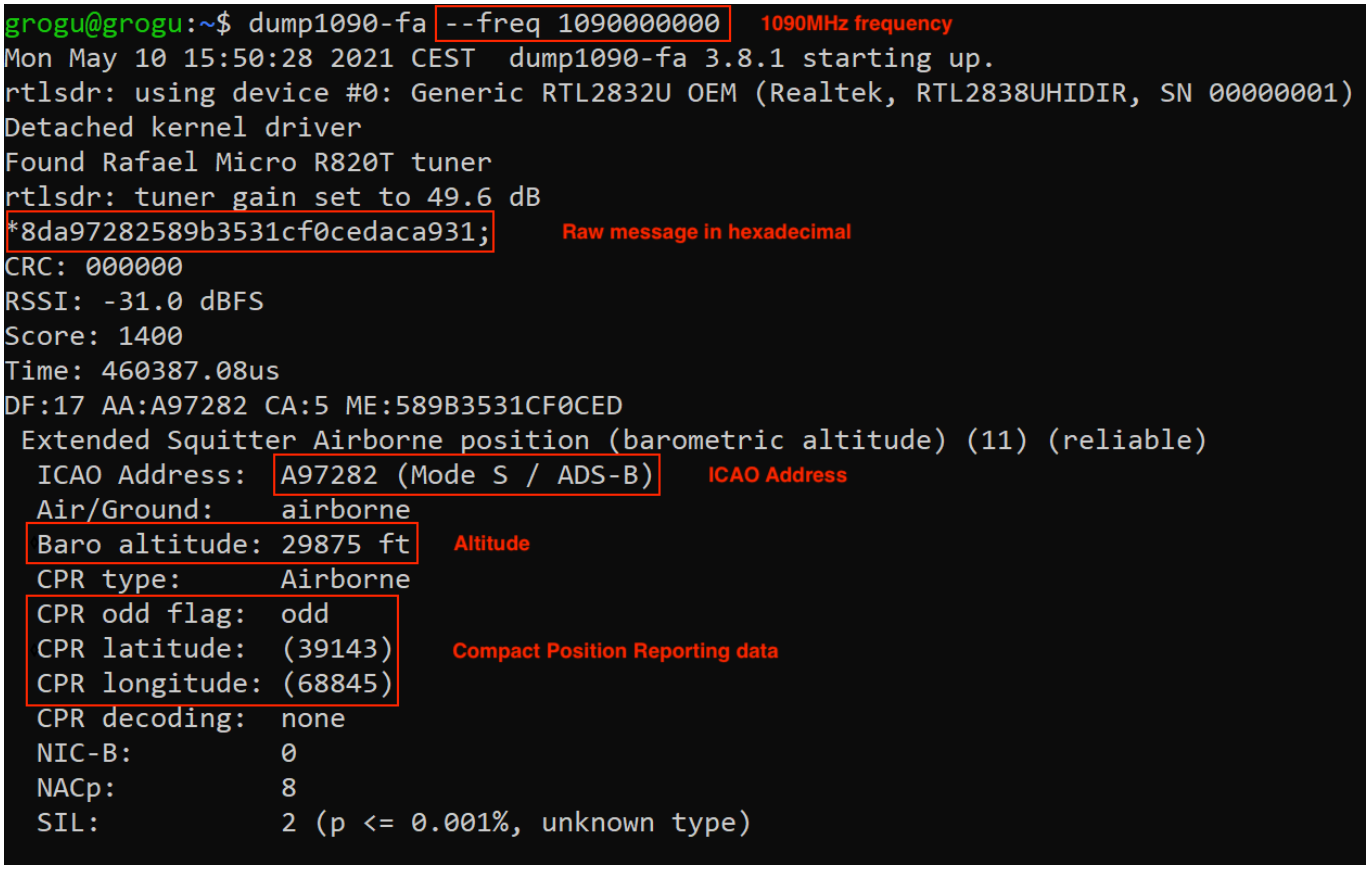

Figure 5.1: Real airborne position message captured by receiver (ICAO: A97282)

The message received is shown in raw hexadecimal format on the top of Figure 5.1. After parsing of the message by the receiver software, it appears it is of type Airborne Position with barometric altitude with the aircraft ICAO address being *A97282*.

The *dump1090-fa* [FlightAware, 2021] software provides also a GUI feature with a map. This interface is accessible through the port 8080 of the Raspberry Pi and a screenshot is shown in Figure 5.2. The position of the aircraft with ICAO address *A97282* can be see in the center of the map and the flight data on the right pane.

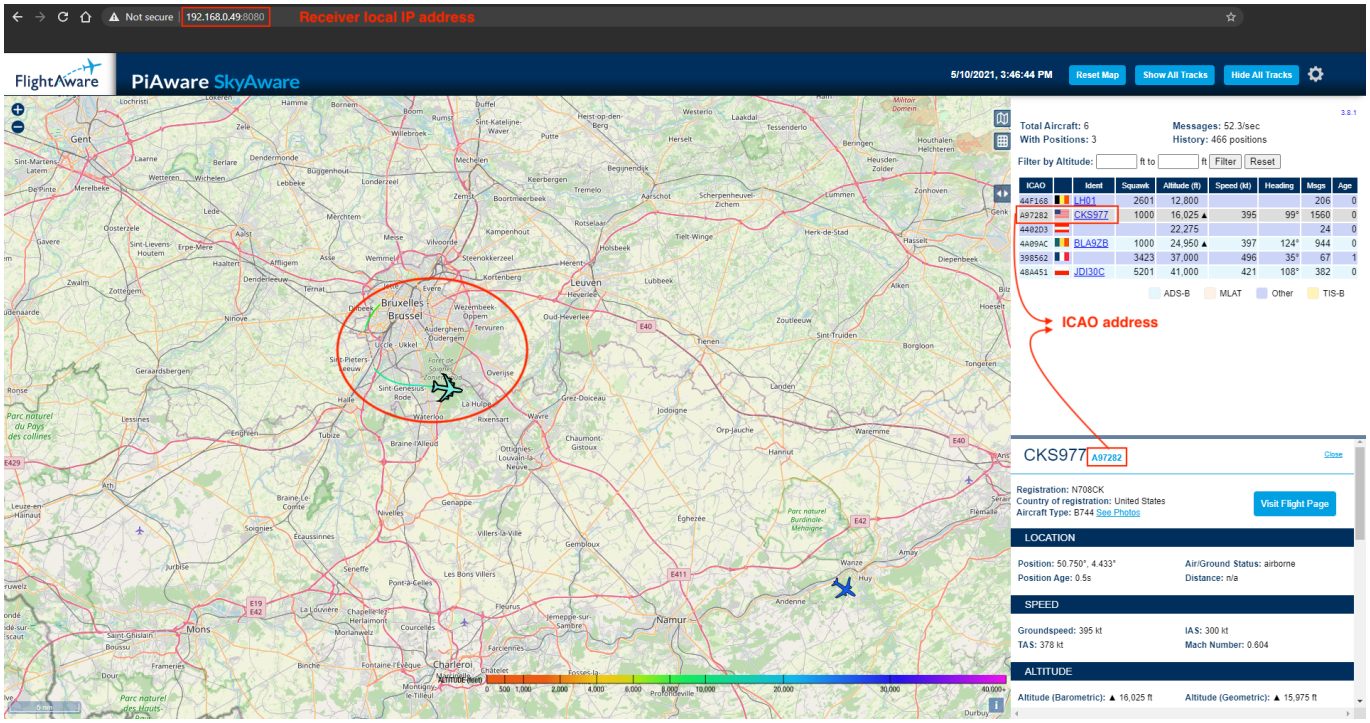

Figure 5.2: Real airborne position message captured by receiver shown on the map (ICAO: A97282)

The screenshots of Figure 5.1 and Figure 5.2 were taken few minutes apart, hence the difference in values such as the altitude.

Within the same command execution, another message of type Aircraft Identification was also captured. In the screenshot shown in Figure 5.3 can be observed the Type Code 4 indicating an ACID message, the ICAO address *A97282* similar as above, the identification value *CKS977* and also the CRC result being zeros which indicate no transmission errors.

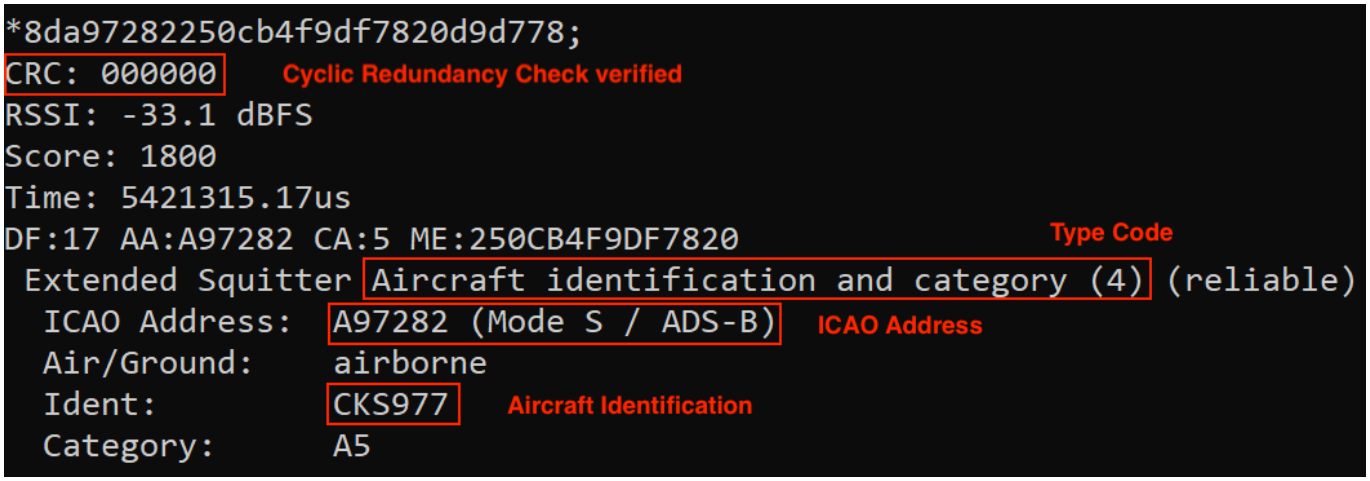

Figure 5.3: Real aircraft identification message captured by receiver (ICAO: A97282)

We can therefore verify this validity of the information by using the popular aircraft tracking website *FlightRadar24* which gather every aircraft available live data. Here is a screenshot around the same time as the data above was captured:

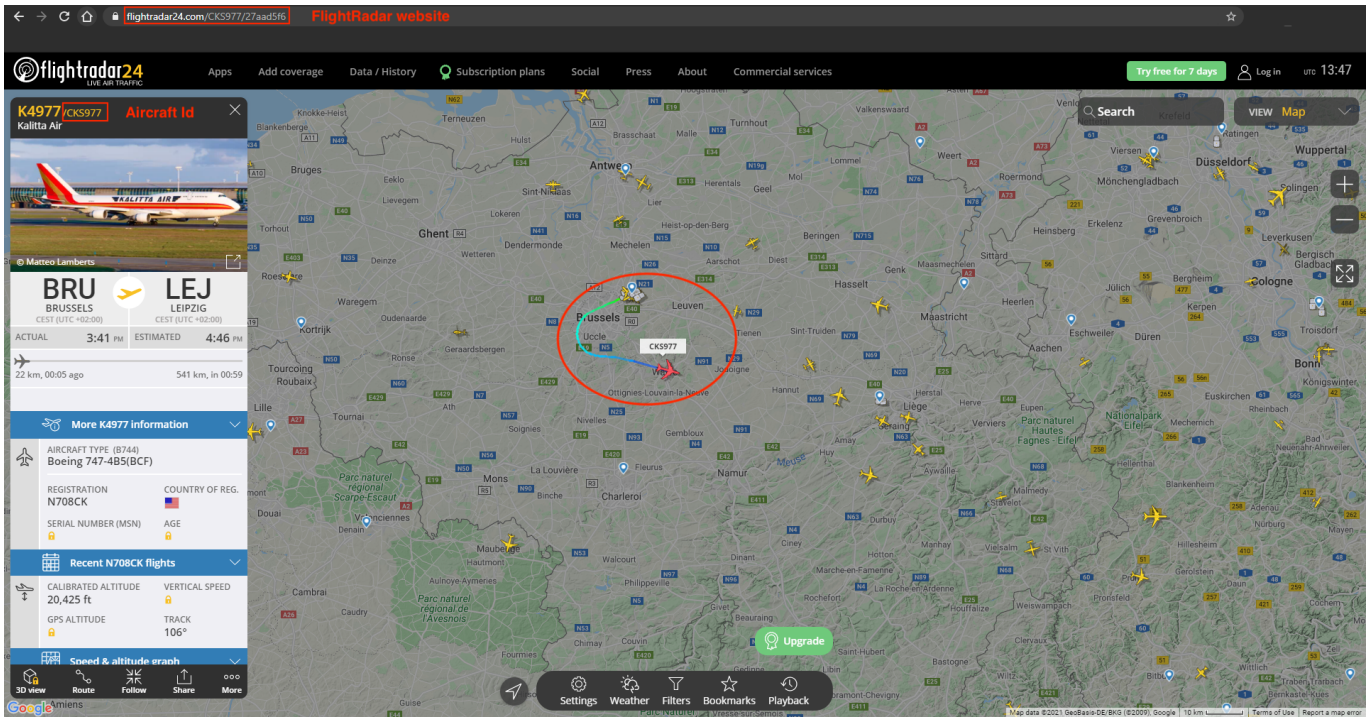

Figure 5.4: Real aircraft on FlightRadar24 (ICAO: A97282)

The ICAO Address of the aircraft can also be used as verification and is present in the data on *FlightRadar*.

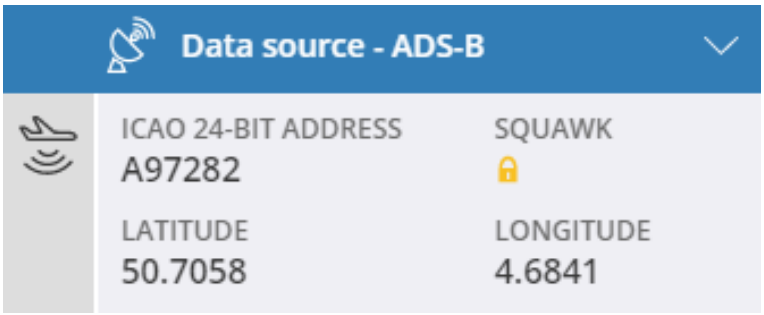

Figure 5.5: Real aircraft on FlightRadar24 - ADS-B data (ICAO: A97282)

5.2 Transmitting valid messages

Now that the receiver has been tested to capture valid messages, the transmitter will be used to recreate those same messages. This is to prove the ability of the program to generate custom known valid messages and broadcast them. For this test, the messages captured is the section 5.1 will be used.

The first one will be the Aircraft Identification message. The data to be sent are the following:

- Downlink Format: 17
- Capability : 5
- ICAO address: A97282
- TC : 4
- EC : 5
- Characters to be mapped : 3, 11, 19, 57, 55, 56, 32, 32 (CKS977)
- Parity information : automatically generated

To do so, this command is executed gathering all information as argument parameters :

```
$ python2 ADSB_Fuzzer.py -t acid --downlinkformat 17 -ca 5 --icao A97282
--typecode 4 -ec 5 -c1 3 -c2 11 -c3 19 -c4 57 -c5 55 -c6 55 -c7 32 -c8 32
```

It was then broadcasted over the frequency 433MHz for the legal reason mentioned previously. As no real aircraft emits a those ranges, we can be sure that the received data is ours. The captured data in Figure 5.9 shown the receiver listening on said frequency hours apart from the real transmission.

```
grogu@grogu:~$ dump1090-fa --freq 433000000   Chosen frequency
Mon May 10 17:19:14 2021 CEST  dump1090-fa 3.8.1 starting up.
rtlsdr: using device #0: Generic RTL2832U OEM (Realtek, RTL2838UHIDIR, SN 00000001)
Detached kernel driver
Found Rafael Micro R820T tuner
rtlsdr: tuner gain set to 49.6 dB
*8da97282250cb4f9df7820d9d778;   Same raw data
CRC: 000000
RSSI: -3.7 dBFS
Score: 1400
Time: 4355110.83us
DF:17 AA:A97282 CA:5 ME:250CB4F9DF7820
 Extended Squitter Aircraft identification and category (4) (reliable)
  ICAO Address:  A97282 (Mode S / ADS-B)
  Air/Ground:    airborne
  Ident:         CKS977
  Category:      A5
```

Figure 5.6: Recreation of valid message (ICAO: A97282)

The same test was made for the Airborne Position message. Here are the values for each field :

- Downlink Format : 17
- Capability : 5
- ICAO address : A97282
- TC : 11
- Latitude : 50.63
- Longitude : 05.12
- Altitude : 29875
- Nicsb : 0
- Ss : 0

- Time : 0

Those data translates in the command:

```
$ python2 ADSB_Fuzzer.py -t ap -df 17 --capability 5 --icao A97282
--typecode 11 -lat 50.63 -lon 05.12 --altitude 29875 -nicsb 0 -ss 0 -time 0
```

The broadcast was received on the frequency 433MHz as shown in Figure 5.7. The position on the aircraft is also similar between Figure 5.2 and 5.8 which confirm the successful recreation of the original captured transmissions.

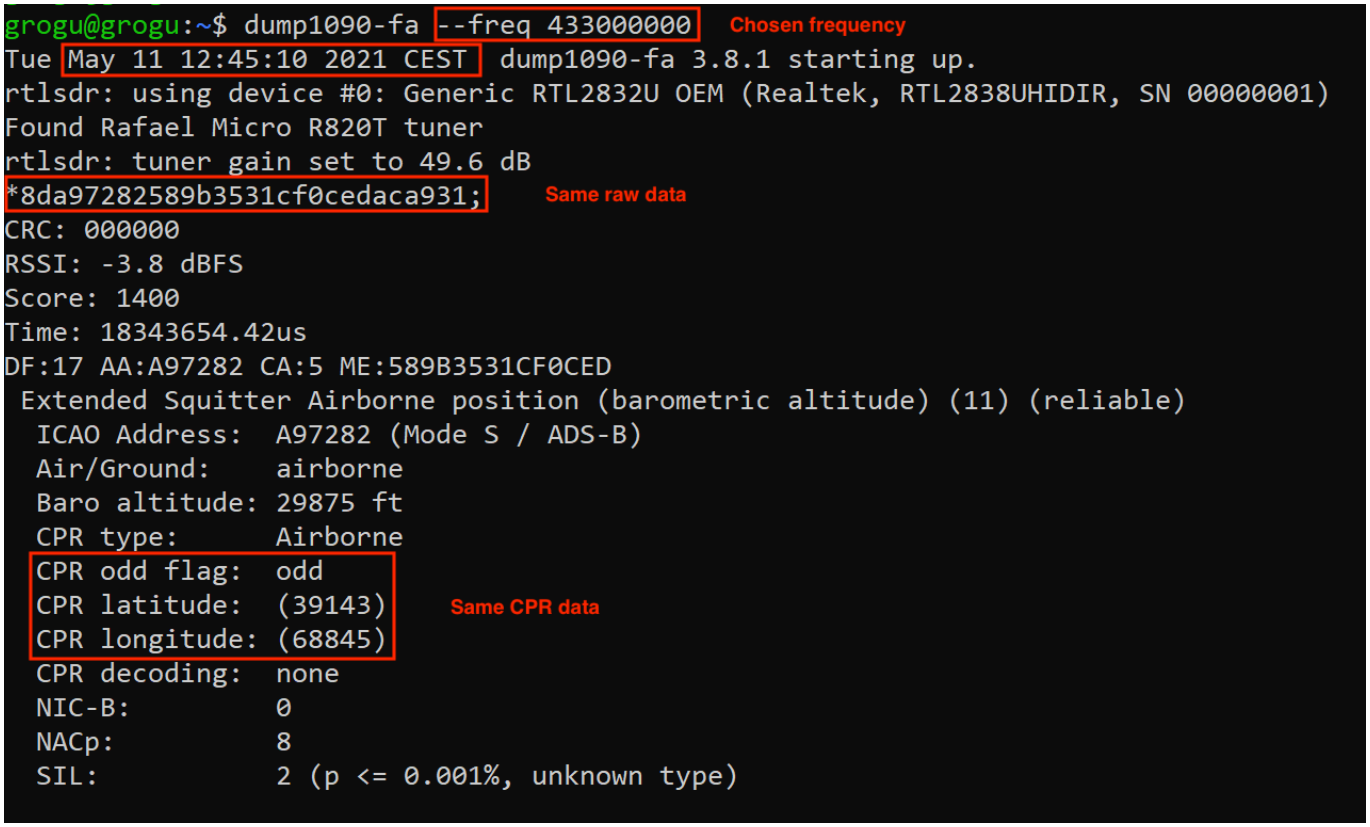

Figure 5.7: Recreation of a valid Airborne Position (ICAO: A97282)

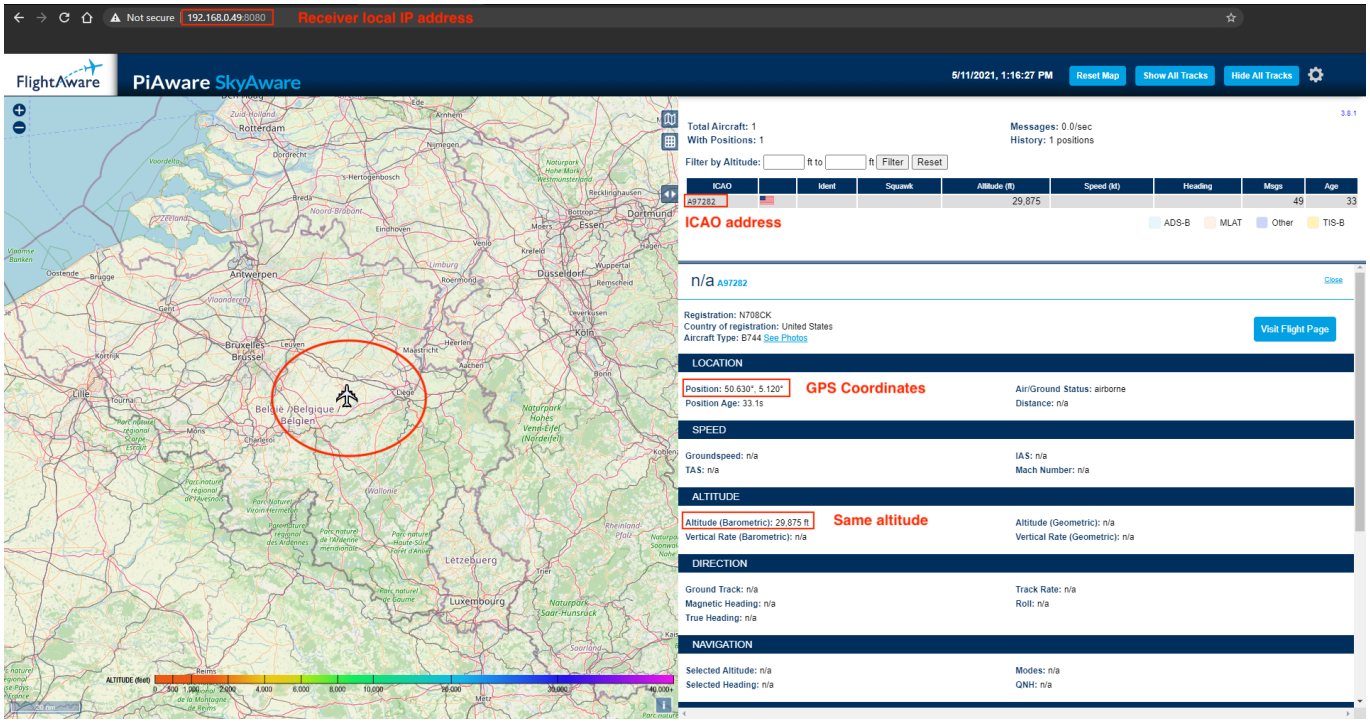

Figure 5.8: Recreation of valid Airborne Position shown on map (ICAO: A97282)

5.3 Transmitting custom messages

As part of the project, the fuzzer should be able to perform a mutation based fuzzing. The Airborne Position message will be used to prove the working state of this feature.
The longitude of the aircraft will be altered and the modified position on the map will be shown. The original value is *05.12 degrees* and the altered value is *00.83 degrees*.
Here is the command to do so:

```
$ python2 ADSB_Fuzzer.py -df 17 --capability 5 --icao A97282 --typecode 11
--latitude 50.63 --longitude 00.83 --altitude 29875 -nicsb 0 -ss 0 -time 0
```

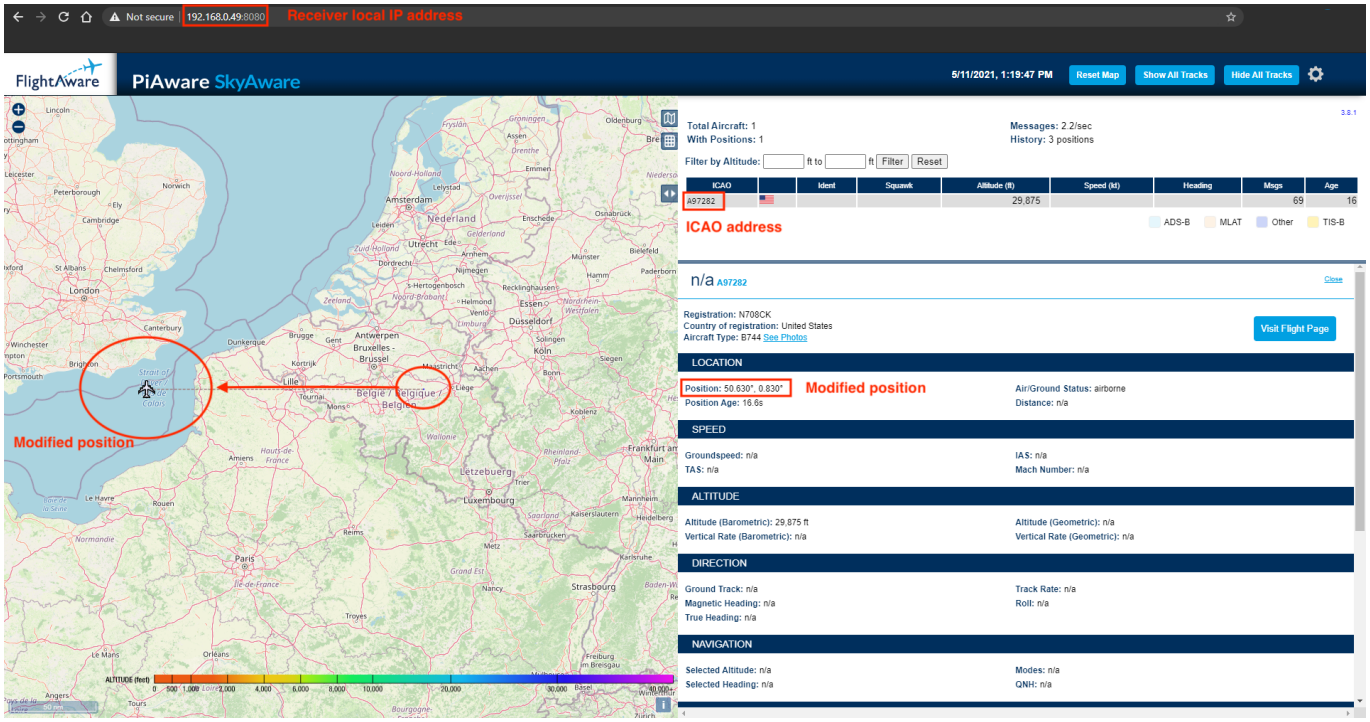

Figure 5.9: Modification of valid Airborne Position shown on map (ICAO: A97282)

This example also shows the security concern related to the possibility to broadcast spoofed position messages that will be interpreted by aircraft.

5.4 Transmitting randomized messages

In order to create a mutation based fuzzer as well as a dumb fuzzer, the option the randomize any field was necessary.
For this example, the Airborne Velocity type code was selected. A few field will be randomized such as the ICAO address, the directions and the velocity. This time, the configuration file was used to set the values of the fields. As mentioned, the question mark (*?*) indicates that the value of the field will be random.
Here is a look at the configuration file for this execution:

```
downlinkformat=17
capability=3
icao=?
typecode=19

[AIRVEL] #Airborne Velocities Part 1
```

```
subtype=1
ic=0
resva=1
nac=0

[AIRVEL2A] #Airborne Velocities Part 2A
swe=?
vwe=?
sns=?
vns=?

[AIRVEL3] #Airborne Velocities Part 3
vrsrc=0
svr=1
vr=14
resvb=0
sdif=0
dif=23
```

More fields could have been randomized but the probability of the message being discarded by the receiver for malformation would have been higher. On the Figures 5.10 we can observe the captured messages. In the examples, 5 messages were sent with the field *ICAO address, swe, vwe, sns* and *vns* randomized for each.

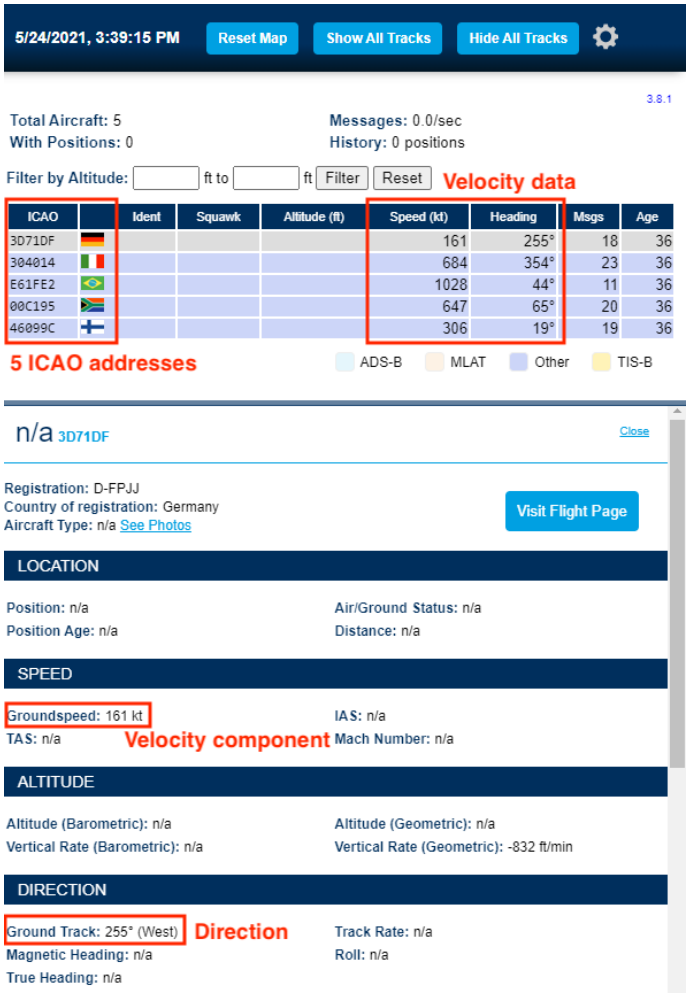

Figure 5.10: 5 Airborne Velocity messages with randomized fields

All the randomized fields specified earlier can be observe with different values. As the ICAO addresses were also chosen randomly, each messages was attributed to a different aircraft. Regarding the flags next to the addresses, it is mapped to the official ICAO allocation for each country. For example, Germany has an allocation of hexadecimal values from *3C0000* to *3FFFFF*. The rest of the table is in the Appendix A.1.

5.5 Large amount of messages

Another aspect of fuzzing involves the testing of the hardware or software to handle a large amount of data being sent.
For this last example, a high number of messages will be broadcasted from the

transmitter while the receiver will be monitored to later compare the amount of data received and lost during transmission.

The type of messages used will be Aircraft Identification messages and the constraints to a well formed ADS-B message will be respected to avoid messages being discarded by the software and not appearing later in the comparison. The exercise will serve as a proof of the working state of the software and its ability to broadcast a large amount at given intervals of ADS-B messages in order to test a receiver device.

The following command line was used:

```
$ python2 ADSBFuzzer.py -nb <number of messages> -i <interval> -t acid
```

In the following table we can observe the results of multiple tests with different given number of messages and intervals. Note that those results are specific to this receiver device and decoding software. The results may differ with more powerful devices.

Sent	Interval (second)	Received	Lost
1	None	1	0
64	None	64	0
512	None	475	37
1000	None	928	72
200	0	200	0
1000	0	1000	0
1000	0.5	1000	0

Table 5.1: Comparison between received and lost

The value *None* indicates that the messages are sent as a batch in a file at a time containing a maximum of 512 messages while the value zero indicates one message per transmission.

Chapter 6

Discussions

There were many challenges while building this tool. Indeed, the documentation regarding the details of the ADS-B technology is quite limited or very technical. Also, no previous fuzzer of such communication on the RF layer was found as a basis for the research. However, this absence of previous similar tool makes the development of this one interesting.

Some improvements could be made on the interface of the software such as creating a graphical user interface. However, in this project, the development was focused on the core of the ADS-B technology while keeping in mind the user friendliness.

6.1 Use cases

There can be many use cases derived from this tool. Those mentioned in this paper are the replication of an errors or behaviors which would have occurred in production and test the limits at which the receiver software or hardware loses data.
The ability to set the value of the parity information if needed allows the testing of error correction algorithm mentioned previously. In such tests, it is not easy to produce voluntary errors during transmission which is why the customization of the message can be of use.
Moreover, the variables of time intervals and repetitions are necessary to simulate a large number of different inputs which is a lengthy process to perform manually.
Performing a message injection is interesting to test the detection systems of fake ADS-B messages in ground stations or in aircrafts.

Considering the numerous specifications of the ADS-B technology, a high level of customization can be necessary. In the case of the CPR method of encoding, the user can choose only one of the two frames usually transmitted or test the altitude field with increments of either 25 or 100 feet.

6.2 Future development

More interesting results might occur with more powerful hardware as it is able to generate more inputs. Aviation professionals could use the tool for more specific testings related to their activities or following real life scenarios and reproductions of such behaviors.

In addition, the remaining type codes which are event-based could be added to the tool or other downlink formats of Mode-S communications. The code of the fuzzer was developed with maintainability in mind and scalability in order to facilitate the addition of more features.

6.3 Future of ADS-B

In early May 2021, EUROCONTROL shared a press release announcing they would integrate real-time air traffic surveillance data provided by Aireon's space-based ADS-B system. Aireon is responsible for the development of the next generation of airspace surveillance through satellites [EUROCONTROL, y 06]. Currently, only 70% of the globe surface is covered by surveillance technology [Aireon, June]. The systems used today are limited by the line of sight. The space-based ADS-B would create a continuous global coverage. The data received by each part of the world covering further than their region will improve the predictability of future traffic by 20% (Eurocontrol integrating space ADS-B). This improvement helps reduce the environmental impact as well as the delays of flights.

While this is expected to be the future of air traffic surveillance, it will first be use in conjunction with other surveillance technologies such as PSR and SSR, and provide a better situational awareness to aircrafts and ATC.

Chapter 7

Conclusion

Aircraft surveillance technology is a critical aspect of aerial security. The development of the ADS-B technology as a substantial part of the Next Generation of aircraft surveillance makes it important to assess the hardware and software operating in aircrafts or ground stations.

This project generated an ADS-B fuzzer tool with verification results that supports its operational feasibility and utility. It validated the tool against format requirements specified for ADS-B 1090ES. It was developed with the focus on the ease of use, maintainability and evolution to welcome other technologies and expand its use cases.

It was demonstrated is this paper that it is capable of broadcasting ADS-B 1090ES format messages on a chosen radio frequency depending on the capabilities of the transmitter it is attached to. The structure of a message is respected based on its type and each values of the corresponding fields can either be chosen manually or randomized. The time and repetition variables can also be selected.

This tool is therefore able to fuzz both the hardware and software of a receiver device and observe its limits and behaviors according to its inputs.

Although it was not the intent of this project, the results of the development shown in this paper indirectly highlights the security weaknesses of the ADS-B technology through the lack of authentication and cryptography. Those lacks directly threaten the integrity of the communication which is crucial and could be mitigated by the use of a public key infrastructure.

Appendix A

Appendix

A.1 Aircraft Identification categories

TC	CA	Category
1	ANY	Reserved
ANY	0	No category
2	1	Surface emergency vehicle
2	3	Surface service vehicle
2	4-7	Ground obstruction
3	1	Glider, sailplane
3	2	Lighter-than-air
3	3	Parachutist, skydiver
3	4	Ultralight, hand-glider, para-glider
3	5	Reserved
3	6	Unmanned Aerial Vehicle
3	7	Space or trans-atmospheric vehicle
4	1	Light (less than 7000kg)
4	2	Medium 1 (between 7000kg and 34000kg)
4	3	Medium 2 (between 34000kg and 136000kg)
4	4	High vortex aircraft
4	5	Heavy (larger than 136000kg)
4	6	High performance and high speed
4	7	Rotocraft

Table A.1: ACID categories based on TC and CA

A.2 Surface Position movement field non-linear speed

MOV Field	Ground speed	Increment
0	Speed not available	
1	Stopped (<125kt)	
2 - 8	0.125kt < v <1kt	0.125kt
9 - 12	1kt < v < 2kt	0.25kt
13 - 38	2kt < v < 15kt	0.5kt
39 - 93	15kt < v < 70kt	1kt
94 - 108	70kt < v < 100kt	2kt
109 - 123	100kt < v < 175kt	5kt
124	v > 175kt	
125 - 127	Reserved	

Table A.2: Surface Position movement non-linear table [Sun, 2021]

A.3 ICAO addresses range allocations

From	To	Country	From	To	Country	From	To	Country	From	To	Country
000000	003FFF	(unallocated)	098000	0983FF	Djibouti	4A8000	4AFFFF	Sweden	730000	737FFF	Iran
004000	0043FF	Zimbabwe	09A000	09AFFF	Gambia	4B0000	4B7FFF	Switzerland	738000	73FFFF	Israel
006000	006FFF	Mozambique	09C000	09CFFF	Burkina Faso	4B8000	4BFFFF	Turkey	740000	747FFF	Jordan
008000	00FFFF	South Africa	09E000	09E3FF	Sao Tome	4C0000	4C7FFF	Yugoslavia	748000	74FFFF	Lebanon
010000	017FFF	Egypt	0A0000	0A7FFF	Algeria	4C8000	4C83FF	Cyprus	750000	757FFF	Malaysia
018000	01FFFF	Libya	0A8000	0A8FFF	Bahamas	4CA000	4CAFFF	Ireland	758000	75FFFF	Philippines
020000	027FFF	Morocco	0AA000	0AA3FF	Barbados	4CC000	4CCFFF	Iceland	760000	767FFF	Pakistan
028000	02FFFF	Tunisia	0AB000	0AB3FF	Belize	4D0000	4D03FF	Luxembourg	768000	76FFFF	Singapore
030000	0303FF	Botswana	0AC000	0ACFFF	Colombia	4D2000	4D23FF	Malta	770000	777FFF	Sri Lanka
032000	032FFF	Burundi	0AE000	0AEFFF	Costa Rica	4D4000	4D43FF	Monaco	778000	77FFFF	Syria
034000	034FFF	Cameroon	0B0000	0B0FFF	Cuba	500000	5003FF	San Marino	780000	7BFFFF	China
035000	0353FF	Comoros	0B2000	0B2FFF	El Salvador	500000	5FFFFF	(reserved, EUR/NAT)	7C0000	7FFFFF	Australia
036000	036FFF	Congo	0B4000	0B4FFF	Guatemala	501000	5013FF	Albania	800000	83FFFF	India
038000	038FFF	Côte d Ivoire	0B6000	0B6FFF	Guyana	501C00	501FFF	Croatia	840000	87FFFF	Japan
03E000	03EFFF	Gabon	0B8000	0B8FFF	Haiti	502C00	502FFF	Latvia	880000	887FFF	Thailand
040000	040FFF	Ethiopia	0BA000	0BAFFF	Honduras	503C00	503FFF	Lithuania	888000	88FFFF	Viet Nam
042000	042FFF	Equatorial Guinea	0BC000	0BC3FF	St.Vincent + Grenadines	504C00	504FFF	Moldova	890000	890FFF	Yemen
044000	044FFF	Ghana	0BE000	0BEFFF	Jamaica	505C00	505FFF	Slovakia	894000	894FFF	Bahrain
046000	046FFF	Guinea	0C0000	0C0FFF	Nicaragua	506C00	506FFF	Slovenia	895000	8953FF	Brunei
048000	0483FF	Guinea-Bissau	0C2000	0C2FFF	Panama	507C00	507FFF	Uzbekistan	896000	896FFF	United Arab Emirates
04A000	04A3FF	Lesotho	0C4000	0C4FFF	Dominican Republic	508000	50FFFF	Ukraine	897000	8973FF	Solomon Islands
04C000	04CFFF	Kenya	0C6000	0C6FFF	Trinidad and Tobago	510000	5103FF	Belarus	898000	898FFF	Papua New Guinea
050000	050FFF	Liberia	0C8000	0C8FFF	Suriname	511000	5113FF	Estonia	899000	8993FF	Taiwan (unofficial)
054000	054FFF	Madagascar	0CA000	0CA3FF	Antigua & Barbuda	512000	5123FF	Macedonia	8A0000	8A7FFF	Indonesia
058000	058FFF	Malawi	0CC000	0CC3FF	Grenada	513000	5133FF	Bosnia & Herzegovina	900000	9FFFFF	(reserved, NAM/PAC)
05A000	05A3FF	Maldives	0D0000	0D7FFF	Mexico	514000	5143FF	Georgia	900000	9003FF	Marshall Islands
05C000	05CFFF	Mali	0D8000	0DFFFF	Venezuela	515000	5153FF	Tajikistan	901000	9013FF	Cook Islands
05E000	05E3FF	Mauritania	100000	1FFFFF	Russia	600000	6003FF	Armenia	902000	9023FF	Samoa
060000	0603FF	Mauritius	200000	27FFFF	(reserved, AFI)	600000	67FFFF	(reserved, MID)	A00000	AFFFFF	United States
062000	062FFF	Niger	201000	2013FF	Namibia	600800	600BFF	Azerbaijan	B00000	BFFFFF	(reserved)
064000	064FFF	Nigeria	202000	2023FF	Eritrea	601000	6013FF	Kyrgyzstan	C00000	C3FFFF	Canada
068000	068FFF	Uganda	280000	2FFFFF	(reserved, SAM)	601800	601BFF	Turkmenistan	C80000	C87FFF	New Zealand
06A000	06A3FF	Qatar	300000	33FFFF	Italy	680000	6FFFFF	(reserved, ASIA)	C88000	C88FFF	Fiji
06C000	06CFFF	Central African Republic	340000	37FFFF	Spain	680000	6803FF	Bhutan	C8A000	C8A3FF	Nauru
06E000	06EFFF	Rwanda	380000	3BFFFF	France	681000	6813FF	Micronesia	C8C000	C8C3FF	Saint Lucia
070000	070FFF	Senegal	3C0000	3FFFFF	Germany	682000	6823FF	Mongolia	C8D000	C8D3FF	Tonga
074000	0743FF	Seychelles	400000	43FFFF	United Kingdom	683000	6833FF	Kazakhstan	C8E000	C8E3FF	Kiribati
076000	0763FF	Sierra Leone	440000	447FFF	Austria	684000	6843FF	Palau	C90000	C903FF	Vanuatu
078000	078FFF	Somalia	448000	44FFFF	Belgium	700000	700FFF	Afghanistan	D00000	DFFFFF	(reserved)
07A000	07A3FF	Swaziland	450000	457FFF	Bulgaria	702000	702FFF	Bangladesh	E00000	E3FFFF	Argentina
07C000	07CFFF	Sudan	458000	45FFFF	Denmark	704000	704FFF	Myanmar	E40000	E7FFFF	Brazil
080000	080FFF	Tanzania	460000	467FFF	Finland	706000	706FFF	Kuwait	E80000	E80FFF	Chile
084000	084FFF	Chad	468000	46FFFF	Greece	708000	708FFF	Laos	E84000	E84FFF	Ecuador
088000	088FFF	Togo	470000	477FFF	Hungary	70A000	70AFFF	Nepal	E88000	E88FFF	Paraguay
08A000	08AFFF	Zambia	478000	47FFFF	Norway	70C000	70C3FF	Oman	E8C000	E8CFFF	Peru
08C000	08CFFF	D R Congo	480000	487FFF	Netherlands	70E000	70EFFF	Cambodia	E90000	E90FFF	Uruguay
090000	090FFF	Angola	488000	48FFFF	Poland	710000	717FFF	Saudi Arabia	E94000	E94FFF	Bolivia
094000	0943FF	Benin	490000	497FFF	Portugal	718000	71FFFF	Korea (South)	EC0000	EFFFFF	(reserved, CAR)
096000	0963FF	Cape Verde	498000	49FFFF	Czech Republic	720000	727FFF	Korea (North)	F00000	F07FFF	ICAO (1)
098000	0983FF	Djibouti	4A0000	4A7FFF	Romania	728000	72FFFF	Iraq	F00000	FFFFFF	(reserved)
									F09000	F093FF	ICAO (2)

Figure A.1: the messages as fast as possible in a single file as a batch

A.4 Source code snippets

```
1 def airborne_velocity(downlinkformat, ca, icao, tc, st
    , ic, ifr, nuc, param1, param2, param3, param4,
    vr_src, s_vr, vr, resv, s_dif,  dif, pi):
2     """
3     Encoding function of the Airborne Velocity type
        code
4     """
5     airborne_velocity_bytes = []
6     airborne_velocity_bytes.append((downlinkformat<<3)
        | ca)
```

```
7       airborne_velocity_bytes.append((icao>>16) & 0xff)
8       airborne_velocity_bytes.append((icao>> 8) & 0xff)
9       airborne_velocity_bytes.append((icao    ) & 0xff)
10      # data
11      airborne_velocity_bytes.append((tc<<3) | (st))
12      airborne_velocity_bytes.append((ic<<7) & 0x80 | (
           ifr<<6) & 0x40 | (nuc<<3) & 0x38 | (param1<<2)
           & 0x04 | (param2>>8) & 0x03)
13      airborne_velocity_bytes.append((param2) & 0xff)
14      airborne_velocity_bytes.append((param3<<7) & 0x80
           | (param4>>3) & 0x7f)
15      airborne_velocity_bytes.append(((param4<<5) & 0xe0
            | (vr_src<<4) & 0x10 | (s_vr<<3) & 0x08 | (vr
           >>6) & 0x07))
16      airborne_velocity_bytes.append((vr<<2) & 0xfc | (
           resv) & 0x03)
17      airborne_velocity_bytes.append((s_dif<<7) & 0x80 |
            (dif) & 0x7f)
18
19      if pi==None:
20        crc_data = crc(airborne_velocity_bytes)
21      else:
22        crc_data = pi
23
24      airborne_velocity_bytes.append((crc_data>>16) & 0
           xff)
25      airborne_velocity_bytes.append((crc_data>> 8) & 0
           xff)
26      airborne_velocity_bytes.append((crc_data    ) & 0
           xff)
27
28      return airborne_velocity_bytes
```

List of Tables

List of Figures

Bibliography

[arg, 2021] (2021). Argparse library documentation. https://docs.python.org/2.7/library/argparse.html. Accessed: 2021-05-28.

[con, 2021] (2021). Configparser library documentation. https://docs.python.org/2.7/library/configparser.html. Accessed: 2021-05-28.

[hac, 2021] (2021). hackrf_transfer man page. https://manpages.debian.org/unstable/hackrf/hackrf_transfer.1.en.html. Accessed: 2021-05-28.

[Aireon, June] Aireon (2016, June). Space based ads-b, icao sat meeting. "https://www.icao.int/WACAF/Documents/Meetings/2016/Lisbon-2016/Sat-21/SAT21%20WP%2009%20%20Space%20based%20ADS-B-Airon.pdf". Accessed: 2021-05-30.

[ASIA and OFFICE, r 04] ASIA, I. C. A. O. and OFFICE, P. (2011, November 04). *ADS-B IMPLEMENTATION AND OPERATIONS GUIDANCE DOCUMENT*. ICAO.

[Azimut, 2021] Azimut (2021). Ami 2700. s-band primary surveillance radar with monopulse mode s secondary surveillance radar. https://en.azimut.ru/netcat_files/141/170/h_ddf71289c325c3f678fe321e61621326. Accessed: 2021-05-30.

[DO-260A/ED-102A, l 10] DO-260A/ED-102A, R. (2003, April 10). Minimum operational performance standards for 1090mhz automatic dependent surveillancebroadcast (ads-b).

[DO-260B, r 02] DO-260B, R. (2009, December 02). Minimum operational performance standards for 1090mhz automatic dependent surveillancebroadcast (ads-b).

[ED-102, r 13] ED-102, R. D.-E. (2000, September 13). Minimum operational performance standards for 1090mhz automatic dependent surveillancebroadcast (ads-b).

[EUROCONTROL, 2021] EUROCONTROL (2021). Automatic dependent surveillance - broadcast airborne equipage monitoring. "https://www.eurocontrol.int/service/adsb-equipage". Accessed: 2021-05-30.

[EUROCONTROL, y 06] EUROCONTROL (2021, May 06). Integrating space-based ads-b into eurocontrol's network operations system brings major predictability gains and will unlock future capacity. https://www.eurocontrol.int/press-release/integrating-space-based-ads-b-eurocontrols-network-operations-system. Accessed: 2021-05-30.

[FlightAware, 2021] FlightAware (2021). dump1090-fa project. https://github.com/adsbxchange/dump1090-fa. Accessed: 2021-05-30.

[FlightRadar24, 2021] FlightRadar24, A. (2021). Flight radar 24 live air traffic. https://www.flightradar24.com/.

[H and S., 2020] H, G. and S., E. (2020). Simulating ads-b and cpdlc messages with sdr (dissertation).

[Institute, 2019] Institute, E. (2019). *Air Traffic Management and Systems III: Selected Papers of the 5th ENRI International Workshop on ATM/CNS (EIWAC2017)*. Lecture Notes in Electrical Engineering. Springer Singapore.

[Knight and Speers, 2018] Knight, M. and Speers, R. (2018). Designing rf fuzzing tools to expose phy layer vulnerabilities. DEF CON. https://doi.org/10.5446/39706 $Lastaccessed : 30May2021$.

[lyusupov, 2021] lyusupov (2021). Adsb-out. https://github.com/lyusupov/ADSB-Out.

[M. Ossman, 2021] M. Ossman, G. S. G. (2021). Hackrf one. https://greatscottgadgets.com/hackrf/. Accessed: 2021-05-30.

[Ma and Zhang, 2014] Ma, Y. P. and Zhang, J. (2014). Design and implementation of crc error correction for ads-b system responding based on fpga. In *Applied Decisions in Area of Mechanical Engineering and Industrial Manufacturing*, volume 577 of *Applied Mechanics and Materials*, pages 994–997. Trans Tech Publications Ltd.

[Nooelec, 2021] Nooelec (2021). Nooelec nesdr mini sdr & dvb-t usb stick (rtl2832 + r820t) w/ antenna. "https://www.nooelec.com/store/sdr/sdr-receivers/nesdr-mini.html". Accessed: 2021-05-30.

[Organization, 2014] Organization, I. C. A. (2014). *Aeronautical Telecommunications, Volume IV Surveillance and Collision Avoidance Systems, 10.IV 3.1.2.8.9.* 5 edition.

[Ossman, 2021a] Ossman, M. (2021a). Hackrf one documentation. https://github.com/mossmann/hackrf/wiki. Accessed: 2021-05-30.

[Ossman, 2021b] Ossman, M. (2021b). Hackrf repository. https://github.com/mossmann/hackrf.

[Ren et al., 2019] Ren, P., Wang, J., and Zhang, P. (2019). Novel error correction algorithms for ads-b signals with matched filter based decoding. *Physical Communication*, 36:100788.

[Schäfer et al., mber] Schäfer, M., Strohmeier, M., Smith, M., Fuchs, M., Lenders, V., Liechti, M., and Martinovic, I. (2017, September). Opensky report 2017: Mode s and ads-b usage of military and other state aircraft. In *IEEE/AIAA 36th Digital Avionics Systems Conference*, DASC.

[SESAR, 2021a] SESAR (2021a). Ads-b and sesar deployment manager role. "https://ads-b-europe.eu/". Accessed: 2021-05-30.

[SESAR, 2021b] SESAR (2021b). Ed-102b/do-260c approved for publication. "https://ads-b-europe.eu/ed-102b-do-260c-approved-for-publication/". Accessed: 2021-05-30.

[Skybrary, 2021a] Skybrary (2021a). Primary surveillance radar (psr). https://www.skybrary.aero/index.php/Primary_Surveillance_Radar_(PSR). Accessed: 2021-05-28.

[Skybrary, 2021b] Skybrary (2021b). Secondary surveillance radar (ssr). https://www.skybrary.aero/index.php/Secondary_Surveillance_Radar_(SSR). Accessed: 2021-05-28.

[Sun, y 30] Sun, J. (2017, January 30). *ADS-B Decoding Guide*. TU Delft OPEN Publishing, 0.3 edition.

[Sun, 2021] Sun, J. (2021). *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*. TU Delft OPEN Publishing, 2 edition.

[Txtav, 2021] Txtav (2021). Ads-b out explained. https://txtav.com/en/journey/articles/articles/adsb-out-explained. Accessed: 2021-05-30.

[UNOOSA, 2021] UNOOSA, U. N. O. f. O. S. A. (2021). Global navigation satellite systems (gnss). "https://www.unoosa.org/oosa/en/ourwork/psa/gnss/gnss.html". Accessed: 2021-05-30.

[Wu et al., 2020] Wu, Z., Shang, T., and Guo, A. (2020). Security issues in automatic dependent surveillance - broadcast (ads-b): A survey. *IEEE Access*, 8:122147–122167.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**