Exploring the Vulnerabilities of Traffic Collision Avoidance Systems (TCAS) Through Software Defined Radio (SDR) Exploitation

Paul M. Berges

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

May 3, 2019

Blacksburg, Virginia

Keywords: Cyber-Physical, Security, TCAS, SSR, SDR

Exploring the Vulnerabilities of Traffic Collision Avoidance Systems (TCAS) Through Software Defined Radio (SDR) Exploitation

Paul M. Berges

(ABSTRACT)

Traffic Collision Avoidance Systems (TCAS) are safety-critical systems that are deployed on most commercial aircraft in service today. However, TCAS transactions were not designed to account for malicious actors. While in the past it may have been infeasible for an attacker to craft arbitrary radio signals, attackers today have access to open-source digital signal processing software like GNU Radio and inexpensive Software Define Radios (SDR). Therefore, this thesis presents motivation through analytical and experimental means for more investigation into TCAS from a security perspective. Methods for analyzing TCAS both qualitatively and quantitatively from an adversarial perspective are presented, and an experimental attack is developed in GNU Radio to perform an attack in a well-defined threat model.

Exploring the Vulnerabilities of Traffic Collision Avoidance Systems (TCAS) Through Software Defined Radio (SDR) Exploitation

Paul M. Berges

(GENERAL AUDIENCE ABSTRACT)

Since 1993, the Federal Aviation Administration (FAA) requires that many commercial turbine-powered aircraft to be outfitted with an on-board mid-air collision mitigation system. This system is known as the Traffic Collision Avoidance System (TCAS) in the United States, and it is known as the Airborne Collision Avoidance System (ACAS) in other parts of the world. TCAS/ACAS is a type of safety-critical system, which means that implementations need to be highly tolerant to system failures because their operation directly affects the safety of the on-board passengers and crew. However, while safety-critical systems are tolerant to failures, the designers of these systems only account for failures that occur in a cooperative environment; these engineers fail to account for "bad actors" who want to attack the weaknesses of these systems, or they assume that attacking such a system is infeasible. Therefore, to demonstrate how safety-critical systems like TCAS/ACAS are vulnerable to such bad actors, this thesis presents a method for manipulating the TCAS/ACAS in the favor of a bad actor. To start, a method for qualitatively and quantitatively analyzing the system's vulnerabilities is presented. Then, using Software Defined Radio (SDR), which is a free and open-source effort to combine the flexibility of software with the power of wireless communication, this thesis shows how an actor can craft wireless signals such that they appear to look like an aircraft on a collision course with a target.

Dedication

For home, for love, and for the present moment. Take your time.

Acknowledgments

I would like to thank my advisor, Dr. Ryan Gerdes, for his guidance throughout this work. His insight, support, and professionalism has proven to be essential towards completing this thesis. There were many times when self-doubt could have gotten the better of me, and he was always instrumental towards helping me find clarity.

Also, I would like to acknowledge my family and friends who have always shown me unconditional love and support. Many laughs, anxieties, successes, and tears were wrought from this work, and you were always there to keep me grounded. I would have done much less without you by my side throughout everything, and I will always be grateful for that.

Finally, I would like to acknowledge the Virginia Tech faculty who have inspired me to be a better leader and engineer. Specifically, I would like to acknowledge the faculty of the Bradley Department of Electrical and Computer Engineering of whom I have had the pleasure of studying under, and whose passion as educators drives the students to invent the future. I would also like to acknowledge those who I have had the pleasure of working with during my service as an on-campus residential adviser. They taught me the lesson that leadership is more than confidence in one's own work; it is also the drive to be an empathetic professional that motivates personal change for the good of an individual, a team, and a community.

Contents

List of Figures

List of Tables

List of Abbreviations

AA	Address Announced field
AC	Altitude Code field
ACA	All-Call Address
ACAS	Airborne Collision Avoidance System
ADC	Analog to Digital Conversion
ADS-B	Automated Dependent Surveillance - Broadcast
ALIM	Altitude Limit
AP	Address/Parity field
AQ	Acquisition field
ARA	Active RAs field
ATC	Air Traffic Control
ATCAC	Air Traffic Control Advisory Committee
ATCRBS	Air Traffic Control Radar Beacon System
AWGN	Additive White Gaussian Noise
BCAS	Beacon Collision Avoidance System
CA	Capability field

CC	Cross-link Capability field
CHC	Cancel Horizontal RAC field
CIA	Confidentiality, Integrity, and Availability
CL	Code Label field
CPA	Closest Point of Approach
CRC	Cyclic Redundancy Check
CVC	Cancel Vertical RAC field
DABS	Discrete Address Beacon System
dB	Decibels
DBPSK	Differential Binary Phase Shift Keyed modulation
DF	Downlink Format header field
DMOD	Distance Modification
DoS	Denial-of-Service
DS	Data Selector field
FAA	Federal Aviation Administration
FIFO	First-in, First-out message queue
FPGA	Field Programmable Gate Array
fpm	Feet per minute
ft	Feet

FTA	Fault Tree Analysis
GA	General Aviation
GRC	GNU Radio Companion
HIL	Hardware-in-the-Loop
HMD	Horizontal Miss Distance
HRC	Horizontal RAC field
HSB	Horizontal Sense Bits field
IC	Interrogator Code field
IFF	Identification Friend and Foe
IMC	Instrument Meteorological Conditions
Interrogation	A request for information from an intruder, broadcast on the 1030 MHz frequency band
kt	Knots
MAC	Mid-Air Collision
MDF	Miss Distance Filter
MID	Interrogating TCAS aircraft address field
MOPS	Minimum Operational Performance Standards
MSB	Most Significant Bit
MTB	Multiple Threat Bit

MTE	Multiple Threat Encounter field
MU	ACAS Message, U-definition
MV	ACAS Message, V-definition
NM	Nautical Miles
NMAC	Near Mid-Air Collision
OOT	Out-of-Tree GNU Radio module
PI	Parity Identification field
PMT	GNU Radio Polymorphic Data Type
PPM	Pulse Position Modulation
PR	Probability of Reply field
RA	Resolution Advisory
RAC	Resolution Advisory Complement or RACs record field
RAT	RA Terminated indicator
Reply	An encoded response of aircraft data, broadcast on the 1090 MHz frequency band
RI	Reply Information field
RL	Reply Length field
RTCA	Radio Technical Commission for Aeronautics
RTT	Round Trip Time
RX	Receiver

SDR	Software Defined Radio
SL	Sensitivity Level or Sensitivity Level field
SLS	Side Lobe Suppression
SSR	Secondary Surveillance Radar
TA	Traffic Advisory
Tau	Time-to-go until CPA threshold
TCAS	Traffic Collision Avoidance System
TVTHR	Time to Co-altitude Threshold
TX	Transmitter
UDS	U-Definition Sub-field
UF	Uplink Format header field
UHD	USRP Hardware Driver
USRP	Universal Software Radio Peripheral
VDS	V-Definition Sub-field
VRC	Vertical RAC field
VS	Vertical Status field
VSB	Vertical Sense Bits field
VSL	Vertical Speed Limit
ZTHR	Absolute Altitude Threshold

Chapter 1

Introduction

In the year 2016, commercial aircraft carried nearly four billion passengers to their destinations [44]. The volume of daily passengers on commercial aircraft makes them the perfect target for bad actors such as hackers or terrorists to cause large-scale harm in a short amount of time. Although flight may seem like a precarious endeavor at times, aviation remains the safest way to travel because of the various safety-critical systems operating at any given moment on the aircraft [32].

One such on-board safety feature is known as the Traffic Collision Avoidance System (TCAS). Internationally, this system is known as the Airborne Collision Avoidance System (ACAS). The purpose of this module is to prevent the mid-air collisions of transponder-equipped aircraft. Many aviation regulatory bodies mandate the use of TCAS on larger aircraft on larger commercial aircraft [10]. However, while manipulation of TCAS's wireless communications may have been infeasible in the past, today's technology such as Software Defined Radio (SDR) enables the manipulation of TCAS through software-defined wireless signals that are designed to appear like one or more aircraft on a collision course with a target aircraft because TCAS has no built-in security features to combat these false signals.

Therefore, this thesis underscores the importance of the TCAS's role in improving the safety of aircraft. It motivates change in the design of TCAS applications by highlighting the difference between "safety-critical" systems and "secure" systems, and it shows that systems designed for safety do not imply security from attack.

To meet these goals, the thesis demonstrates the fallacy between safety and security by creating an attack against TCAS. First, an analytical approach is taken to find vulnerabilities and model the effect of failures from attacks. Then, a threat model is assumed, and an attack is developed within that model; this attack shows that open-source software can be used to spoof critical TCAS II components (e.g. a Mode S Transponder [49]), which enables an attacker to masquerade as a collision-bound aircraft.

1.1 Motivation

TCAS plays an essential role in airborne safety. When Air Traffic Control (ATC) towers cannot react in time, TCAS is critical for warning pilots to change course and prevent a mid-air collision. Some safety studies estimate that the mandated use of TCAS improves safety by a factor of between three and five. Therefore, these estimates demonstrate that improving TCAS is a worthwhile endeavor [10].

Before the TCAS mandate, mid-air collisions were a very real possibility in air spaces without ATC coverage. Most documentation cites the incident over the Grand Canyon in 1956 as the catalyst for developing a collision mitigation safety system [15]. The mid-air collision over the Grand Canyon killed all 128 occupants in the two airplanes involved. This accident motivated the creation of the Federal Aviation Administration (FAA) because, "The accident dramatized the fact that, even though U.S. air traffic had more than doubled since the end of World War II, little had been done to mitigate the risk of midair collisions" [9, 12].

It is evident that the creation of the FAA and deployment of technologies like TCAS have reduced the risk of mid-air collision. However, studies on TCAS show that there is still a risk of TCAS *inducing* two planes that are not on a collision course to steer into each other [10, 15, 40]. The problem of induction is greater than indicated because the design of TCAS

Figure 1.1: Burned area shows where a United Airlines flight crashed after its mid-air collision [9].

predates the creation of technologies like Software Defined Radio (SDR) which could enable anyone to spoof TCAS signals for the purpose of masquerading as a threatening aircraft. This implies that TCAS was never intended to perform under adversarial conditions of which are entirely feasible in today's environment of inexpensive, powerful computers. If an attacker were to compromise TCAS, they can bypass the safety benefits granted by TCAS equipage. Worse, if an attacker has the ideal conditions, a TCAS-equipped aircraft could have a *greater* chance of mid-air collision than an unequipped aircraft.

1.2 Contributions

This thesis contributes to the ongoing research into the defense of cyber-physical systems. Specifically, this thesis is intended to demonstrate that the TCAS Secondary Surveillance Radar (SSR) transactions, which are comprised of Mode S transponder messages, are vulnerable to attack. TCAS vulnerabilities will be explored using the weakest possible attack model using only open-source software, publicly-accessible knowledge about TCAS, and a SDR. It should be noted that the demonstrated attack capabilities are rudimentary, and a more sophisticated attack could be developed against TCAS. Hopefully, this thesis motivates further research into the defense of TCAS and Mode S transponders.

Most prior work into the security of the Mode S transponder has been limited to a specific type of message spoof: Automated Dependent Surveillance - Broadcast (ADS-B) message spoofing [3, 69]. TCAS and ADS-B are closely linked because both messages are transmitted through the Mode S transponder, and ADS-B messages can be used in TCAS operation in a technique called *hybrid surveillance* [15]. While works on ADS-B propose some defenses against spoofing attacks, ADS-B is not part of any safety-critical systems on an aircraft, so sweeping changes to the transponder for the purpose of security are most likely to be thought as too expensive. However, if TCAS is also shown to be vulnerable to attack, then the safety of on-board passengers would be compromised, which should be a greater motivator of changes to the system.

1.3 Organization

Chapter 2 introduces the background material necessary to understand the basis of the attack. Specifically, an introduction to TCAS and Mode S transponders will be presented, and then, a discussion about *safety-critical systems* will occur. Then, the required security basics to understand TCAS from an adversarial perspective will be discussed. Finally, the tools used to develop an attack will be explained, such as attack trees and Fault Tree Analyses (FTA).

Chapter 3 outlines how an old FTA of TCAS is reverse engineered for the purpose of quantifying how attacks can manipulate the risks associated with TCAS. Such a quantification introduces objectivity into the claims asserted against TCAS's security.

Chapter 4 discusses how to analyze the weaknesses of TCAS through an *attack tree*. Then, in Chapter 5, an attack is chosen, and a model of the attack is presented. The attacker's goals and capabilities are outlined, the attack itself is modeled, and key protocols are analyzed to effectively enact the attack.

Chapter 6 demonstrates the attack platform developed. The platform uses GNU Radio - an open-source software platform for processing streams of digital radio data - as the foundation of its architecture. Block diagrams of the platform are shown, and the methods for simulation are presented. Lastly, an outline of the hardware used for this platform is presented.

Chapter 7 outlines the successes and failures of the design in Chapter 6 through a series of verification tests. Finally, Chapter 8 discusses the results of the development, the security implications for safety engineers of TCAS to consider, and future work for TCAS security.

Chapter 2

Background Topics

2.1 Introduction to ACAS/TCAS and Mode-S

A Traffic Collision Avoidance System (TCAS) is a system designed to reduce the risk of near mid-air collisions (NMAC) between aircraft. It operates independently of Air Traffic Control (ATC) to create a *protected volume* of airspace that, when intruded by other aircraft, warns the pilot of danger. It also works cooperatively with an on-board active transponder (such as a Mode S Transponder) to facilitate the air-to-air communications required for collision avoidance.

The focus of this thesis is on the TCAS II system which assumes that all aircraft are equipped with a TCAS II computer and a Mode S transponder. TCAS II can only provide its full benefits when it encounters another TCAS II because they can coordinate a maneuver together that will maximize vertical separation. Unless otherwise specified, *TCAS* will specifically refer to the TCAS II Version 7.1+ release of the TCAS specification [15].

To understand the purpose of TCAS, it is necessary to study its development history. Many factors, including the rise of mid-air collisions, put pressure on the industry to develop technology that could thwart NMACs. The development of TCAS and its Mode S transponder spans greater than a decade, and many institutions are maturing the technology today [38].

After studying the history of TCAS and Mode S, it is imperative to define the system components and the communication protocols they use. Then, a summary of how TCAS discovers, tracks, and thwarts NMACs will be presented.

2.1.1 History of TCAS

In December, 1960, the citizens of New York City experienced an unprecedented disaster. The incident, which is known as the *Park Slope Crash of 1960*, happened when a Trans World Airline flight collided mid-air with a United flight over the skies of Brooklyn. Unlike the disaster of the Grand Canyon in 1956 [9], this disaster was one of the first of its kind to be televised. As the *New York Times* describes, ". . . viewers were jolted by the footage that was finally shown, hours after the crash, of firemen digging bodies out of the debris." The accident killed 136 people [2, 6]

Figure 2.1: Park Slope, Brooklyn, NY the day of the 1960 crash [2]

In 1961, the FAA was requested to develop a long term plan to improve the safety of U.S. air space control. A committee from the FAA identified two opposing forces of change to air traffic control. One solution to air traffic control was a complete redesign of ATC computing, including a "3D-radar" that was never realized. However, ATC operators insisted that the focus should be on improving existing technologies. Ultimately, the ATC operators won, and efforts were focused on improving primary radar with the end users in mind [6].

The engineers of the effort created the Air Traffic Control Radar Beacon System (ATCRBS). The system was adapted from Identification Friend and Foe (IFF) communicators, in which ground-based interrogators would expect a reply broadcast from nearby planes that could decode the ground interrogations. It shares the same frequency bands as IFF and had two types of interrogations:

- Mode A interrogations would elicit a target's identity
- Mode C interrogations would elicit a target's altitude information [6]

ATCRBS was intended to extract extraneous information from aircraft and was designed to assist primary radar [1]. However, initial deployments found many limitations. Primarily, any aircraft that received the interrogation would reply; this would flood the reply channel, and any information from the responding aircraft would be rendered undecipherable [6].

Figure 2.2: Aftermath of the mid-air collision in Cerritos, CA which killed 82 people in 1986 [24]

Therefore, a committee formed by the FAA called the Air Traffic Control Advisory Committee (ATCAC) realized a new system that was backwards compatible with ATCRBS while also implementing garble-limiting measures like discrete aircraft addressing. In 1971, the FAA contracted Lincoln Labs to design a new system called the Discrete Address Beacon System (DABS). This system would go on to become Mode S [6].

The largest constraint in the redesign was interoperability - the concept that Mode S should work entirely with ATCRBS to lower costs. There were three overall themes to the design of Mode S:

1. *Discreteness*: In general, interrogations and replies should occur at a 1:1 ratio.
2. *Backwards Compatibility*: Mode S should communicate with Mode A and Mode C transponders.
3. *Transparency*: ATCRBS stations should not attempt to interpret a Mode S reply. [6].

In 1981 the FAA decided to shift its efforts from creating a ground-based collision avoidance system to an air-to-air system. This changed the original ground-based designs - the Beacon Collision Avoidance System (BCAS) - into TCAS. However, although the official proposal for Mode S happened in 1975, many manufacturers were not willing to manufacture Mode S transponders and sensors, citing cost concerns. It was not until after a 1986 mid-air collision over Cerritos, CA that Congress enacted a 1987 law called the *Airport Airway Safety and Capacity Expansion Act* which mandated TCAS II to be fitted onto aircraft with more than 30 passenger seats by 1993 [24]. After the TCAS mandate, it was not until 1997 when the last major revision was made to TCAS. This major revision - Version 7 - became the international standard for call TCAS implementations [6, 8].

2.1.2 ACAS/TCAS System Description

System Description

TCAS uses the active Mode S transponder (SSR) to communicate messages. Each TCAS interrogates any nearby aircraft on the 1030 MHz frequency band, and all aircraft reply on the 1090 MHz frequency band. TCAS uses these transactions to track nearby aircraft and monitor any aircraft that *intrude* the nearby airspace. This protected airspace is defined by altitude and the remaining time until an intruder reaches its Closest Point of Approach (CPA) [15, 58]. Figure 2.3 shows the a protected region generated by TCAS.

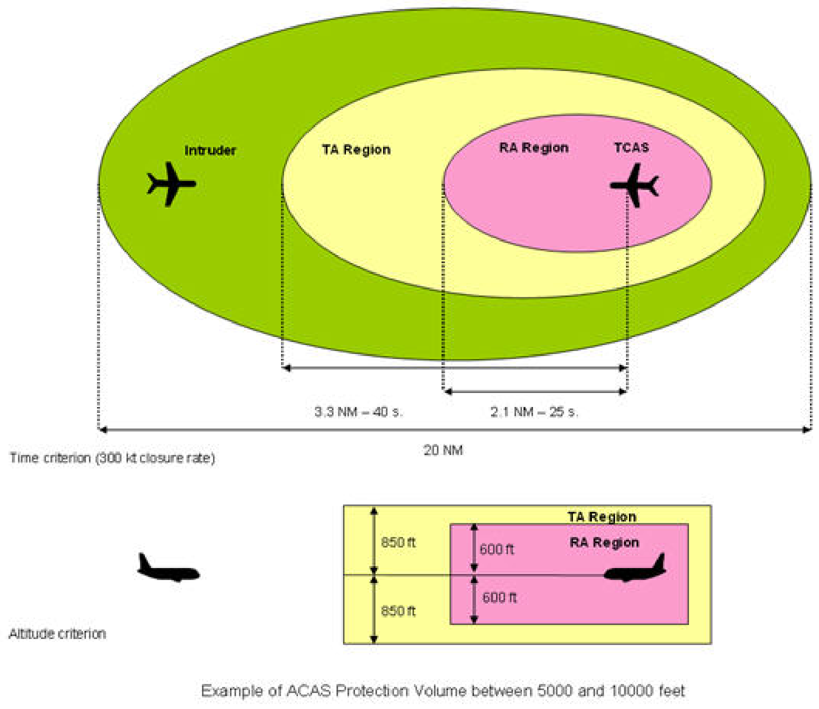


Figure 2.3: The region protected by TCAS [58].

TCAS can be operated in three modes which are controlled by the pilot [15]. The level of functionality TCAS provides is defined by its Sensitivity Level (SL). These levels are:

1. (SL 1) *Standby*: No TCAS tracking or squitters. Mode S transponder will respond to interrogations.

2. (SL 2) *TA Only*: TCAS will track intruding aircraft, but it will only announce a TA to the pilot. Mode S transponder transmits squitters and responds to interrogations.

3. (SL 3+) *TA-RA*: TCAS will automatically select the best SL for the altitude. Table

A.1 provides the altitude thresholds for each SL. TCAS performs all functions and will issue TAs and RAs.

In some instances, ATC towers can control the SL of a TCAS. However, ATC cannot fully disable TCAS functions. Ground controllers can only place TCAS into SL 2 or greater [15, 34, 35].

Although there are more semantics to each state in the functional flow of TCAS, these can be generalized for the purpose of an arbitrary encounter with an intruder. Figure 2.4 shows the actual overall functions of TCAS, which will be generalized as follows:

1. Surveillance: The detection of transponder-equipped aircraft in the proximity.
2. Tracking: Estimating the state of all intruding aircraft, including, but not limited to, altitude, time-to-CPA, etc.
3. Advisory: An intruder meets the requirements to be declared a threat, and a TA is announced to the pilot.
4. Coordination: TCAS coordinates a maneuver with the intruder and announces said maneuver to the pilot.

There is a large set of rules that govern when alerts arise. A selection of these rules are in Appendix A. There are two types of advisories that a TCAS can produce, which is illustrated in Figure 2.5:

- A **Traffic Advisory** (TA), which is intended to help a pilot visually locate an intruding aircraft and happens earlier than an RA.
- A **Resolution Advisory** (RA) will recommend maneuvers or positional holds in order to increase the separation between the aircraft itself and the intruder [15].

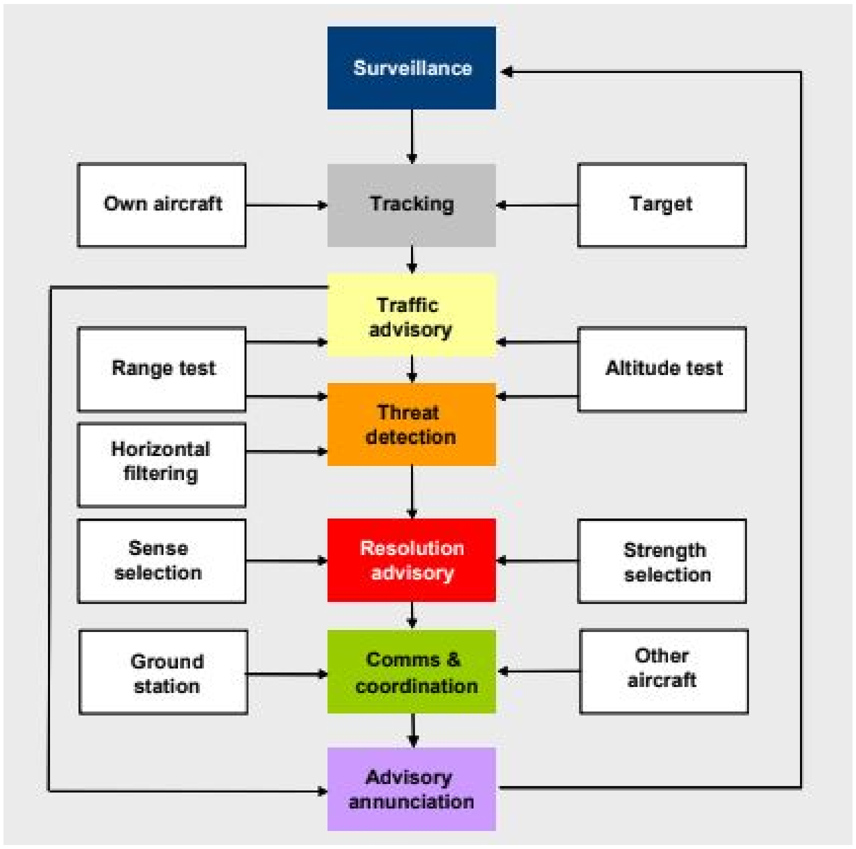


Figure 2.4: High-level diagram of TCAS functions [11].

When TCAS is choosing the types of maneuver to select, there are a few factors TCAS will favor over others. In a coordinated maneuver, the aircraft that announces its RA first controls the maneuvers of the the second aircraft. In other words, the aircraft that makes the first decision will *suppress* the possible maneuvers the other aircraft can do. Then, the TCAS will choose the maneuver's **sense** and **strength** [15].

A **sense** is the direction the aircraft will maneuver. TCAS favors choosing a sense that does not cross paths with the other aircraft, provided that the maneuver will sufficiently separate the two aircraft (this threshold is defined as **ALIM**). Required ALIM for a given altitude of the encounter is described Table A.1. In TCAS II, there are two possible senses: *upward* or *downward*; these senses prescribe that the plane either increases or decreases its altitude,

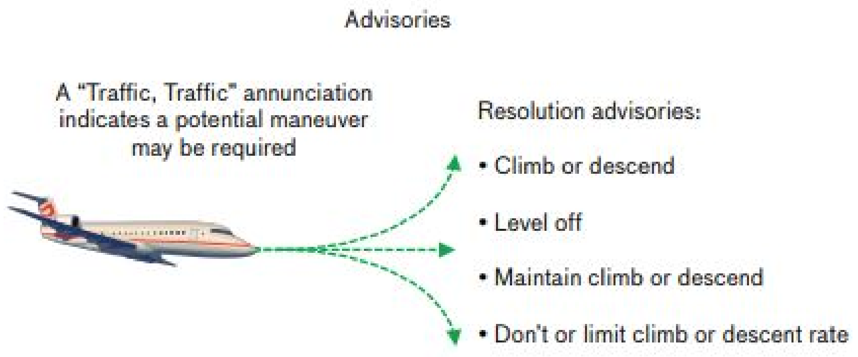

Advisories
A "Traffic, Traffic" annunciation indicates a potential maneuver may be required
Resolution advisories:
• Climb or descend
• Level off
• Maintain climb or descend
• Don't or limit climb or descent rate

Figure 2.5: TAs alert the pilots to the presence of an intruder, whereas RAs will recommend a maneuver for the pilot to take [38].

respectively. The difference between these senses are illustrated in Figure 2.6.

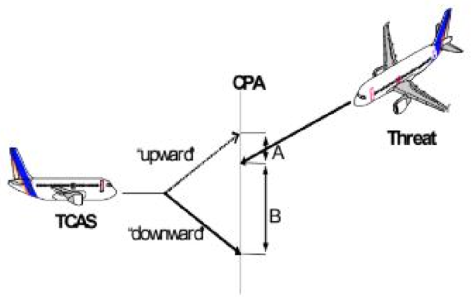


Figure 2.6: Encounter "sense" selection [15]

After a sense selection, TCAS will choose a maneuver strength. A **strength** can either be *positive* or *negative*. Positive strengths command that the plane climb or descend, whereas negative strengths impose, "Vertical Speed Limits" (VSL) on the plane, e.g., "limit climb to 500 fpm." TCAS will favor strengths that impose the least change on a plane's flight path. Table A.2 shows all possible corrections against a single intruder [15].

According to [58], "Pilots are required to comply immediately with all RAs, even if the RAs are contrary to ATC clearances or instructions." Therefore, it can reasonably be assumed that if an RA is present to a pilot, they will likely follow the RA. When an RA is announced, TCAS will send special messages to the intruder that will coordinate a maneuver to maximize separation [15]. Some ground stations can retrieve these special messages to notify ATC, although such ground stations are not currently required by any standard [58, 60].

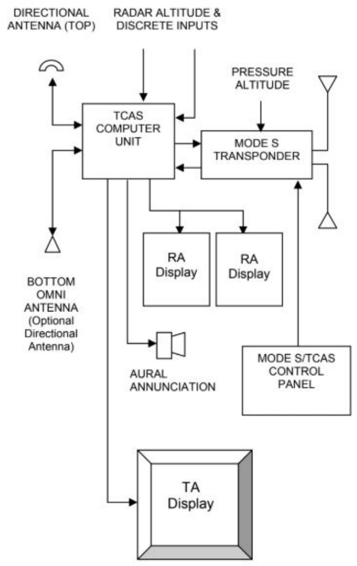

DIRECTIONAL ANTENNA (TOP)
RADAR ALTITUDE & DISCRETE INPUTS
PRESSURE ALTITUDE
TCAS COMPUTER UNIT
MODE S TRANSPONDER
RA Display
RA Display
BOTTOM OMNI ANTENNA (Optional Directional Antenna)
AURAL ANNUNCIATION
MODE S/TCAS CONTROL PANEL
TA Display

Figure 2.7: Block diagram of TCAS II [15].

System Components

TCAS works independently from the Mode S transponder in general. Its components are illustrated in Figure 2.7. There are three main components to a TCAS:

- TCAS CPU: A computer that performs interrogations, surveillance, tracking, threat declaration, etc.
- Antennas: There are three required antennas and one optional antenna, as shown in Figure 2.7. However, many implementations choose a shared antenna design to save cost.
- Cockpit displays: The cockpit announces a TA or an RA using an aural announcement

and a visual display An example display is shown in Figure 2.8.

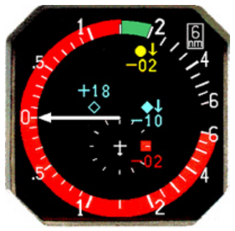


Figure 2.8: Example of a TCAS II traffic display with a “Climb 1500 ft/min” RA, as indicated by the square, red icon [58].

Generally, TCAS behaves within a request-response architecture. A TCAS CPU is responsible for using the 1030 MHz channel to request information from an intruder. This action is called an **interrogation**. The Mode S transponder, which is on an intruding aircraft, is responsible for responding to interrogations [34, 49].

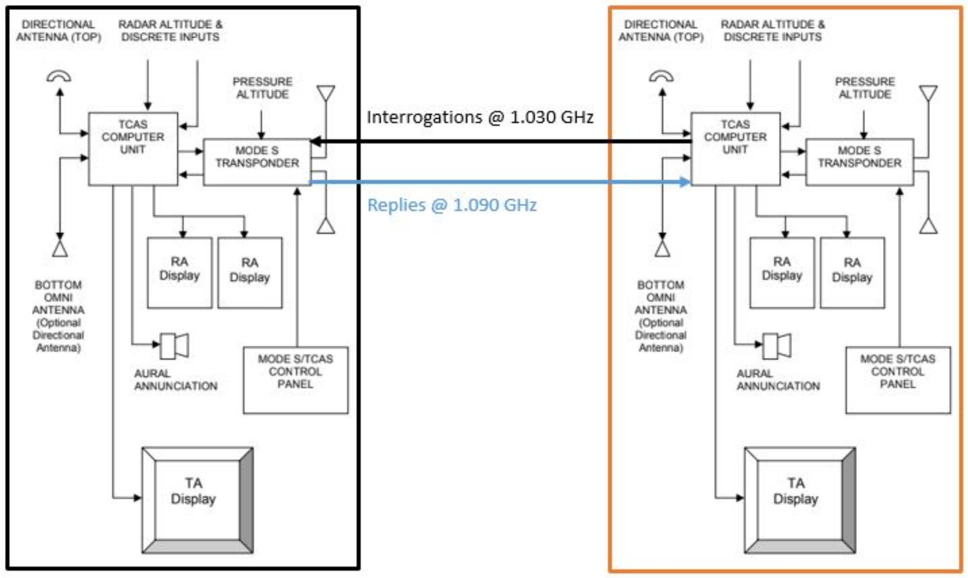

DIRECTIONAL ANTENNA (TOP)
RADAR ALTITUDE & DISCRETE INPUTS
PRESSURE ALTITUDE
TCAS COMPUTER UNIT
MODE S TRANSPONDER
RA Display
RA Display
BOTTOM OMNI ANTENNA (Optional Directional Antenna)
AURAL ANNUNCIATION
MODE S/TCAS CONTROL PANEL
TA Display
Interrogations @ 1.030 GHz
Replies @ 1.090 GHz

Figure 2.9: Two TCAS systems communicating to each other [15].

An interrogation is comprised of the pulse sequence shown in Figure 2.10. The pulses labeled P1, P2, and P5 are used in a feature called *side-lobe suppression* which prevents Mode A/C transponders from attempting to decode the message. The side-lobe suppression functionality can be ignored for the purpose of this thesis; only their synchronization functionality is considered. The pulse labeled P6 contains the interrogation data and is Differential Binary Phase Shift Keyed (DBPSK) modulated at a rate of 4 mega-symbols per second [35].

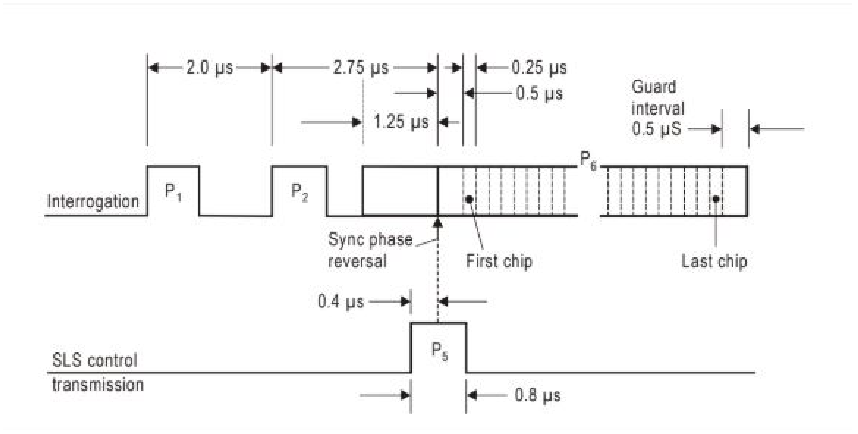


Figure 2.10: Interrogation Pulse Sequence [35].

01011	PR:4	IC:4	CL:3	16	AP:24

Figure 2.11: Example of a UF 11 message [35].

To interrogate a plane, its address must be known. Each plane has a unique 24-bit address called its ICAO address [34, 35]. All interrogations are addressed to the aircraft receiving the interrogation. ICAO addresses are combined using an exclusive-or function with a unique Mode S cyclic redundancy check (CRC) calculation to form the Address/Parity (AP) field [30].

The data encoded in an interrogation is referred to as an **Uplink Format** (UF) message. UF messages can request a variety of data from an intruder and are either 56 or 112 bits long. UF messages have a fixed-length 5 bit header, and its value indicates what type of message it is. Figure 2.11 shows an example of a UF 11 message, which is also known as the "Mode S only All-Call". The leftmost bit is transmitted first, where the first bit transmitted is the most significant bit (MSB). It has a 5 bit header, a two 4 bit fields, a three bit field, 16 bits of unused space, and a 24 bit address field [34, 35].

Likewise, the Mode S transponder of an intruding aircraft formulates a response to the interrogation. This response is called a **reply**. Replies are broadcast on the 1090 MHz frequency band and are comprised of pulses that differ from those shown in Figure 2.10. The Mode S reply pulse sequence is given in Figure 2.12.

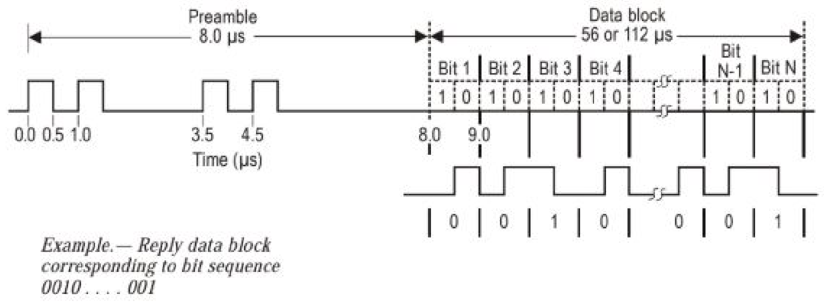
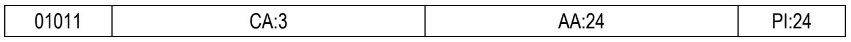


Figure 2.12: Reply Pulse Sequence [35].



01011	CA:3	AA:24	PI:24

Figure 2.13: Example of a DF 11 Message [35].

Replies only happen if an interrogation's decoded AP field matches the ICAO address of the receiving aircraft. There is also a special address comprised of 24 binary ones called the **All-call Address** (ACA). When this address is used with the UF 11 interrogation in Figure 2.11, all transponders that receive this interrogation will reply.

The data encoded in a reply is referred to as a **Downlink Format** (DF) message. DF messages can also be 56 or 112 bits long, and they also have a fixed-length 5 bit header which indicates what type of reply it is. The DF header's value will always match the UF header's value of the message that started the transaction. Figure 2.13 shows an example of a DF 11 reply, which is the response formed from the message from Figure 2.11.

Unlike all other messages, DF 11 messages contain the plain ICAO address of the plane that is sending the reply. These messages allow TCAS to determine the addresses of nearby planes. This address is contained in the 24 bit Address Announced (AA) field. A different parity calculation result is appended at the end of the message in the Parity Identification (PI) field [30].

A UF 11 interrogation transmitting with the ACA is not required to get the address of each nearby plane. DF 11 is a special reply message which is used as a **squitter**. A squitter is an exception to the request-response architecture because Mode S transponders periodically broadcast it while in operation. ADS-B mainly uses squitters, but TCAS can also use squitters to passively acquire nearby aircraft for the purpose of reducing the congestion of the interrogation frequency band [34, 35, 49]. This thesis will disregard the squitter for the sake of keeping the request-response model consistent.

With the TCAS thoroughly described and its components outlined, a discussion on how a TCAS acquires, tracks, and advises against intruding aircraft is possible. The next section will outline how TCAS uses these components in a hypothetical airspace intrusion.

2.1.3 Example Near-Miss Encounter

First, assumptions are made for the example encounter. No other aircraft are being tracked by either plane at the time of encounter. Also, squitters will not be used in this example encounter for the purpose of maintaining a consistent request-response architecture. Instead, the reference point aircraft will simply know when to interrogate the nearby airspace. Finally, each TCAS is set to `TA-RA` mode, so their sensitivity levels will be automatically selected for them.

The example encounter is shown in Figure 2.14. Two and only two aircraft are approaching directly towards each other with 500 ft of separation. No other aircraft are within surveillance range. According to the tables of Appendix A, these planes are approaching a NMAC. The time until CPA, or Tau, is 48 seconds, which is the exact moment the planes should announce a Traffic Advisory to their respective pilots.

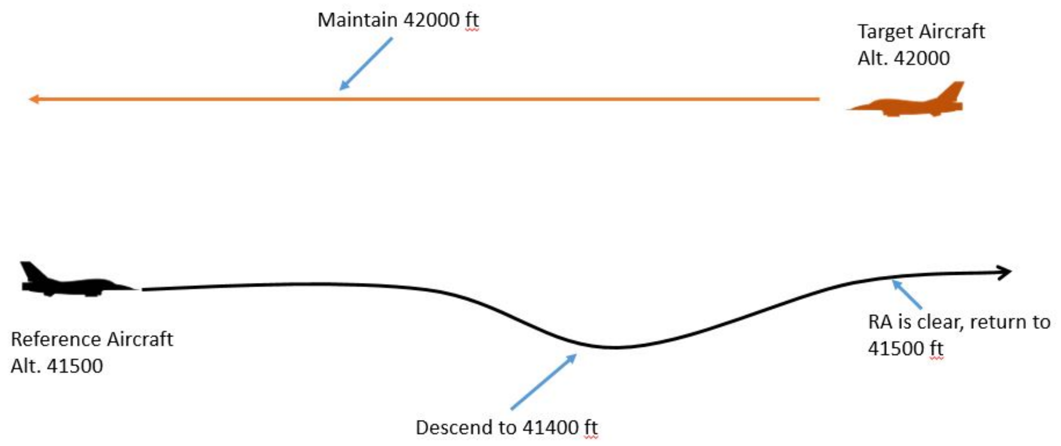


Figure 2.14: Simple example of a NMAC scenario

With respect to the reference aircraft, the order of subsequent events is as follows:

1. Reference aircraft surveys the target aircraft.

2. Reference aircraft acquires the target aircraft.

3. Reference aircraft tracks the target aircraft.

4. Reference aircraft declares the target as a threat, and a TA is announced to the pilot.

5. RA selection and coordination occurs; RA is announced to the pilots.

6. Maneuver moves the planes into ALIM vertical spacing.

Surveillance/Acquisition

TCAS must acquire intruding aircraft in its protected airspace (Figure 2.3) before performing any of its functions (Figure 2.4). Although acquisition is also possible by detecting squitters, this example will survey the nearby airspace by using an All-Call interrogation, UF 11. The target aircraft receives this interrogation, verifies that the decoded AP field is equivalent to the ACA, and replies with its address in the AA field of its DF 11.

TCAS can choose when to transition into tracking based on its implementation. However, in general, a TCAS will verify that it receives DF 11 from the same aircraft over multiple interrogations - a process known as **acquisition interrogation**. The target is declared valid after some amount of acquisition interrogation [34].

For altitude acquisition, the target is interrogated with UF 0. Altitude is sent as information encoded in the DF 0 reply. This altitude is quantized in either 100 ft or 25 ft increments, or altitude can be reported in meters. Targets must be within 10,000 ft of the interrogating aircraft to be tracked [34, 35].

For range acquisition, TCAS uses an *alpha-beta* state space estimator. An instantaneous range estimate is inferred from the round-trip time (RTT) of the surveillance sequence. This estimator is defined by the equations:

$$r(t)_e = r(t)_p + [\alpha \times (r(t)_m - r(t)_p)]$$

$$s(t)_e = s(t - T_p)_e + [(\beta / T_p) \times (r(t)_m - r(t)_p)]$$

where:

- T_p is the time difference between the current and previous estimate
- $r(t)_m$ is the range measurement inferred by round-trip-time
- $r(t)_e$ and $s(t)_e$ are range and speed estimates, respectively
- $\alpha = 0.67$ and $\beta = 0.25$

α and β are control gains that are determined experimentally. The range predicting function $r(t)_p$ is defined as:

$$r(t + T_n)_p = r(t)_e + [s(t)_e \times T_n]$$

where T_n is the difference in time between the next measurement and the current measurement.

Based on this estimator, a window of plausible range estimates is placed on the target. If the range estimate falls outside of this window for six or more iterations, then the track is assumed to be erroneous, and the target is assumed to be coasting at its last known velocity and altitude. The size of this window increases as the target gets closer to the reference aircraft [34].

Tracking

Like the altitude and range acquisitions, interrogations with UF 0 continue to occur provided that the altitude and range estimates fall within expected windows. For each plane that replies with DF 0, TCAS updates the tracking entry for that plane with new range, altitude, and velocity estimates. Each plane will maintain its surveillance in this manner for as long

Conditions									***Performance***
Quadrant						***Maximum Traffic Density***		***Maximum number of intruding ACAS within 56 km (30 NM)***	***Probability of success***
Forward		***Side***		***Back***					
Maximum Closing Speed									
m/s	*kt*	*m/s*	*kt*	*m/s*	*kt*	*aircraft/ sq. km*	*aircraft/ sq. NM*		
260	500	150	300	93	180	0.087	0.3	30	0.9
620	1 200	390	750	220	430	0.017	0.06	30	0.9

Table 2.1: TCAS's maximum closing speeds and traffic densities [34].

as planes are within range. TCAS should nominally survey aircraft within a 26 km range and should maintain up to 30 simultaneous tracks in this matter [34].

Interrogation rates can vary depending on traffic density, intruder time until CPA, etc. The default rate of interrogation is once per second, and this rate is defined as the "surveillance update period." The track for an intruder is only dropped if a DF 0 reply is not found in ten surveillance update periods. As traffic density increases, the reliable surveillance range of 26 km becomes smaller. However, TCAS should always be able to track targets in a maximum traffic density of 0.017 aircraft per square km. Table 2.1 defines the maximum densities and closing speeds that TCAS can track. If TCAS's capacity of tracked aircraft is exceeded, intruders should be deleted in the order of decreasing range [34].

Threat Declaration

In order for an intruder to be declared a threat, it must pass a **range test** and a **altitude test** in that order, as shown in Figure 2.4. An intruder can also be declared a threat if the range test passes and the intruder's TCAS transmits a Resolution Advisory Complement (RAC). In order for these tests to pass, the tracks must remain established for the intruding aircraft, and the estimated position of the aircraft must be in threat range for the given SL. TCAS can also filter out certain encounters to reduce "nuisance RAs", but these types of encounters are omitted for this thesis [11].

The range test passes when the intruder is within threat range or is projected to be within threat range. Range is compared with the *Tau* and *DMOD* thresholds in Table A.1. *Tau* (τ) is a variable threshold value that is compared with an intruder's time-to-go until CPA. *DMOD* is a relative distance modifier to account for parallel or slow-closure-rate intruders. The range test passes when either the *Tau* or *DMOD* are violated [11, 15].

Then, the altitude test is run. The tested values in the vertical sense are *ZTHR*, *TVTHR*, and *Tau*. Similar to the range test, the altitude threshold is also a variable, time-to-go until CPA value. While climbing or descending, TCAS will use *Tau* in its altitude test. However, in level flight, TCAS will use the stricter Time to Co-altitude Threshold (*TVTHR*). The altitude test also has a relative distance modifier, *ZTHR*, to account for level or slow-closure rate encounters [11, 15].

Advisory

When the range and altitude tests pass, the intruder is declared a threat, i.e., the reference aircraft has identified that the intruder is in its protected airspace. Generally, a TCAS will first declare a TA against the intruder. Tables A.1 and A.3 indicate that this declaration occurs with at most 48 seconds until CPA and that a "Traffic, traffic" aural announcement is given to the pilots. The threat will be displayed as a yellow icon as in Figure 2.8. If any other intruders did not cause the TA but are located within 6 NM and ±1,200 ft, they are displayed as *proximate traffic* (the blue icons of Figure 2.8).

Then, TCAS will then generate an RA. Recall that an RA must check any received RACs from the intruder before selecting a *sense* and a *strength*. In this encounter, the intruder has not yet transmitted a RAC to the reference aircraft. Therefore, a plausible RA the reference aircraft may announce that it should descend 100 ft to achieve ALIM separation, as indicated by Table A.1. TCAS will select a *downward* sense with a *positive* strength. Table A.2 labels this RA as a `Descend` RA, which elicits a "Descend, descend" announcement to the pilot, as shown in Table A.3.

Now that the reference aircraft has selected its maneuver, it will transmit a special *interrogation* message to the intruder. The first aircraft to transmit this interrogation controls the maneuver of the second aircraft. When both planes release this interrogation simultaneously, the plane with the lower ICAO address takes priority.

The special interrogation message is the RAC, and it is the second message - along with squitters - that breaks the request-response paradigm. When the intruder's Mode S transponder receives the RAC, it does not respond with a reply, and it instead sets registers in the TCAS CPU that suppresses certain RAs. In this example, the reference aircraft's RAC would indicate to the intruder to "Do not pass below" [11, 35].

Finally, the maneuver occurs. The minimum RA duration is 5 seconds. Once this minimum duration is over, TCAS checks that the range test is failing (i.e. the planes are flying away from each other), or TCAS verifies that the horizontal distance between the two planes at CPA is sufficient. The planes are then *clear of conflict* and can return to their normal flight paths unless instructed otherwise by ATC [11, 15].

2.2 Safety-Critical Systems

2.2.1 Definition of Critical and Safety-Critical Systems

Safety-critical systems are a subset systems called *critical systems*. As defined by Sommerville [61], "Critical systems are systems whose failure may lead to injury or loss of life, damage to the environment, unauthorized disclose of information, or serious financial losses." These critical systems can be divided into three different sub-categories:

1. *Safety-critical*: A system whose failure can result in death, injury, or serious environmental damage.
2. *Mission-critical*: A system whose failure results in an entire goal becoming infeasible.
3. *Business-critical*: A system whose failure can result in extremely high costs for a business.

In other words, critical systems have a high cost of failure. Therefore, the type of software engineering that goes into a critical system is much different than a non-critical system. Critical systems developers are "naturally conservative" with their approaches such that, in general, they only employ thoroughly tested methods [61].

Figure 2.15: An IEC-compliant emergency-off button mounted on an industrial machine's console [68].

There are a variety of factors that play a role in critical software development. The largest factors that play a role in development are process engineering, well-tested tools, and regulatory requirements [15]. As a result of these factors, formal methods have been defined for many critical industries like automotive, infrastructure, and medicine to mathematically prove the safety of the software. These methods have proven to be the most cost-effective and efficient tools that critical system engineers have [4, 16, 28, 68].

2.2.2 TCAS With Respect To Safety-Critical Systems

Due to the high safety risk that flight imposes, an aircraft has many on board safety-critical systems. Furthermore, heavy regulations imposed on the industry by the FAA requires frequent law compliance validation. The aviation industry has its own set of standards and formal methods to meet the demand of frequent compliance checking during the design of aircraft systems.

In the aviation industry, these standards and formal methods are referred to as Minimum Operational Standards (MOPS) [59]. A MOPS defines formal methods, validation procedures, compliance guidelines, and more. Different organizations are responsible for maintaining the standards of different aircraft systems. For example, the standards written and maintained by the Radio Technical Commission for Aeronautics (RTCA) cocern all aeronautical communication [14].

The RTCA document, `DO-178B`, defines the MOPS for all safety-critical airborne software. This includes the standards for designing and testing a TCAS system. Therefore, if TCAS is designed to pass the testing procedures of `DO-178B`, then it must be a safety-critical system in function [4, 14, 16, 28].

Intuitively, TCAS must also be a safety-critical system. If TCAS (and its transponder) fail while in operation, the aircraft is put at risk of mid-air collision. This directly affects the safety of on-board passengers by increasing their risk of injury or death.

2.3 Security Basics

Security is an abstract concept; it is difficult to categorize how “secure” a system is. There are many questions to consider in the security of a system. For example, one could analyze the security of a room in various ways. How many entry points are there to the room? How many exits are there? Can the entrances be secured by locks or other means? How robust are those locking methods to picking or to force? It quickly becomes clear that there are innumerable ways to look at simple things from a security perspective.

Therefore, there is a need to define a standardized medium in which to communicate the effectiveness of a system’s security. An example of such a model is called the **confidentiality, integrity, and availability** model. Also called the **CIA triad**, the model is intended to inform the decision making behind information security policies. These three elements - confidentiality, integrity, and availability - are the most crucial elements that form the core of a system’s security [26].

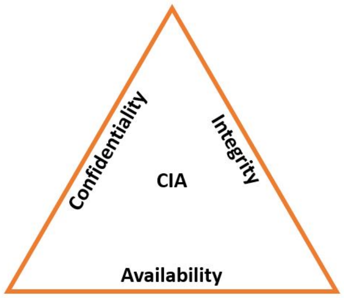


Figure 2.16: The confidentiality, integrity, and availability (CIA) triad.

2.3.1 Confidentiality

Confidentiality and *privacy* are closely related ideas. Confidentiality is the idea that all information is on a *need-to-know* basis. Allowing everyone to access all information on a system is a bad security practice because malicious actors can gain access to sensitive information easily. Therefore, systems that practice confidentiality often follow the principle of *least-privilege*: the concept that users should have the least amount of access to a system's data as possible [42].

However, confidential systems come at the cost of convenience or scale. Some methods scale well to wide-public adoption, such as the encryption and two-factor authentication in everyday Internet usage [23, 39]. However, some systems require more robust methods of confidentiality. For example, the computers with the most sensitive information on them are *air-gapped*; they are either completely disconnected from any network, or they connected to a network that is physically separate from unsecured networks like the public Internet [56].

2.3.2 Integrity

Integrity implies that data in transit cannot be modified by unauthorized people during, for example, a data (confidentiality) breach. In short, data with integrity implies that the data received is trustworthy and accurate [26].

An example of a security vulnerability that exploits integrity is a *man-in-the-middle* attack. In this attack, the bad actor can intercept the data that is in transit from a system. Then, they modify the data in some way, and then this modified payload is sent to the user [63].

Integrity also encompasses data modification that is non-malicious. For example, a *checksum* attached to the payload is a method for determining that the amount of data received is the amount of data sent. Common network transport protocols like TCP and UDP use a checksum to add integrity to data transmission [17].

2.3.3 Availability

Availability is the concept that that reliable access is guaranteed for authorized users. In summary, systems with high availability crash never or infrequently, are repaired immediately, and are frequently maintained. Available systems are never bottle-necked, are always up to date with operating system changes, and have many redundancies to ensure fast recovery [26].

One of the most difficult challenges to availability is denial-of-service attacks (DoS). A DoS attack is when legitimate users cannot access systems because malicious actors use all the resources a system has. This can occur when a target system is flooded with communications so that legitimate communications are effectively blocked. The challenging part about detecting DoS attacks is that they seem like legitimate actions. In wireless communications, jamming or interference could also be considered a form of DoS [18].

2.4 Attack Trees

Section 2.3 makes evident that there is more to a system's security than just a binary *insecure* or *secure*. In fact, the more accurate factors to consider in a system's security are questions like:

- "From whom is this system secure?"
- "How long will this system be secure?"
- "What are the most vulnerable points in this system?"

While Section 2.3 introduces the CIA model for modeling a system's security, it is also necessary to model the attacks that are possible against a system. If a designer can understand every conceivable attack against their system, then they can know where to focus their defense efforts. Also, if they can understand the kind of actor who would attack the system, then the designer can anticipate what the attacker is most likely to do.

One way to model a system's security based on attacks is called an *attack tree*. Attack trees are a graphical representation of an attack. The root of the attack tree is some goal the attacker wishes to achieve. From there, the different ways of achieving the goal are shown in its leaves. In short, an attack tree is simply a series of "What if. . ." questions that allows the designer to conceive how they would carry out an attack. The result is an extensive list of known attack vectors to the system [55].

An example

Assume that a student had information that led them to believe that a grade book which contains the record of a recently failed exam is located in a locked classroom. The student knows this classroom as they carefully studied its layout during the first and only day they attended class. They noted that the door is normally locked with a keypad lock, and there are a number of secured windows which would also lead to the room. This student decided that they are going to attack the security of this room.

First, the attacker must decide their attack goal. In this case, the attack goal is clear: the attacker must enter the room. This goal becomes the root of the attack tree. Then, the attacker thinks about all the ways they could enter the classroom like the door or the window. They then can explore those options as far as they would like until they are satisfied with an attack plan. Figure 2.17 shows the attack tree that the student could come up with.

In general, each leaf is mutually exclusive. However, in cases like the `Eavesdrop` attack, multiple attacks must be successful in order for the remaining goals to be achieved. These

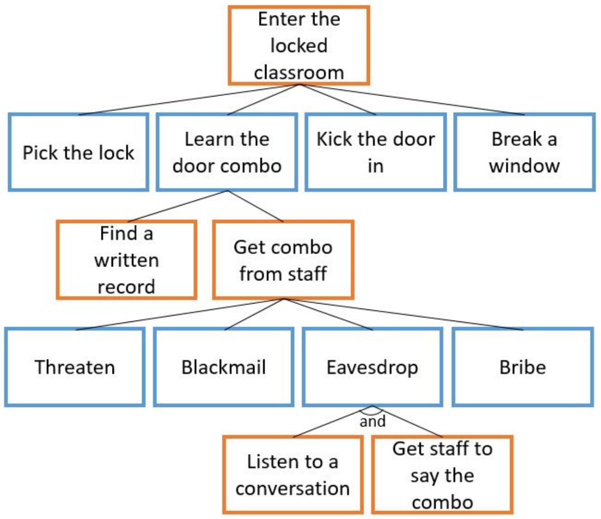


Figure 2.17: Example attack tree against a secured room.

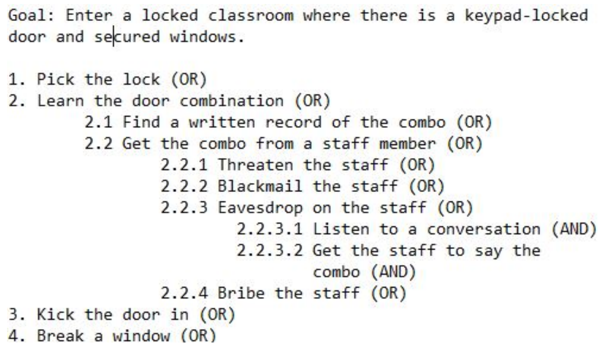
are indicated with an `and` connection.

Then, the attacker can further characterize each leaf on the tree, such as:

- Which attacks are detectable? Which attacks are low-profile?
- Do these attacks require equipment? How specialized is this equipment?
- How much money does this attack require?

The values that quantify these attacks can change over time as methods become cheaper or more information is gathered. Eventually, it becomes clear which points of a system are the most vulnerable. For example, maybe the staff of this institution assumed that students would never bribe employees for a room's key code. The attack tree can also reveal the impact that would happen if an attack occurred. For example, a successful attack could cost the learning institution an employee for negligence and a loss of its academic integrity.

Perhaps the greatest value that attack trees have is their portability. The attack tree of Figure 2.17 is not limited to the classroom the student is attacking; the attack tree can be


```
Goal: Enter a locked classroom where there is a keypad-locked
door and secured windows.

1. Pick the lock (OR)
2. Learn the door combination (OR)
        2.1 Find a written record of the combo (OR)
        2.2 Get the combo from a staff member (OR)
                2.2.1 Threaten the staff (OR)
                2.2.2 Blackmail the staff (OR)
                2.2.3 Eavesdrop on the staff (OR)
                        2.2.3.1 Listen to a conversation (AND)
                        2.2.3.2 Get the staff to say the
                                combo (AND)
                2.2.4 Bribe the staff (OR)
3. Kick the door in (OR)
4. Break a window (OR)
```

Figure 2.18: Example attack tree against a secured room in text form.

applied to most rooms' circumstances: a locked door with a window. In other words, the root node of the attack tree is still valid for similar locations.

In summary, security is not just the product of good design. Security is actually a way of thinking that accounts for all the ways one could try to break a system, and the attack tree is just one way to formalize the approach an attacker would make. As Schneier [55] says, "[Security] is a process. Attack trees form the basis of understanding that process."

2.5 Fault Tree Analysis (FTA)

2.5.1 Description of FTAs

System failures are an inevitability. Nevertheless, they present important learning opportunities. Similar to how humans constantly learn from mistakes, engineers often have to understand how systems fail in order to correct the failures. However, it is beneficial for engineers to reduce the cost of making mistakes. Therefore, there is a demand for creating formal models that can predict how likely an undesired failure state is to occur.

The **Fault Tree Analysis** (FTA) is one such tool for modeling failure. FTAs are created to model failure in very complex systems. Specifically, FTAs can show how a series of events can lead to a larger undesired failure event using probability, Boolean algebra, and logic. FTAs are essential for developing a safety-critical system because those systems are mandated to meet certain safety expectations, as discussed in Section 2.2 [4, 16, 28, 31].

An FTA is a deductive, top-to-bottom approach to modeling a failure. Similar to the attack tree in Section 2.4, the root of an FTA starts from a *goal* (i.e. starting from the desired failure to investigate) and then deduces the specific components that could cause the failure. From a quantitative perspective, the resulting FTA makes calculating a probability of failure simpler provided that the stated failure event is reasonable to model.

Continuing the example from Section 2.4, Figure 2.19 shows an FTA for a closed door. The hypothetical closed door has a keypad lock with a deadbolt. Because it is a simplified FTA, leaf nodes are not explored further than the first level. A, B, C, and D are the basic probabilities for each respective basic event.

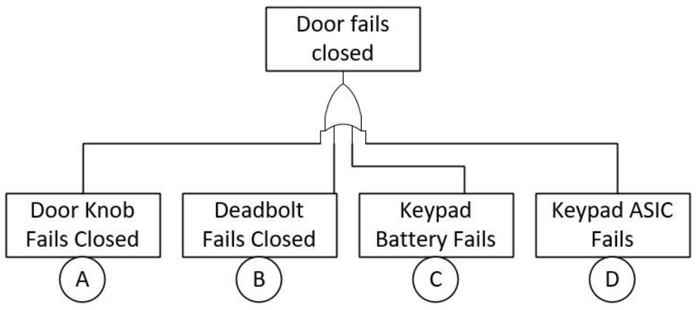


Figure 2.19: Simple qualitative FTA modeling the failure of a closed keypad-lock door.

There are many ways that an FTA can inform design decisions. FTAs can be used in a root cause analysis where the components of an undesired event are identified. FTAs can also be used to quantify the risk associated with a failure. These calculations enable objective measurements of failure when a design changes. Finally, FTAs can demonstrate compliance with regulatory requirements and standards.

In summary, the FTA is a powerful tool for modeling undesired events in a complex system.

However, it is also important to note that FTAs are models of how failures are perceived by the engineers that created the system. An FTA will not provide a true-fidelity model of how something fails, but they are powerful tools when used properly [31].

Chapter 3

FTA from the Adversarial Perspective

The FTA is a tool engineers can use to inform decision making about design revision with respect to some undesired event [31]. Many safety-critical systems require an FTA to demonstrate compliance with regulations. TCAS, as a system that runs safety-critical aeronautical software, is also required to demonstrate robustness to failure through compliance with the MOPS *DO-178B* [4, 14, 16, 28].

As is discussed in Section 2.1, The MITRE Corporation partially developed TCAS. MITRE designed much of the TCAS logic; therefore, they were the most familiar with the system to create its FTA. Part of their effort resulted a study on TCAS to prove its impact on aeronautical safety. The final report, *System Safety Study of Minimum TCAS II*, was created in December, 1983 [40].

The TCAS safety study places emphasis on the limitations and failures of TCAS. In particular, the study explores how transponder equipage, altimetry errors, and sudden maneuvers affect the performance of TCAS. Then, the study addresses the failure mechanisms of surveillance, bit errors, avionics failures, etc. Finally, different scenarios involving pilot errors are also taken into account.

The study defines its undesired (or top) event as the result of faults which could cause two planes to result in a Near Mid-air Collision (NMAC). NMAC is defined as a scenario where two aircraft have a vertical separation of 100 ft or less and a horizontal separation of 500 ft or less. The study defines a term called **risk ratio**: a factor that describes the risk of encountering NMAC when equipped with a TCAS relative to the risk of encountering NMAC without a TCAS. In other words, a risk ratio of 1 would mean that the probability of experiencing NMAC is unchanged with the equipage of TCAS, a risk ratio less than 1 would mean the aircraft is more safe with TCAS equipped, and a risk ratio greater than 1 would indicate the aircraft is less safe with TCAS equipped.

Although the results of this study quantify a risk ratio less than 1, the details of this safety study make evident that adversarial encounters are not considered in the study's FTA. In the remainder of this chapter, *System Safety Study of Minimum TCAS II* is examined from an adversarial perspective.

Section 3.1 first summarizes key components of the safety study, and it highlights the components which can be modified when an adversary is considered. Then, Section 3.2 discusses reverse engineering the calculations used in the safety study into a computer model. Then, Section 3.3 shows how the computer model can assist an attacker in making inferences on how a successful attack will affect the safety of TCAS. Finally, Section 3.4 analyzes TCAS against the CIA triad and shows how TCAS has weak security guarantees.

3.1 FTA Description

In order to quantify a **risk ratio** of relative probability of NMAC, an estimate of the absolute probability of encountering an NMAC is required. Using flight data as well as incident reports from the FAA, some of the key discoveries of TCAS are:

- 70% of NMAC incidents occurred in bright daylight conditions. The study assumes visual acquisition is impossible for all other conditions.

- 16% of NMAC incidents happened in Instrument Meteorological Conditions (IMC) [13].
- The distribution of relative altitude amongst all CPA encounters is approximately uniform.
- **The absolute probability of encountering an NMAC is estimated as 1 NMAC per 100,000 flight hours.**

Some of the report findings would be unrealistic to model today. For example, the safety study found that of all NMAC encounters, 61% of encounters involved aircraft with altitude-reporting transponders. However, since this study predates the TCAS mandate discussed in Section 2.1, it can be speculated that the number of incidents involving altitude-reporting transponders would be closer to the total number of transponders in service: 92%.

After exploring the TCAS environment, an FTA can relate the probability NMAC to relevant TCAS failures. NMAC failures are qualitatively categorized into two mutually-exclusive failure modes. These modes are defined as:

1. **Unresolved NMAC**: Two planes are already on a collision course, and the pilots fail to maneuver the planes in a way that avoids NMAC.
2. **Induced NMAC**: Two planes that were not previously on course towards NMAC are maneuvered into a NMAC.

A graphical representation of the top-level FTA is shown in Figure 3.1. The component of particular interest is the **induced** component of NMAC because the induced component is antithetical to the design goals of TCAS. As an adversary, demonstrating control over the induced NMAC component would imply that an attacker could arbitrarily redirect two nearby planes into each other.

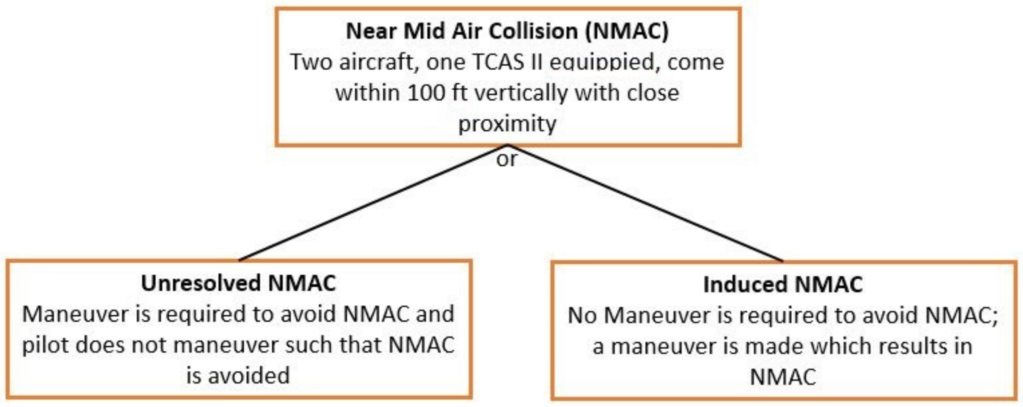

Near Mid Air Collision (NMAC)
Two aircraft, one TCAS II equipped, come within 100 ft vertically with close proximity
or
Unresolved NMAC
Maneuver is required to avoid NMAC and pilot does not maneuver such that NMAC is avoided
Induced NMAC
No Maneuver is required to avoid NMAC; a maneuver is made which results in NMAC

Figure 3.1: First and second levels of the TCAS FTA [40].

The study further explores the limitations and failure mechanisms of TCAS. Table 3.1 describes these basic probabilities and attaches an identifier to each condition. Some of these basic probabilities could be modified when considering the adversary. For example, if an attacker could force TCAS into releasing an RA without any visible aircraft nearby, events `n` and `o` would become 1 since there is no aircraft for a pilot to visually acquire.

Another important failure component is the pilot. TCAS is dependent on a well-trained pilot properly maneuvering on TCAS instructions. Therefore, it is worthwhile to also associate plausible mistakes the pilot may make with the FTA, thereby converting the original FTA into a fuzzy-logic FTA [65]. Table 3.2 summarizes five human failures that can arise from visual, RA, and TA information [40].

With basic events and human failures defined, a formulation for the top event is possible. However, the full fault tree shown in [40] is very complicated, and certain sub-branches can be simplified if some nominal operating conditions are assumed. After the formulation is made, the assumptions made in the nominal conditions can be adjusted in a sensitivity analysis. These nominal operating conditions are:

1. Pilots will always avoid planes they can see.

Item	**Condition**	**Probability**
a	Instrument Meteorological Conditions	0.16
b	Bright Daylight Conditions	0.7
c	Encountering General Aviation (GA) Aircraft	0.79
d	Intruder is Transponder Equipped	0.92
e	Intruder is Mode C Transponder Equipped	0.61
f	Risk Ratio for GA Altimetry	0.317
g	GA Altimetry Error: Unresolved Component	0.0143
h	GA Altimetry Error: Induced Component	0.0174
i	Risk Ratio for Maneuvering Intruder	0.027
j	Probability of No Track at Time of TA	0.06
k	Probability of No Track at Time of RA	0.03
l	Risk Ratio for "Stuck C-Bit"	0.002
m	Risk Ratio for Equipment Failure	0.0001
n	Not Visual Acquisition in Bright Daylight by 15 seconds before CPA	0.17
o	Not Visual Acquisition in Bright Daylight by time of RA	0.35

Table 3.1: Basic events and their probabilities used in the original TCAS II safety study [40].

2. If there is no opportunity to visually acquire the threat, the pilot will follow the RA.

3. Visual acquisition is only possible in bright daylight.

4. The intruder is not TCAS-equipped.

5. Bad maneuvers are not made by the TCAS pilot.

Now, a formulation for the two components are made and combined into the top event.

Unresolved Component

The unresolved component of all NMACs is comprised of nine failure modes. Each of these components are summed as the total unresolved component. Some events have human factors

Label	**Full Name**	**Description**
VNA	Visual Acquisition; No Avoidance	The pilot makes visual acquisition but does not avoid the NMAC.
VMIR	Visual Acquisition; Maneuvers on Incorrect RA	Although the pilot visually acquires the aircraft, their RA chooses an incorrect maneuver, and the pilot chooses to do the incorrect maneuver.
RNF	RA; Not Followed	Pilot does not follow their RA.
TNA	TA; No Avoidance	Based on the TA alone, the pilot disregards the RA and does not avoid NMAC.
TI	TA; Induces	The pilot induces an NMAC by maneuvering on a TA.

Table 3.2: Plausible human errors as variable factors in the TCAS FTA [40].

associated with their probabilities. Each human factor is represented as a variable, where the variable corresponds to the labels outlined in Table 3.2.

The failure rate for the unresolved component is defined as:

$$P_u = 0.413 + 0.259(VNA + TNA) + 0.327(RNF) + 0.0008(VMIR)$$

Induced Component

The induced component is only comprised of five failure modes. Their sum results in the total probability of the induced component. Some events that make up the induced component are also dependent on human failures.

The failure rate for the induced component is defined as:

$$P_i = 0.11 + 0.014(VMIR) + 0.59(TI)$$

Top Event

The top event is simply the sum of the two major components because the induced component is mutually exclusive from the unresolved component. This top event probability, $P = P_u + P_i$, is therefore fully expressed as:

$$P_n = 0.424 + 0.259(VNA + TNA) + 0.327(RNF) + 0.0008(VMIR) + 0.014(VMIR) + 0.59(TI)$$

The final P function is recreated in the reverse engineered model of the FTA sensitivity analysis, Section 3.2. P_n defines the risk ratio of NMAC risk with TCAS equipped versus having no TCAS. The derivation the relative probabilities and their related human factors used in each components' functions are explored in depth in Section 3.2 as well.

3.2 Reverse Engineering the FTA Sensitivity Analysis

In order to properly create a model of the quantified TCAS FTA, its derivations must be understood. Then, the failure probabilities can be calculated at run-time in a computer model, which only requires the user to adjust the inputs and observe the effects made on the TCAS risk ratio.

The model's correctness can be verified by following the sensitivity analyses of Chapter 8 [40] and comparing the percent error between the model's results and the original study's results. Some rounding error is tolerable between the presented model and the model of the study because this study predates standardized floating point arithmetic [7].

In summary, the goals of recreating the FTA are:

Item	**Description**	**Function From Basic Probabilities**
A	No TA is displayed	$1 - e(1 - j)$
B	Encounter is with non-altitude-reporting transponder	$1 - e$
C	Surveillance fails to acquire aircraft by TA	j
D	Inadequate Visual Conditions	$1 - b$
E	IMC	a
F	Pilot does not see threat in time to avoid NMAC	n
G	Pilot does not see threat prior to RA	o
H	No RA is displayed	$1 - e(1 - k)$
I	Surveillance fails to acquire aircraft by RA	k
J	Inadequate RA displayed	$c \times g$

Table 3.3: Failure probabilities used in the quantization of the **unresolved** component of NMAC [40].

1. Locate or derive the operations performed on the basic probabilities. Recreate the study's mathematics with as much fidelity as possible.
2. Verify the model's correctness by performing the sensitivity analysis done in Chapter 8 of the study [40].
3. Create the model in such a way that only adjustments to the input are necessary for observing effects on the TCAS risk ratio.

To begin, the unresolved component is analyzed first. The unresolved component contains the "principal failure mode for TCAS" of which is claimed to have been caused by lack of altitude-reporting transponder equipage. This principal failure mode accounts for one of nine total drivers of unresolved NMACs.

A common set of basic failure probabilities are created from the basic probabilities of Table 3.1. These failures are outlined in Table 3.3 where each item is described by a function of basic probabilities. For example, the failure item C is the result of the equation: $C = j$.

A similar approach is taken to determine the effect of the basic probabilities on the induced component of NMAC. These probabilities are shown in Table 3.4.

Unlike the unresolved component, the induced component of NMAC uses constant estimates P and Q. Both of these estimates are created from the observation that encounter altitudes are approximately uniformly distributed. Using the uniformly distributed encounter geometry assumption, it can be assumed that P is approximately 10 **proximate encounters** for every NMAC because a **proximate encounter** is an encounter that occurs in approximately 10 times the altitude range of NMAC. Similarly, using a conservative assumption that a pilot makes a **random** maneuver on a TA, Q is the probability that such a maneuver induces an NMAC given that two aircraft are in a proximate encounter [40].

Tables 3.1, 3.2, 3.3, and 3.4 define the entire vernacular in which to understand the TCAS failure events. Therefore, the induced and unresolved components can finally be defined in terms of the baseline components. The unresolved components' quantization functions as well as their descriptions are outlined in Table 3.5. Likewise, the induced components' quantization functions and descriptions are aggregated into Table 3.6.

Each table can now be translated into a computer model. The FTA model is realized as a spreadsheet with modifiable input probabilities and check boxes to enable or disable components as desired. The risk ratio contributed for each failure and their sub-components of Tables 3.5 and 3.6 are exposed for debugging. After calculating a risk ratio for each failure component, their sum is taken until the top-level event is reached. The output of the model is represented as three risk ratios: the total, the unresolved component, and the induced

Item	**Description**	**Function From Basic Probabilities**
K	Altimetry error causes an inductive RA	$c \times h \times e(1-k)$
L	Intruder suddenly maneuvers; causes an inductive RA	$i \times e(1-k)$
M	C-bit errors cause an inductive RA	$l \times e(1-k)$
N	No TA was displayed on-time given the existence of an RA	$1 - \frac{(1-k)}{(1-j)}$
O	TA was displayed on-time given the existence of an RA	$\frac{(1-k)}{(1-j)}$
P	Ratio of proximate encounters to NMACs	10
Q	Probability that a pilot's chosen maneuver induces an NMAC if maneuver occurs on TA alone	0.1

Table 3.4: Failure probabilities used in the quantization of the **induced** component of NMAC [40].

component of risk ratio.

To verify the model, the sensitivity analysis is performed. Then, a percent error is calculated for each case considered in the model versus the results reported in the safety study. As an example, a detailed examination of the nominal case without any human error as described in Section 3.1 is provided. Figure 3.2 shows the input to the model with each basic probability set to their nominal conditions.

Calculation of the nominal input is performed, and the output from the model is reported in Figure 3.3. For this case, the top level risk ratio is **0.422**. This is slightly different than the result of the safety study which reports the nominal risk ratio as **0.424**. However, with such a low percent error (approximately **0.47%**) for the top level event, the recreated model performs well within reasonable expectations. In general, the only significant sources of error come from the induced component. These large errors are acceptable though because they

Failure Description	**Failure Quantization**
Neither TA nor RA is received	$A \times \frac{H}{A}$
Bright daylight conditions in which: TA is received; visual acquisition does not occur; inadequate RA is received	$(1 - A) \times (1 - D) \times G \times J \times \frac{F}{G}$
TA is received; visual acquisition is not possible; inadequate RA is received	$(1 - A) \times D \times J$
TA is not received prior to RA; inadequate RA is received	$A \times (1 - \frac{H}{A}) \times J$
Bright daylight conditions in which: TA is received; visual acquisition does occur before RA; pilot fails to avoid the NMAC	$(1 - A) \times (1 - D) \times (1 - G) \times ((VNA) + (TNA))$
Bright daylight conditions in which: TA is received; visual acquisition does not occur; RA is received; pilot does not follow RA	$(1 - A) \times (1 - D) \times G \times (1 - J) \times (RNF)$
Bright daylight conditions in which: TA is received; visual acquisition occurs after RA; inadequate RA is received; pilot follows inadequate RA	$(1 - A) \times (1 - D) \times G \times J \times \frac{F}{G} \times (VMIR)$
TA is received; visual acquisition is not possible; pilot does not follow RA	$(1 - A) \times D \times (1 - J) \times (RNF)$
TA is not received; an RA is received; pilot does not follow RA	$A \times (1 - \frac{H}{A}) \times (1 - J) \times (RNF)$

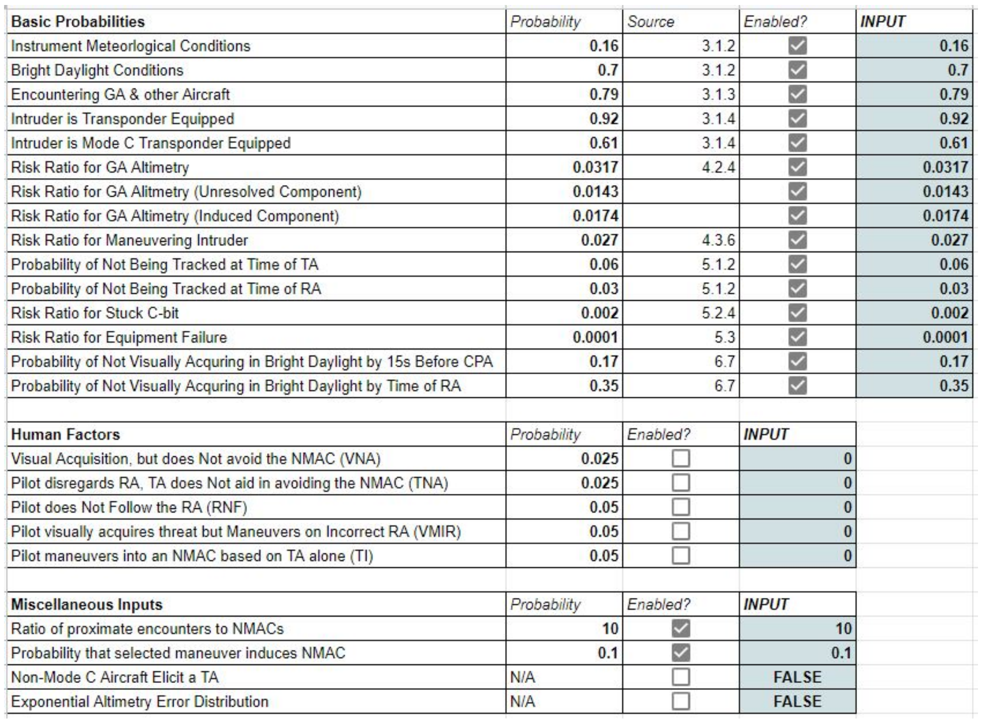
Table 3.5: Unresolved component quantization functions using items from Table 3.3. The total unresolved component is the sum of these parts.

have little effect to the error in the top component. Table 3.7 compares a selection of the results of the model and the safety study.

With these results, it is assumed that the model created from the safety study is an adequate model of failure for TCAS. With this assumption, better-informed decisions can be made about TCAS faults. An attacker would primarily be focused on using this failure sensitivity

Failure Description	**Failure Quantization**
Visual acquisition occurs; pilot maneuvers on incorrect RA anyway	$(K + L + M) \times O \times (1 - D) \times (1 - F) \times (VMIR)$
TA is received and visual acquisition is possible; visual acquisition does not occur	$(K + L + M) \times O \times (1 - D) \times F$
TA is received but visual acquisition is not possible	$(K + L + M) \times O \times D$
TA is not received	$(K + L + M) \times N$
Pilot induces NMAC from TA alone	$P \times (1 - B) \times (1 - I) \times Q \times (TI)$

Table 3.6: Induced component quantization functions using items from Tables 3.3 and 3.4. The total induced component is the sum of these parts.



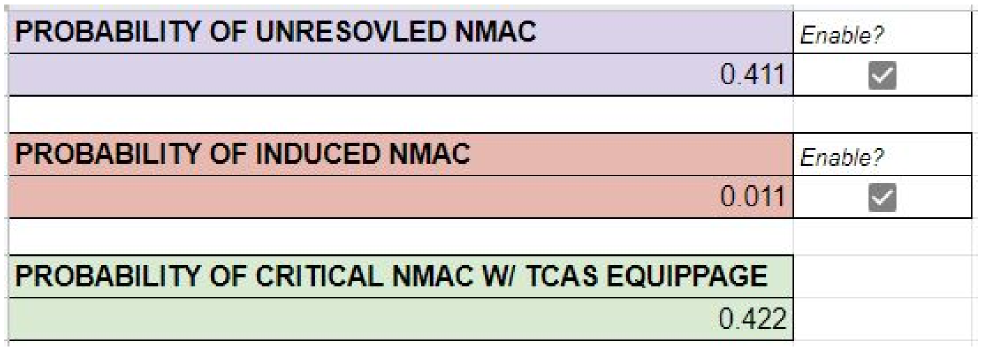
Basic Probabilities	*Probability*	*Source*	*Enabled?*	***INPUT***
Instrument Meteorlogical Conditions	**0.16**	3.1.2	✓	**0.16**
Bright Daylight Conditions	**0.7**	3.1.2	✓	**0.7**
Encountering GA & other Aircraft	**0.79**	3.1.3	✓	**0.79**
Intruder is Transponder Equipped	**0.92**	3.1.4	✓	**0.92**
Intruder is Mode C Transponder Equipped	**0.61**	3.1.4	✓	**0.61**
Risk Ratio for GA Altimetry	**0.0317**	4.2.4	✓	**0.0317**
Risk Ratio for GA Alitmetry (Unresolved Component)	**0.0143**		✓	**0.0143**
Risk Ratio for GA Altimetry (Induced Component)	**0.0174**		✓	**0.0174**
Risk Ratio for Maneuvering Intruder	**0.027**	4.3.6	✓	**0.027**
Probability of Not Being Tracked at Time of TA	**0.06**	5.1.2	✓	**0.06**
Probability of Not Being Tracked at Time of RA	**0.03**	5.1.2	✓	**0.03**
Risk Ratio for Stuck C-bit	**0.002**	5.2.4	✓	**0.002**
Risk Ratio for Equipment Failure	**0.0001**	5.3	✓	**0.0001**
Probability of Not Visually Acquring in Bright Daylight by 15s Before CPA	**0.17**	6.7	✓	**0.17**
Probability of Not Visually Acquring in Bright Daylight by Time of RA	**0.35**	6.7	✓	**0.35**

Human Factors	*Probability*	*Enabled?*	***INPUT***
Visual Acquisition, but does Not avoid the NMAC (VNA)	**0.025**	☐	**0**
Pilot disregards RA, TA does Not aid in avoiding the NMAC (TNA)	**0.025**	☐	**0**
Pilot does Not Follow the RA (RNF)	**0.05**	☐	**0**
Pilot visually acquires threat but Maneuvers on Incorrect RA (VMIR)	**0.05**	☐	**0**
Pilot maneuvers into an NMAC based on TA alone (TI)	**0.05**	☐	**0**

Miscellaneous Inputs	*Probability*	*Enabled?*	***INPUT***
Ratio of proximate encounters to NMACs	**10**	✓	**10**
Probability that selected maneuver induces NMAC	**0.1**	✓	**0.1**
Non-Mode C Aircraft Elicit a TA	N/A	☐	**FALSE**
Exponential Altimetry Error Distribution	N/A	☐	**FALSE**

Figure 3.2: Input to the recreated TCAS sensitivity analysis model for the nominal case [40].

model to inform their attack decisions. Therefore, Section 3.3 explores this model's usefulness as a source of reconnaissance against TCAS.



PROBABILITY OF UNRESOVLED NMAC	Enable?
0.411	✓
PROBABILITY OF INDUCED NMAC	Enable?
0.011	✓
PROBABILITY OF CRITICAL NMAC W/ TCAS EQUIPPAGE	
0.422	

Figure 3.3: Output of the recreated TCAS sensitivity analysis model for the nominal case [40].

Cases	**Probability of...**	**% error**		
	Top Event	*Top Event*	*Unresolved*	*Induced*
Nominal case	0.422	0.47%	0.48%	0.00%
If 100% of aircraft had altitude-reporting transponders	0.053	0.00%	0.00%	0.00%
If surveillance failure rate was doubled	0.44	0.23%	0.23%	0.00%
If GA altimetry error was exponentially distributed	0.43	0.23%	0.38%	6.67%
If a sudden intruder maneuver was 50% more likely	0.426	0.23%	0.48%	7.14%
If human factors fail with 0.05 probability	0.484	0.00%	0.00%	0.00%

Table 3.7: Selected results of the recreated TCAS sensitivity model and their error from the safety study's results [40].

3.3 Modeling Failures From Attacks

An adversary to TCAS would use the FTA to determine how effective a hypothetical attack would be. In general, the attacker can assume that their attack is guaranteed to work, and they would deduce how an attack would affect the inputs of to the fault model. An attacker is equally as interested in the assumptions made in creating the model as much as they are interested in what **is not** assumed in the model. Therefore, it is valuable to explore this safety study from an adversarial perspective to figure out which attacks could have the most impact on safety.

First, a **threat model** must be defined. The **threat model** relays the attacker's goals capabilities. When defining the threat model, it is generally best to analyze a system with the weakest possible attacker while maintaining the strongest defenses possible. For example, the attacker can revisit the simple encounter geometry shown in the example encounter of Section 2.1. The attacker's goal is to make the reference aircraft's altitude appear to be "stuck" at a non-proximate encounter level. Then, the attacker can have the following capabilities:

- The attacker knows the ICAO address of the reference aircraft.
- The attacker can selectively jam any Mode S reply messages from the reference aircraft.
- The attacker can spoof a Mode S reply message using the reference aircraft's ICAO address.
- The attacker's transmitter can appear closer than it actually is and can appear like a plane in motion.

With the threat model defined, the attacker can adjust the inputs to the FTA model to suit their needs. There are a few adjustments the attacker can make if their goal is to make an aircraft's altitude appear "stuck" to induce an NMAC. These adjustments are:

- There are probably more transponders in service today than in 1983, and there are probably more altitude-reporting transponders than non-reporting ones.
- General Aviation (GA) planes' altimetry error is more realistically represented as a heavy-tailed exponential distribution than a Gaussian distribution [40].

- The probabilities associated with visual acquisition of the intruder become 0 because the attacker is adequately jamming the true plane's transponder. In other words, items `n` and `o` in Table 3.1 can become 1.

- A stuck altitude can be properly represented as the probability of a "stuck C-bit" becoming 1 because C-bits are related to how altitude is encoded in a Mode-C transponder's reply [35].

When these adjustments are made, the sensitivity model results in a total risk-ratio of **1.166** with an induced component of **1.021**. In other words, under these adversarial assumptions, the attacker would have made the TCAS-equipped aircraft a greater safety risk than its non-equipped counterparts. Figure 3.4 shows how the probability of a stuck c-bit affects the top-level risk ratio.

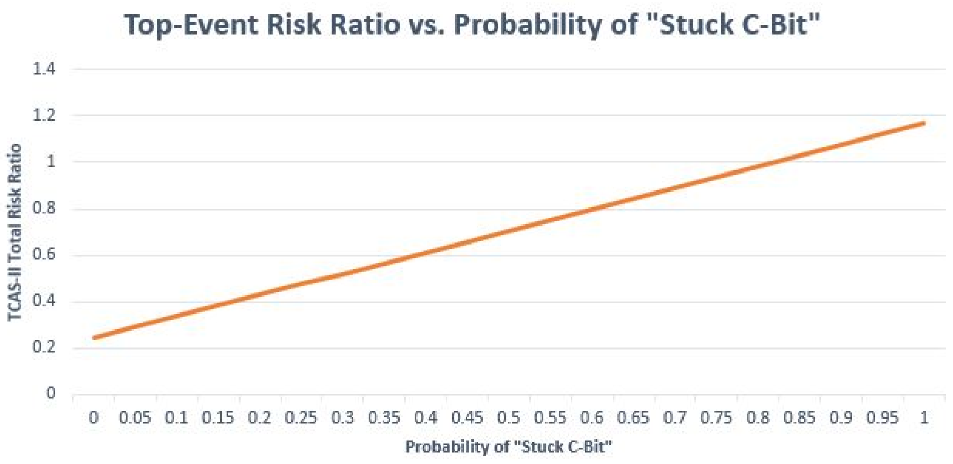


Figure 3.4: Probability of a stuck c-bit and its effect on the TCAS risk ratio under an assumed threat model.

Granted, while there are heavy assumptions in place about the attacker's capabilities, this section is intended to simply demonstrate how the FTA can be observed in an rudimentary adversarial perspective. A more thorough approach would be to recreate the FTA with adversaries in mind and to add adversarial affects to the bottom-level, basic events. That way, a clear relationship between known-attacks and system faults can be illustrated, and the attacks' effects can be more accurately quantified.

3.4 FTA + CIA Triad Comparison

In this section, a brief analysis of TCAS with respect to the CIA triad is performed to qualify TCAS's security guarantees. This analysis will provide insights into TCAS's confidentiality, integrity, and availability, and it will reveal where TCAS may be the most vulnerable to attack.

Confidentiality

Confidentiality is the concept that all data in some information system is on a need-to-know basis. Confidential systems follow the "principle of least-privilege" in general where only authorized users can access privileged information. There are many ways to improve a system's confidentiality such as password-protected accounts, two-factor authentication, and cryptographic masking of data.

However, TCAS provides no confidentiality measures in its operation. Data is broadcast in the open air in an unsecured fashion. While this provides some user convenience in the form of tools like flight tracking [19, 48], an attacker can take advantage of this unsecured network in many ways. For example, since there is no form of authentication in TCAS exchanges, any Mode S transponder trusts any properly formatted message as if it was sent by TCAS.

Chapter 5 exploits the fact that any entity - malicious or not - can fully participate in the network. This design is entirely antithetical to confidentiality, and TCAS in its current state would have difficulties with integrating confidentiality mechanisms into its communications due to its restrictive data throughput and real-time requirements.

Integrity

Integrity is the concept that data is trustworthy and accurate. While TCAS transactions do not contain particularly interesting data, they are nevertheless easily intercepted because of their transmission over an unsecured channel.

Mode S messages do contain some form of integrity check. This check is the Mode S CRC calculation which simultaneously encodes parity and ICAO address in the same field [30]. However, this form of integrity check is only designed for non-malicious transmission errors like bit-flips. As observed in TCAS's lack of confidentiality, there are no mechanisms that could prevent *man-in-the-middle* like manipulation because an attacker could receive a Mode S message, modify its data *and* its CRC, and transmit the modified payload again with greater power than the original. Therefore, while TCAS does provide integrity checking, it fails to perform under adversarial circumstances.

Availability

Availability is the concept of data being readily available. In a safety-critical system like TCAS, unavailable transponder replies are unacceptable in collision course scenarios. And, while the probability of TCAS dropping a track in a NMAC scenario is low [40], TCAS still has operational limits (Table 2.1) which could be exploited in an adversarial context.

For example, a TCAS could be too busy processing **false tracks** to interrogate real transponders. The safety study on TCAS [40] assumes that false tracks are not possible in Mode S encounters "because of the discrete addressed interrogations." However, an SDR could craft an arbitrary amount of “trustworthy” signals. If enough false tracks are generated to exceed the limits posed in Table 2.1, all real tracks would be dropped. Therefore, an attack on TCAS availability - a DoS - is entirely possible.

In summary, TCAS has very few guarantees on confidentiality, integrity, or availability; this is expected because TCAS was never designed for non-cooperation. Now that high-level security attributes of TCAS are contextualized, attacks can be proposed in Chapter 5.

Chapter 4

TCAS Attack Tree

In Section 2.4, the purpose of attack trees in general is explored. So, an attack tree for TCAS would be a helpful representation of attacks against TCAS. The TCAS attack tree is first compared to the TCAS fault tree in Chapter 3. Then, an attack is chosen and justified as a reasonable attack. Finally, an examination of trade-offs between a selection of alternatives and the chosen attack is presented.

First, an intended attack outcome must be defined in order to qualify the root of the tree. The attacker's goal is to create an attack which causes NMAC since NMAC is shown to be a sufficient failure event of TCAS. From this goal, a deductive approach is taken to identify the basic failures that an attack could cause.

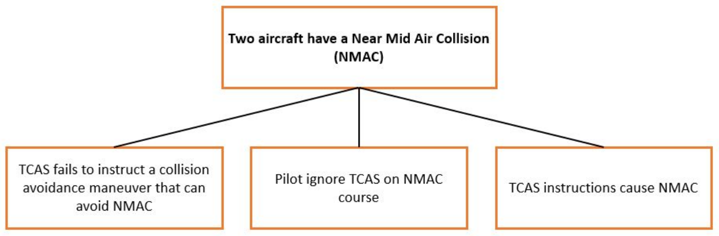

Two aircraft have a Near Mid Air Collision (NMAC)
TCAS fails to instruct a collision avoidance maneuver that can avoid NMAC
Pilot ignore TCAS on NMAC course
TCAS instructions cause NMAC

Figure 4.1: The root and major leaf nodes of the TCAS attack tree

The attack tree of this thesis shares two major components with the TCAS fault tree; these major components identified are shown as the top of the the attack tree in Figure 4.1. The left event represents **unresolved NMAC** attacks, and the right event represents the **induced NMAC** attacks. The central event accounts for any attacks on people like social engineering attacks [64]. Since social engineering attacks are outside the scope of this thesis, the former two major branches are explored further.

Unresolved NMAC Attacks

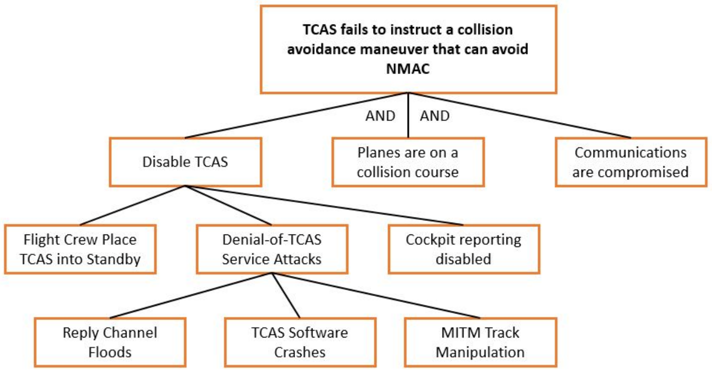

TCAS fails to instruct a collision avoidance maneuver that can avoid NMAC
AND
AND
Disable TCAS
Planes are on a collision course
Communications are compromised
Flight Crew Place TCAS into Standby
Denial-of-TCAS Service Attacks
Cockpit reporting disabled
Reply Channel Floods
TCAS Software Crashes
MITM Track Manipulation

Figure 4.2: A simplified version of the branch on **unresolved** NMAC attacks in the TCAS attack tree.

The definition of **unresolved** NMAC attacks is similar to the the **unresolved NMAC** of Chapter 3. An unresolved attack attempts to increase the probability that two aircraft on a collision path do not maneuver in time to avoid NMAC. To fit the scope of this thesis, the simplified attack tree diagrams emphasize communications attacks.

Figure 4.2 shows a section of the unresolved NMAC component. In general, attacks that leave an NMAC unresolved are idealized to interfere with the TCAS communications in some way. Specifically, attacks should focus on denying the 1090 MHz frequency band because reply messages on the 1090 MHz channel are necessary for maintaining a track against an intruder. While jamming of the 1090 MHz channel is a simple example of a DoS attack that achieves the goal, arbitrary jamming attacks are not a noteworthy threat against TCAS because they are easily detected or thwarted [25].

Instead, there are more effective approaches to garbling the 1090 MHz channel. These types of attacks are classified under the **Reply Channel Flood** category of attacks in Figure 4.2. Two examples of a reply channel flood are:

- **All-Call Flood**: The attacker abuses a special Mode S interrogation - the All-Call interrogation[35] - on the 1030 MHz channel to force all Mode S transponders that receive it to reply with its ICAO address. This attack is more effective in higher traffic densities.

- **Squitter Flood**: The attacker can pose as an aircraft by transmitting various All-Call reply messages on the 1090 MHz channel. This attack takes advantage of the fact that TCAS passively monitors for nearby All-Call replies to reduce congestion of the 1030 MHz channel. The attacker can utilize any address in the size 2^{24} ICAO address space to force nearby TCAS into continuously repeating the acquisition phase of intruder tracking without repeating false addresses (Section 2.1) [34, 35].

The main drawback to unresolved NMAC attacks is the reliance on an existing NMAC scenario. As shown in Chapter 3, flight data indicates that a plane would experience an NMAC approximately once every 100,000 flight hours on average [40]. From an adversarial perspective, it would be best if the adversary could cause NMACs arbitrarily. A more powerful attack grants the full control of inducing NMAC at the attacker's will.

Induced NMAC Attacks

The definition of an **induced** NMAC attack is similar to the **induced NMAC** from Chapter 3. Unlike the unresolved attacks, induced NMAC attacks are reliant on abusing the Mode S message protocol to fool a TCAS into generating RAs that, if followed, would induce NMAC.

A section of the induced NMAC attack is shown in Figure 4.3. Attacks that induce NMAC

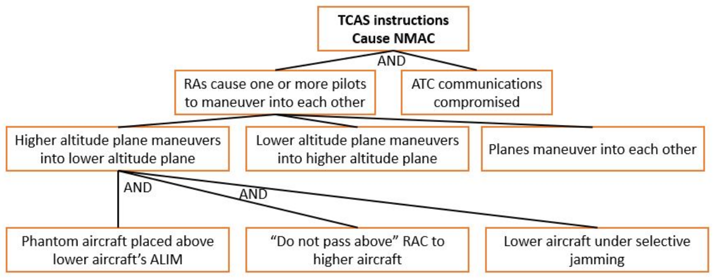

TCAS instructions Cause NMAC
AND
RAs cause one or more pilots to maneuver into each other
ATC communications compromised
Higher altitude plane maneuvers into lower altitude plane
Lower altitude plane maneuvers into higher altitude plane
Planes maneuver into each other
AND
AND
Phantom aircraft placed above lower aircraft's ALIM
"Do not pass above" RAC to higher aircraft
Lower aircraft under selective jamming

Figure 4.3: A simplified version of the branch on **induced** NMAC attacks in the TCAS attack tree.

require that the expected Mode S protocols are obeyed to trick a TCAS into a *false track*. False tracks that are used in a malicious way defines the **Phantom Aircraft** attack.

The concept of the Phantom Aircraft attack is straightforward; TCAS simply trusts that anything responding to its interrogations is an aircraft, and TCAS estimates that the plane is travelling in a certain manner (Section 2.1). Therefore, if an attacker can respond to TCAS's interrogations and appear to move like a plane, then the false track should be successfully created and maintained.

The foundation of the Phantom Aircraft starts with an accurate spoofer of Mode S signals; it must also consistently fool the range measurements that TCAS performs [34, 37]. This is more complex to realize than the unresolved attacks, and there is still probability of a human error in which the pilot ignores the inductive RA. Furthermore, the induced NMAC attacks are still reliant on two threats being near in altitude to increase the probability of successful attack.

The Phantom Aircraft attack should be further explored because of the airspace control an attacker could gain from a fully functional Phantom Aircraft. Therefore, in Chapter 5, the design of a Phantom Aircraft attacker is explored. Chapters 6, 7, and 8 concern the feasibility of realizing the Phantom Aircraft attacker, and these chapters provide some insights on improving the Phantom Aircraft and on creating other attacks discussed.

Chapter 5

Designing an Attack

In Chapter 4, possible attacks are outlined against TCAS. There it is shown that attacks from a remote actor can be classified into either unresolved or induced NMAC attacks. In this chapter, the scope is focused to a particular induced NMAC attack: the **Phantom Aircraft** attack. First, the threat model of the attack is defined. Second, the attack model is outlined. Third, key assumptions made in the attack model are defined. Fourth and finally, the relevant components of TCAS which must be attacked or spoofed are identified.

It should be noted that the attack design in this chapter is only a specific example of a TCAS attack. The design of any attack in the attack tree of Chapter 4 would follow similar steps to the procedure shown in this chapter.

5.1 Defining the Attack

In this section, an outline of the attacker's goals and capabilities are defined. Then, a model of the intended attack is demonstrated, and finally, assumptions about the attack are outlined.

5.1.1 Threat Model

A **threat model** relays an attacker's goals and capabilities. The purpose of the threat model is to demonstrate an attack performed with the fewest possible resources. Items like expensive equipment or access to privileged information tends to limit the scope of an attack. In general, the more limited an attacker is by these capabilities, the easier the defense against the attacker becomes.

The attacker's goal is to induce two planes into an NMAC without any modification of TCAS. This requires the capabilities to perform a Phantom Aircraft attack (see 4), which implies that the attacker must reverse engineer the data link capabilities of TCAS. This will require emulating the surveillance, tracking, and threat detection algorithms that a TCAS CPU performs (see 2.1). Therefore, the attack also requires replying to relevant interrogations as well as emulating *distance closing* signal behavior in which the RTT assumption of the range estimation is fooled with proper response timing.

The attacker is using free, open-source software frameworks for their architecture, and they also want flexible hardware that supports their software. They develop their attack using GNU Radio - a free and open source development platform for SDR that uses a graphical approach to radio design and supports development in C++ or Python [20]. GNU Radio also has support for relatively low-cost SDR hardware from Ettus Research [33].

In a short list, the attacker's capabilities are the following:

- Attacker is only knowledgeable about TCAS through publicly available, no-cost documentation and standards. This excludes most MOPS like DO-178B [4, 16, 28] which give guidance on the development and testing of TCAS.

- Attacker must use free, open-source software.

- Attacker can use an SDR with a programmable on-board FPGA.
- Attacker determines where two aircraft will be in proximity prior to their attack.
- Attacker adequately filters out messages received from itself.
- Attacker selectively jams an aircraft automatically using knowledge of its ICAO address and its reply periodicity.
- Attacker can properly time its replies with respect to the interrogation periodicity of an aircraft to appear closer or farther than it really is.

5.1.2 Attack Model

The **attack model** describes the assumed attack circumstances. In this section, a description of the attack encounter is provided, and a basis for this scenario is justified.

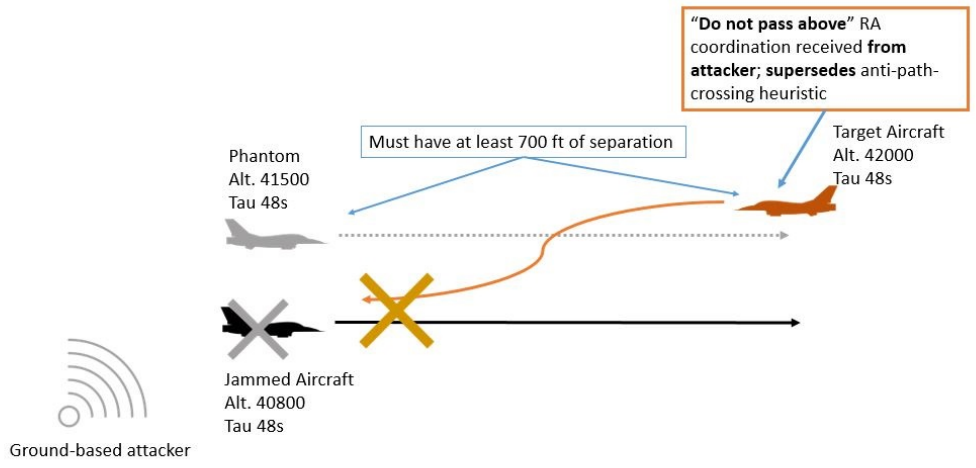

"Do not pass above" RA coordination received from attacker; supersedes anti-path-crossing heuristic
Must have at least 700 ft of separation
Phantom
Alt. 41500
Tau 48s
Target Aircraft
Alt. 42000
Tau 48s
Jammed Aircraft
Alt. 40800
Tau 48s
Ground-based attacker

Figure 5.1: Planned Phantom Aircraft attack against two nearby intruders

First, the intended encounter is the scenario shown in Figure 5.1. Two aircraft are approaching each other without any horizontal offset. These aircraft are also approaching head-on such that there is an exact 180 degree difference between their bearings, and they have no vertical rate on approach. The aircraft are placed such that they have adequate vertical separation that would not elicit any TCAS warnings (see Table A.1).

Prior to the encounter, the attacker can use interrogations to gain intelligence about the two aircraft such as address, range, altitude, etc. Then, the attacker begins selectively jamming the black aircraft such that its track is dropped from the target aircraft. A false track is baited from the target aircraft, and the attacker maintains the track by responding to interrogations. The attacker must maintain knowledge of the encounter geometry by periodically measuring the range of the target aircraft.

Once the attacker has determined that the target is within TA range of the phantom, the phantom must send its RAC first. The attacker can simply announce an RA arbitrarily because TCAS trusts that the RAC it receives is acting in good faith. Combined with a lower ICAO address than the target to force arbitration in the phantom's favor, it should be highly probable that the target aircraft follows the RA coordination.

The attacker forces the target TCAS to cross paths with the phantom by constructing a "Do not pass above" coordination [35]. In combination with the required ALIM for this flight level (see A.1) the target would need to descend to an altitude no greater than 40,800 ft.

In summary, this attack is quite complex; many factors need to be balanced properly in order for it to succeed. Much of the complexities can be attributed to the requirements TCAS faces as a safety-critical system [4, 14, 16, 28, 35, 49]. Nevertheless, the phantom aircraft attack remains a dangerous edge case against a safety-critical system - a system that should be robust to any threats towards safety.

5.1.3 Key Encounter Assumptions

The assumptions made in the attack model are summarized as follows:

- Exactly two aircraft are assumed to be approaching at the altitudes specified, exactly 180 degrees opposed to each other in bearing, and have the altitudes specified in Figure 5.1.

- Each aircraft has an altitude rate of 0 prior to the encounter.

- Each aircraft is TCAS II Version 7.1+ equipped [15].

- Squitter transmission is disabled for each aircraft.

- Messages are 56 bits in size unless otherwise specified.

- Each aircraft is a airline-standard turbine powered aircraft with a carrying capacity of greater than 30 passengers.

- The "jammed aircraft" is jammed early enough such that the target aircraft does not maintain its track on the jammed aircraft.

- The probability of the target aircraft receiving a reply message from the jammed aircraft is 0 due to selective jamming.

- The jammed aircraft does not maneuver.

- The target aircraft will follow an RA if it is generated.

- Weather conditions are such that visual acquisition is either impossible or will not happen in time to avoid NMAC.

- ATC communications are failed or compromised prior to the encounter.

5.2 Attack Components

In section 5.1, a scenario for the Phantom Aircraft attack is proposed, and the attacker's goals and capabilities are shown. In this section, the components of TCAS - from message protocols to high-level operation outlined - are reexamined from an adversarial perspective. Since the threat model assumes that the planes that must be selectively jammed are being jammed, this section is only concerned with the components required to create and maintain the Phantom Aircraft. The components of the TCAS protocol that are useful to the attacker are identified and reasoned.

To accomplish the Phantom Aircraft, there are five goals the attacker must accomplish through the TCAS protocol. The attacker should be able or have the potential to:

1. Perform reconnaissance by interrogating the airspace in the attacker's vicinity.
2. Estimate the trajectory of some victim aircraft through tracking.
3. Bait a false track from the victim by emitting squitters and detecting when interrogations begin.
4. Maintain the false track by responding to interrogations.
5. Declare a threat against the victim and detect evidence that the victim declared its own RA.

5.2.1 Adversarial Interrogations and Replies

It is important to generalize which message format is used at some time because the attacker is operating with one application handling two different functions. A key observation is made to generalize the functions of UF and DF messages:

- **UF** is used when **information must be gained** from a target like altitude, range, etc.
- **DF** is used when **the phantom aircraft's position must be updated**, especially when interrogated for such information.

With this generalization, specific UF and DF messages are crafted to achieve concrete functionality. The following subsections describe these functions; the binary paylods used for each message type is shown in Appendix B.

UF Messages & Their Functions

Three UF messages are identified to perform proximity detection, surveillance, and threat declaration from the attacker's perspective. These messages are UFs 11, 0, and 16, respectively. UF 0 and UF 11 are 56 bit messages, whereas UF 16 is a 112 bit message.

UF 0 is used to obtain an aircraft's altitude code, and its RTT is used to derive range. UF 0 can also request some additional information from an aircraft's transponder like the target's maximum cruising speed instead of. The expected replies to such interrogations are structured as shown in Figure B.3 [15].

UF 11 messages are useful for finding the ICAO addresses of nearby aircraft. When a UF 11's AP field is set to the ACA, *all* transponders that receive the message will respond with their ICAO address in the AA field of its DF 11 response. In practice, commercial aircraft are frequently transmitting their AA in either a DF 11 or 17 message [34, 35]. However, squitter transmission is assumed to be disabled in this model, so acquiring a plane's ICAO address is only possible through a UF 11 interrogation. The UF 11 interrogation and an expected response is shown in Figure B.2.

UF 16 messages are used to coordinate a maneuver. The receiver of the RAC must choose a maneuver that compliments the RAC “unless no more than three [surveillance] cycles have elapsed since the RAC was received” [35]. In a good-faith TCAS coordinated maneuver, RACs are assumed to only occur when TCAS has declared a threat. However, an attacker can abuse this trust by simply transmitting a RAC when it is convenient (every three seconds or less) to guarantee that the aircraft under manipulation follows our intended flight path (see Figure 5.1). An example of the contents of an RAC is shown in Figure B.8.

DF Messages & Their Functions

Three DF messages are the replies to the three UF previously identified messages. These messages can bait a false track from a TCAS, maintain the false track, and maintain a threat declaration against the Phantom Aircraft. These messages are DF 11, DF 0, and DF 16, and they are the replies for UF 11, UF 0, and UF 16, respectively.

DF 0 messages are the responses formed when a Mode S transponder is being interrogated normally. Along with an altitude code, other important information is encoded in the payload like SL, maximum airspeed capabilities, and Vertical Status (VS). VS is a particularly interesting field that, when set, indicates that the plane is on the ground; if the attacker wanted manipulate a track of another aircraft, they could send many DF 0 replies with the VS field equal to 1 because TCAS will not declare aircraft that recently had its VS set to 1 as a threat [34]. An example of an expected DF 0 message sent by the attacker is shown in Figure B.5.

DF 11 messages are normally used as squitters. The majority of their payload is used to transmit the aircraft’s plain 24 bit ICAO address. Nevertheless, they are sufficient to bait a track from TCAS when successfully received by a TCAS multiple times (see Section 2.1).

An example of such a squitter is shown in Figure B.4.

DF 16 messages are essentially DF 0 messages with extra information about the RACs it recently received. They do not command any RA coordination, unlike its UF 16 counterpart. However, DF 16 messages are useful in verifying that another TCAS has active RAs; in other words, it can verify that the attacker successfully generated an RA from a Phantom Aircraft. Example contents of the DF 16 message are shown in Figure B.9.

5.2.2 Protocol Exploitation

The five required functions of the phantom aircraft are shown in a state machine representation in Figure 5.2. The attack starts by acquiring the ICAO address of nearby aircraft. Then, the attacker surveys a target for as long as necessary to develop an accurate model of each plane. The attacker then goes through baiting a track, maintaining the track, and declaring a threat. The remainder of this section will detail how each message is used to accomplish the functional goals.

Proximity Detection

Proximity Detection is the concept of locating nearby areas for aircraft to interrogate. It allows the attacker to acquire the ICAO address nearby TCAS; this enables the attacker to discretely interrogate transponders with other UF messages. The message the attacker transmits and the expected responses are shown in Figure B.2.

In the state machine of Figure 5.2, proximity detection occurs until DF 11 is received from the expected target. In practice, the attacker would likely repeat proximity detection until they receive multiple DF 11 replies to add confidence that the aircraft will remain in range for a sustained duration.

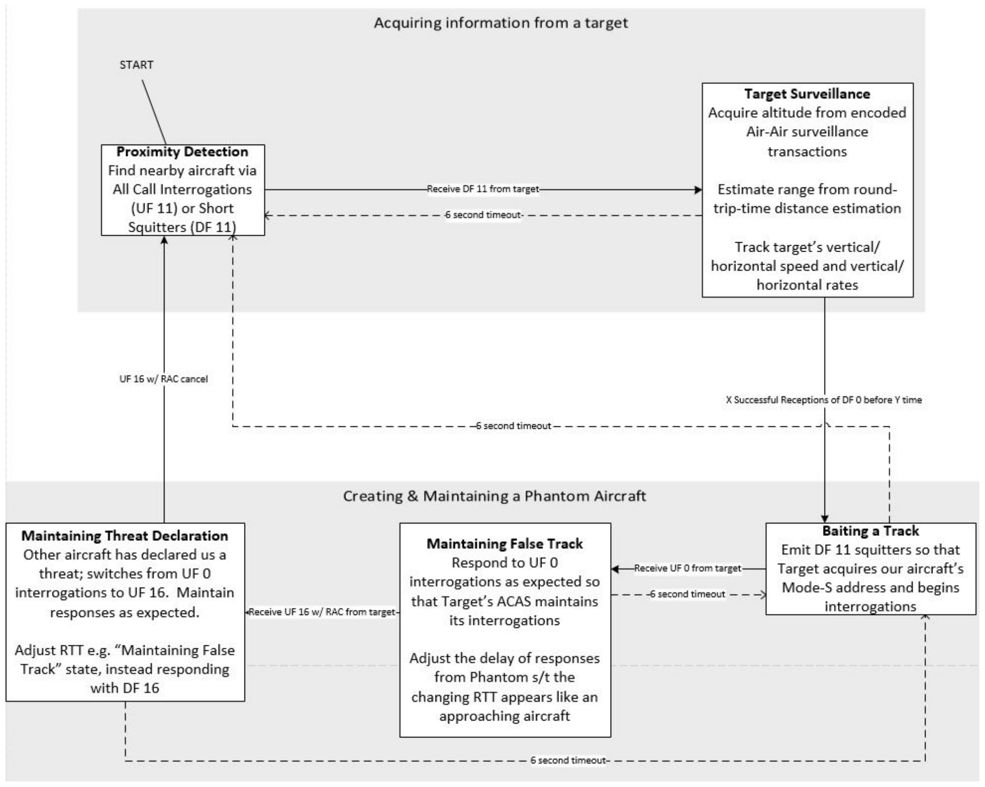

Acquiring information from a target
START
Proximity Detection
Find nearby aircraft via All Call Interrogations (UF 11) or Short Squitters (DF 11)
Receive DF 11 from target
6 second timeout
Target Surveillance
Acquire altitude from encoded Air-Air surveillance transactions
Estimate range from round-trip-time distance estimation
Track target's vertical/ horizontal speed and vertical/ horizontal rates
UF 16 w/ RAC cancel
X Successful Receptions of DF 0 before Y time
6 second timeout
Creating & Maintaining a Phantom Aircraft
Maintaining Threat Declaration
Other aircraft has declared us a threat; switches from UF 0 interrogations to UF 16. Maintain responses as expected.
Adjust RTT e.g. "Maintaining False Track" state, instead responding with DF 16
Receive UF 16 w/ RAC from target
Maintaining False Track
Respond to UF 0 interrogations as expected so that Target's ACAS maintains its interrogations
Adjust the delay of responses from Phantom s/t the changing RTT appears like an approaching aircraft
Receive UF 0 from target
6 second timeout
Baiting a Track
Emit DF 11 squitters so that Target acquires our aircraft's Mode-S address and begins interrogations
6 second timeout

Figure 5.2: Phantom Aircraft state machine

Once the attacker is confident in acquiring the nearby aircraft necessary, they can move on to surveying each aircraft for more data.

Target Surveillance

Target Surveillance is the concept of obtaining data about a particular aircraft. The purpose of the attacker's surveillance is that the attacker develops a tracking model for each aircraft. By tracking important points like altitude, range, and their respective derivatives, the attacker can predict how the victim aircraft is moving and create a phantom aircraft accordingly. Experimentation is required to find a sufficient amount of surveillance cycles. The transmitted message and its expected responses are shown in Figure B.3.

Baiting a Track

Baiting a Track is the concept of tricking a TCAS into interrogating the attacker. If a TCAS begins to interrogate the attacker and maintains interrogations for an extended period of time, it is inferred that the target's TCAS must be maintaining a track against an aircraft that does not exist. TCAS can interrogate the attacker with UF 11, and the attacker *should* respond to these interrogations like a Mode S transponder would.

However, TCAS in practice tends to simply monitor the nearby airspace for squitters [34]. When the attacker is ready to deploy its phantom aircraft, they can periodically transmit a squitter once per second to appear like an intruder releasing squitters. Then, the attacker can wait until nearby aircraft begin interrogating them before moving onto the next state. The transmitted message and its expected response is shown in Figure B.4.

Maintaining a Track

Maintaining a Track requires that the attacker do two critical things: they must respond to all UF 0 interrogations, and they must time the response to *appear* some distance away.

Responding to UF 0 interrogations is simple enough. The expected response to an interrogation is shown in Figure B.5. In a practical attack, the attacker would likely need to account for all the data a TCAS may request from a transponder [35].

In this attack model, it is assumed that the victim will only interrogate with one of two variants: either the AQ field is set or not set. This modifies the contents of the RI field to be either 12 or 3, respectively. These values report the maximum cruising speed of an aircraft and the aircraft's TCAS capabilities, respectively. Therefore, the attacker should adjust the RI field when the interrogation's AQ is 1 depending on the trajectory the attacker wishes the phantom aircraft to emulate. When AQ is 0, RI should always report 3 to report that the phantom aircraft is only capable of vertical RAs.

Timing the transmission properly to fool the RTT assumption of range estimation is something the attacker will need to predict at run-time because the time at which TCAS begins interrogations is up to implementation of the TCAS software [35]. However, the threat model defined in Section 5.1 assumes the attacker can time its responses properly, and, therefore, the timing of such messages is outside the scope of this thesis.

Creating a RAC and Maintaining a Threat Declaration

Creating a RAC is as simple as transmitting UF 16 to a victim aircraft. Figures B.7 and B.8 demonstrate how one would craft an RAC to prevent an aircraft from generating an RA that goes below another aircraft's altitude and how to cancel the same RAC in another transmission.

UF 16 is the only interrogation that breaks the request-response paradigm. Although a Mode S transponder receives the interrogation, it sets the appropriate registers in the TCAS CPU and does not respond. In order to verify that the RAC was received successfully, the attacker needs to *also* use a UF 0 interrogation with the RL field set and AQ unset so that the transponder responds with its active RACs in a 112 bit message.

Maintaining a Threat Declaration is the concept that the attacker must keep responding to interrogations so that the TA or RA against them is not dropped. UF 0 is still used for tracking throughout the threat encounter. However, the RL field should be set and the AQ field should be unset such that the response formed is DF 16. Figure B.6 shows how the attacker would respond to such interrogations, and Figure B.9 shows the contents of the MV field of that response.

Chapter 6

TCAS Attack Implementation

In this Chapter, the threat model defined in Chapter 5 is implemented into a model communications application. The intention of this design is to demonstrate the **proof-of-concept** that a GNU Radio application can bait a TCAS into surveying the phantom aircraft platform. A simplified general aircraft state machine is presented, block diagrams of the major components are presented in Section 6.2, and its implementation in GNU Radio is discussed in Section 6.4. A discussion of the hardware, simulation performance, and Hardware-in-the-Loop (HIL) operation is also presented.

6.1 Defining Constraints

The goal of this system is to have a pair of zero-speed aircraft track each other. This is sufficient for the proof-of-concept because the core functionality of an attacker and a normal aircraft are identical; in other words, the difference between an malicious and a non-malicious phantom aircraft is *intention*. How one uses the phantom aircraft dictates whether it is malicious or not. Therefore, the design will henceforth be referred to as a **phantom aircraft generator** since this system does not make assumptions on the user's intentions.

The generator will operate on simplified tracking and threat detection models such that a threat is declared against an aircraft when it is in RA distance without any trajectory prediction. Also, the two different parity checks for interrogations and replies are combined into a single parity check scheme [30].

The design presented is focused on being as modular as possible within the GNU Radio ecosystem and to build upon other open-source GNU Radio modules. The basic concept is that one could swap in different “core applications” that change the behavior of the phantom aircraft application while maintaining a consistent communications front end.

The Python 2.7.12 **Out-of-Tree** (OOT) module created for this thesis is called `gr-modes`. It contains custom aircraft application code and modules that are designed to aid in message transformations. Some custom blocks of `gr-modes` designed in this chapter are based off the ADS-B OOT for GNU Radio, `gr-adsb` [29]. Also, the reception and demodulation of DF packets is handled by some `gr-adsb` blocks.

The total OOT dependencies for `gr-modes` are:

- `gr-adsb`: demodulation and decoding of ADS-B messages [29]
- `gr-burst`: blocks for building PSK modems [52]
- `gr-eventstream` [51]
- `gr-mapper` [53]
- `gr-pyqt` [50]

6.2 Block Diagrams

A series of block diagrams are presented to clarify the architecture of the software system, the sub-routines of each high-level block, and the interfaces between each function. Figure 6.1 defines the overview of the phantom aircraft generator as well as expected carrier frequencies, modulations, and sample rates. The system uses a mixture of **asynchronous** and **streamed** data. Interfaces that use data *streams* are shown with solid lines, and interfaces that use *asynchronous* data are shown with dashed lines.

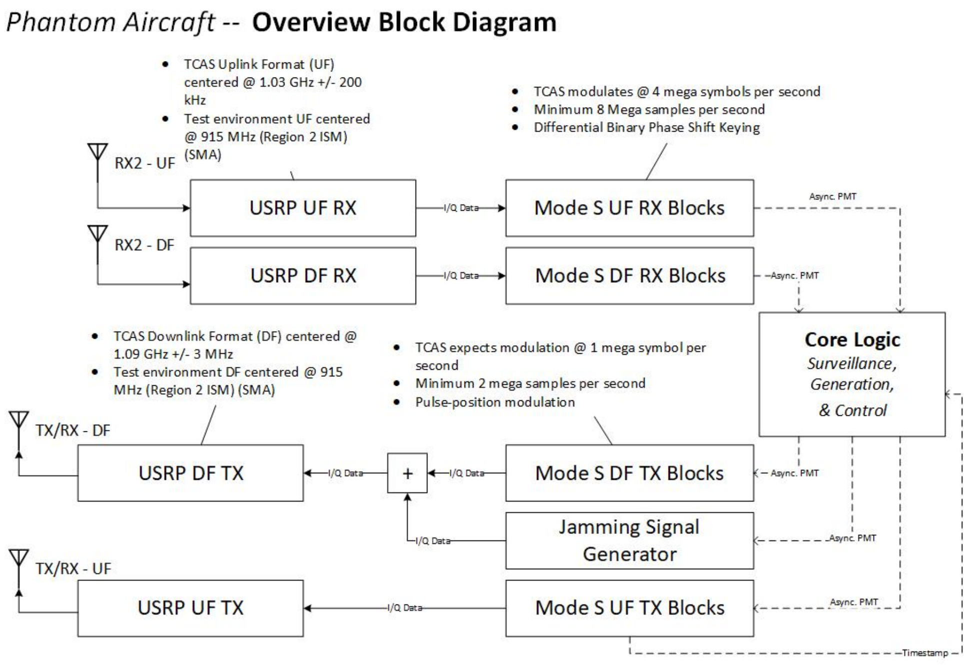
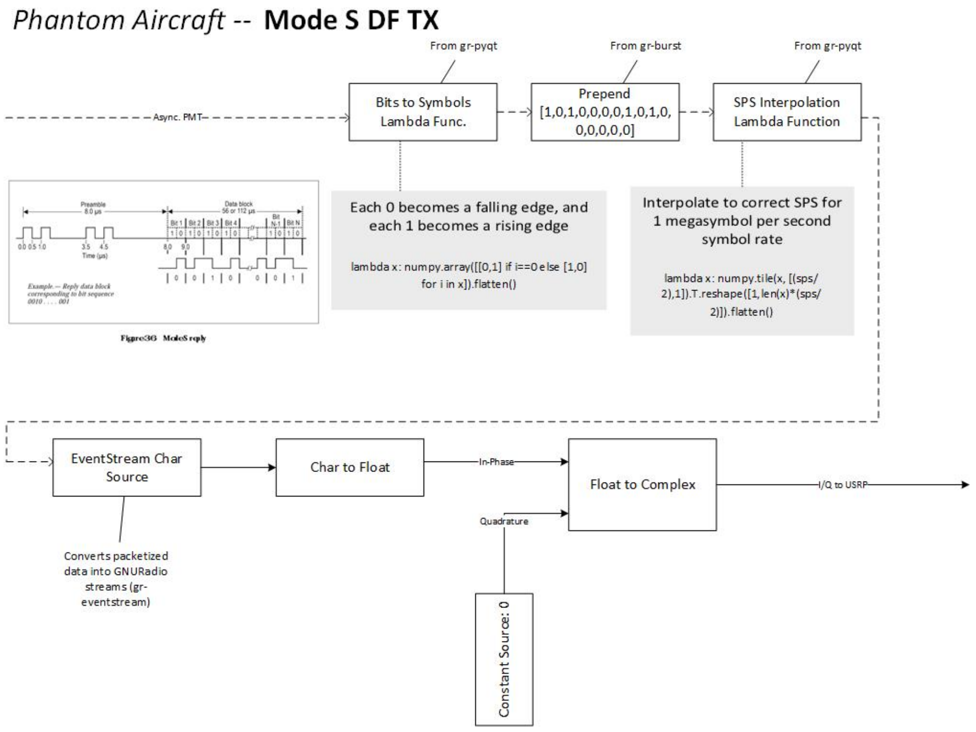

Phantom Aircraft -- Overview Block Diagram
TCAS Uplink Format (UF) centered @ 1.03 GHz +/- 200 kHz
Test environment UF centered @ 915 MHz (Region 2 ISM) (SMA)
TCAS modulates @ 4 mega symbols per second
Minimum 8 Mega samples per second
Differential Binary Phase Shift Keying
RX2 - UF
USRP UF RX
I/Q Data
Mode S UF RX Blocks
Async. PMT
RX2 - DF
USRP DF RX
I/Q Data
Mode S DF RX Blocks
Async. PMT
Core Logic
Surveillance, Generation, & Control
TCAS Downlink Format (DF) centered @ 1.09 GHz +/- 3 MHz
Test environment DF centered @ 915 MHz (Region 2 ISM) (SMA)
TCAS expects modulation @ 1 mega symbol per second
Minimum 2 mega samples per second
Pulse-position modulation
TX/RX - DF
USRP DF TX
I/Q Data
+
I/Q Data
Mode S DF TX Blocks
Async. PMT
I/Q Data
Jamming Signal Generator
Async. PMT
TX/RX - UF
USRP UF TX
I/Q Data
Mode S UF TX Blocks
Async. PMT
Timestamp

Figure 6.1: Overview block diagram of the phantom aircraft generator.

Like the TCAS block diagram in Figure 2.7, this system also requires four RF interfaces: a TX and RX interface for both interrogations and replies. The interfaces do not use the normal frequencies bands of TCAS [15, 34, 35]. Instead, a nearby carrier frequency in the industrial, scientific, and medical (ISM) band is chosen to open the opportunity for wireless testing [36]. Each RX interface has its own set of blocks that have the task of converting raw samples into packets the `Core Logic` can understand; each TX interface performs the conversion in the opposite direction.

Recall that range estimation occurs from the RTT of the interrogation-reply cycle. For estimation of RTT, the UF TX blocks feedback a timestamp to the `Core Logic`. The elapsed time is calculated from a timestamp that is delivered by the DF RX blocks upon receipt.

The overview supports a jamming interface from the `core logic`. The core logic would `enable` or `disable` the jam for an arbitrary amount of samples. The jamming signal generator would handle the generation of a random samples independent of the `core logic`. However, as discussed in Chapter 5, a selective jammer implementation is outside the scope of this thesis.


Phantom Aircraft -- Mode S DF TX
From gr-pyqt
From gr-burst
From gr-pyqt
Async. PMT
Bits to Symbols Lambda Func.
Prepend [1,0,1,0,0,0,0,1,0,1,0, 0,0,0,0,0]
SPS Interpolation Lambda Function
Each 0 becomes a falling edge, and each 1 becomes a rising edge
Interpolate to correct SPS for 1 megasymbol per second symbol rate
EventStream Char Source
Char to Float
In-Phase
Quadrature
Float to Complex
I/Q to USRP
Converts packetized data into GNURadio streams (gr-eventstream)
Constant Source: 0

```
lambda x: numpy.array([[0,1] if i==0 else [1,0]
for i in x]).flatten()
```

```
lambda x: numpy.tile(x, [(sps/
2),1]).T.reshape([1,len(x)*(sps/
2)]).flatten()
```

Figure 6.2: Communications interface for reply message transmission.

All communication interfaces support **unpacked bytes**. For the purpose of this design, this means that an 8 bit sample can only hold the value of a binary one or binary zero. While inefficient in its memory usage, this design choice does make the design of other components - especially `lambda function` blocks [54] - much simpler.

Figure 6.2 shows the conversion of data from the `Core Logic` to samples that the `USRP` blocks expect. This communication interface also handles the packing of message data into a complete Mode S message (see Figure 2.12). A `lambda function` block from `gr-pyqt` and a `prepend preamble` block from `gr-burst` are convenient for packing the data into a Mode S reply [50, 52].

The data that enters the DF TX interface from the `Core Logic` is 56 or 112 samples of bytes in length (although this interface is designed to be independent of payload size). To convert this message into a Mode S reply, the following procedure is performed:

1. PPM is performed on each sample [21].
2. The Mode S reply preamble (a constant 16 sample vector with 1 sample per symbol assumed) is attached to the beginning of the packet (see Figure 2.12).
3. The packet is interpolated to the application's base sample rate.

Then, the packet must be converted to a *stream* of complex samples because GNU Radio only supports transmitting data over USRP that is in 32 bit complex stream format. Once the payload is packed into a complete Mode S reply, `gr-eventstream` is used to insert the data from an *asynchronous* packet into the sample stream, and conversion to a 32 bit complex type is performed with built-in conversion blocks.

Figure 6.3 shows the receiver interface for DF messages. The receiver is made from `gr-adsb` blocks, which expect squared magnitude 32 bit float samples [29].

The `ADS-B Framer` is slightly modified such that the `ADS-B Framer` block creates a timestamp that propagates to the `Core Logic` when a preamble is detected. The `ADS-B Framer` block correlates the input with the expected preamble (see Figure 2.12) and creates a `tag` object on the first sample of the preamble when a match is detected. This `tag` serves as the synchronization point which the `ADS-B Demodulator` can use for PPM demodulation. The decoder in the `gr-adsb` OOT module is bypassed entirely, and the asynchronous output is

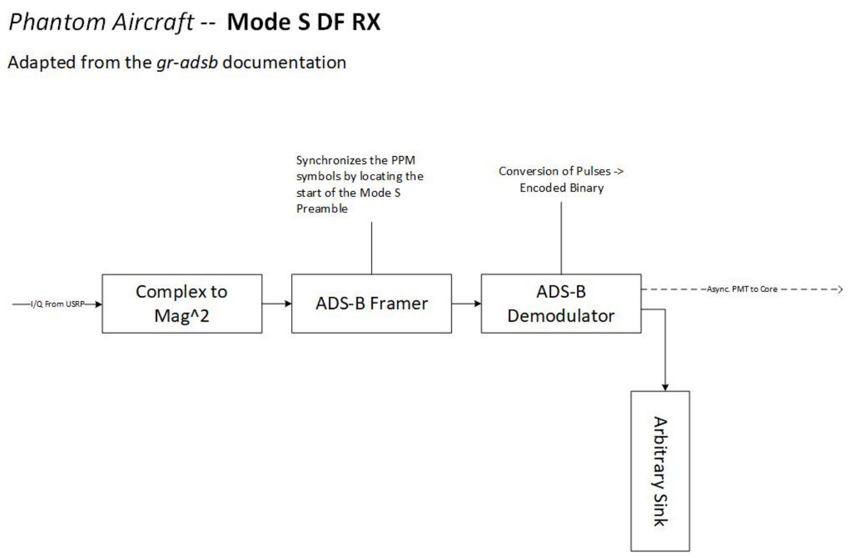

Phantom Aircraft -- Mode S DF RX
Adapted from the gr-adsb documentation
Synchronizes the PPM symbols by locating the start of the Mode S Preamble
Conversion of Pulses -> Encoded Binary
I/Q From USRP
Complex to Mag^2
ADS-B Framer
ADS-B Demodulator
Async. PMT to Core
Arbitrary Sink

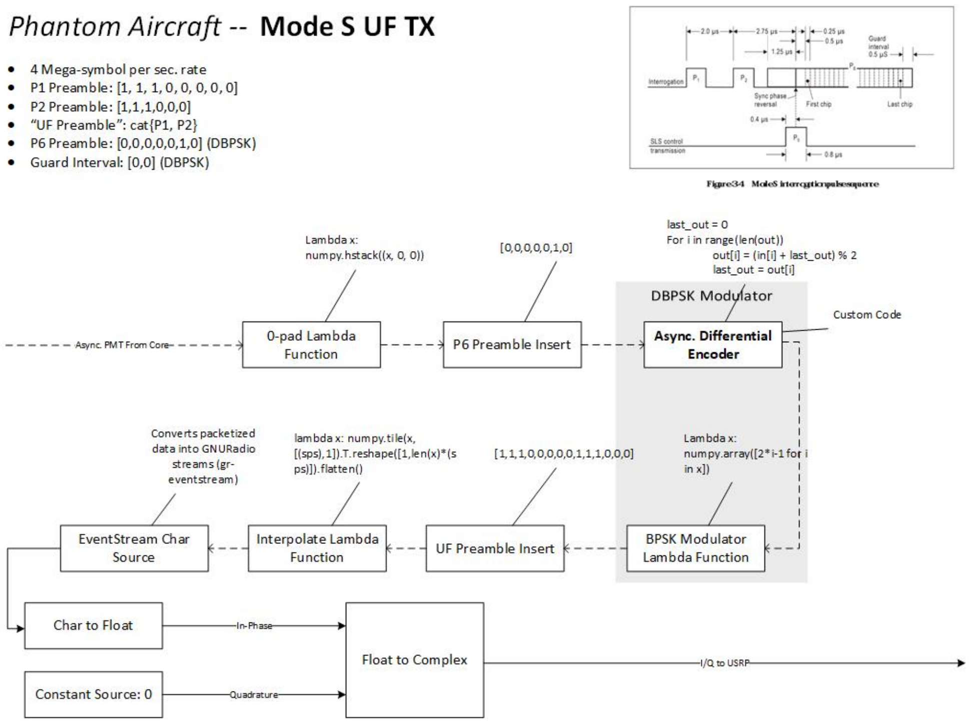
Figure 6.3: Communications interface for reply message reception.

sent directly to the `Core Logic` for custom decoding routines.

Figure 6.4 mimics the design of the reply transmission interface such that the interface remains independent of actual message data. However, some adjustments have to be made to support the different pulse sequence and modulation scheme of Mode S interrogations (see Figure 2.10).

Namely, Mode S interrogations have slightly more padding than replies. There is a two sample pad of zeros (assuming 1 sample per symbol) appended to the end of the message payload, and a seven sample preamble is attached to the message. The preamble is structured such that one phase shift occurs at the exact point labeled "Sync. Phase Reversal" [35] in Figure 2.10 after the preamble is put through DBPSK modulation.

Separate from the preamble that contains the "Sync. Phase Reversal", another preamble is attached to the output of the DBPSK modulator. This second preamble suppresses Modes A and C transponders from processing the interrogation. The interval between the first and second pulse dictates how Modes A and C transponders reply (if at all) to an interrogation


Phantom Aircraft -- **Mode S UF TX**
- 4 Mega-symbol per sec. rate
- P1 Preamble: [1, 1, 1, 0, 0, 0, 0, 0]
- P2 Preamble: [1,1,1,0,0,0]
- “UF Preamble”: cat{P1, P2}
- P6 Preamble: [0,0,0,0,0,1,0] (DBPSK)
- Guard Interval: [0,0] (DBPSK)
2.0 μs
2.75 μs
0.25 μs
0.5 μs
1.25 μs
Guard interval 0.5 μS
Interrogation
P1
P2
P6
Sync phase reversal
First chip
Last chip
0.4 μs
SLS control transmission
P5
0.8 μs
Figure 34 Mode S interrogation pulse sequence
Lambda x: numpy.hstack((x, 0, 0))
[0,0,0,0,0,1,0]
DBPSK Modulator
Custom Code
Async. PMT From Core
0-pad Lambda Function
P6 Preamble Insert
Async. Differential Encoder
Converts packetized data into GNURadio streams (gr-eventstream)
lambda x: numpy.tile(x, [(sps),1]).T.reshape([1,len(x)*(sps)]).flatten()
[1,1,1,0,0,0,0,0,1,1,1,0,0,0]
Lambda x: numpy.array([2*i-1 for i in x])
EventStream Char Source
Interpolate Lambda Function
UF Preamble Insert
BPSK Modulator Lambda Function
Char to Float
In-Phase
Constant Source: 0
Quadrature
Float to Complex
I/Q to USRP

```
last_out = 0
For i in range(len(out))
        out[i] = (in[i] + last_out) % 2
        last_out = out[i]
```

Figure 6.4: Communications interface for interrogation message transmission.

[35]. After the last preamble is inserted, the payload is finally transmitted and converted to a data stream format that the USRP expects.

Note that the “[Side Lobe Suppression] (SLS) control transmission” is omitted in interrogations for simplification. The purpose of this transmission is to prevent aircraft to the side or behind the aircraft from responding in special circumstances. The SLS control is transmitted on a separate antenna and is designed to hide the “Sync. Phase Reversal” of an interrogation, which prevents a Mode S transponder from synchronizing its incoming waveform [35].

Figure 6.5 shows the receiving interface for interrogations. The design is similar to the approach taken in Figure 6.3; an `Interrogation Framer` block performs correlation of the preamble and creates a `tag` on the first sample of the preamble.

Using the aforementioned `tag` as a synchronization point, the `Tag Consumer` block then

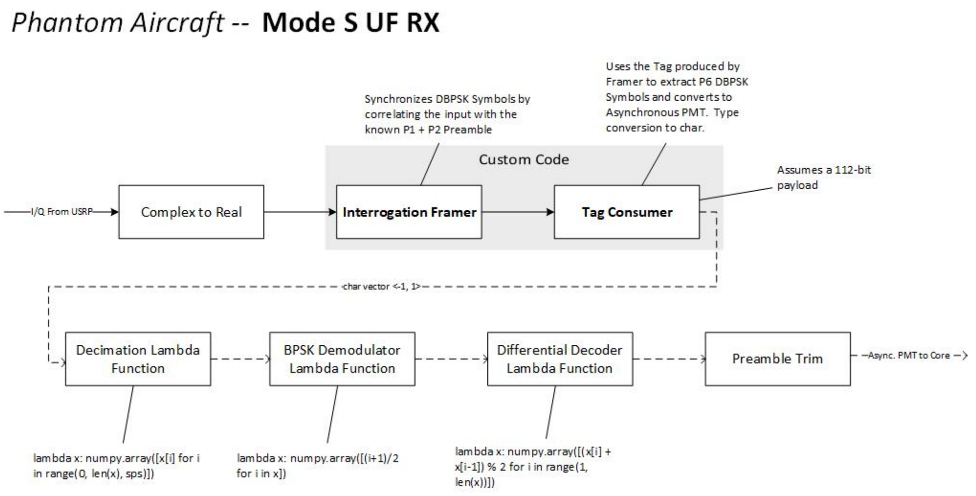

Phantom Aircraft -- Mode S UF RX
Synchronizes DBPSK Symbols by correlating the input with the known P1 + P2 Preamble
Uses the Tag produced by Framer to extract P6 DBPSK Symbols and converts to Asynchronous PMT. Type conversion to char.
Custom Code
Assumes a 112-bit payload
I/Q From USRP
Complex to Real
Interrogation Framer
Tag Consumer
char vector <-1, 1>
Decimation Lambda Function
BPSK Demodulator Lambda Function
Differential Decoder Lambda Function
Preamble Trim
Async. PMT to Core
lambda x: numpy.array([x[i] for i in range(0, len(x), sps)])
lambda x: numpy.array([(i+1)/2 for i in x])
lambda x: numpy.array([(x[i] + x[i-1]) % 2 for i in range(1, len(x))])

Figure 6.5: Communications interface for interrogation message reception.

creates a "slice" of samples. The slice starts from the first sample of the DBPSK samples and ends 120 samples later (i.e. length of the preamble + 112 symbols assumed). This slice is packaged into a format acceptable for asynchronous message passing, and then, demodulation is performed asynchronously.

Although a 112 symbol payload is assumed, the extraneous information packaged when a short message was actually received is tolerable because the core knows the true message size based on the fixed 5 symbol header.

Figure 6.6 shows the receiving interfaces of the `Core Logic` module. They convert interrogation and reply samples into a format the state machine logic can understand.

The conversion process is based off the routines of the `ADS-B Decoder` of the `gr-adsb` package. Because of its fixed length, The header is decoded first to provide information about verifying the checksum in the final 24 bits of the payload. If the parity check passes, then the vector of samples is converted into a `downlink object` or `uplink object` depending on which interface is used.

The `downlink` and `uplink` objects are provided by a Python module written for supporting

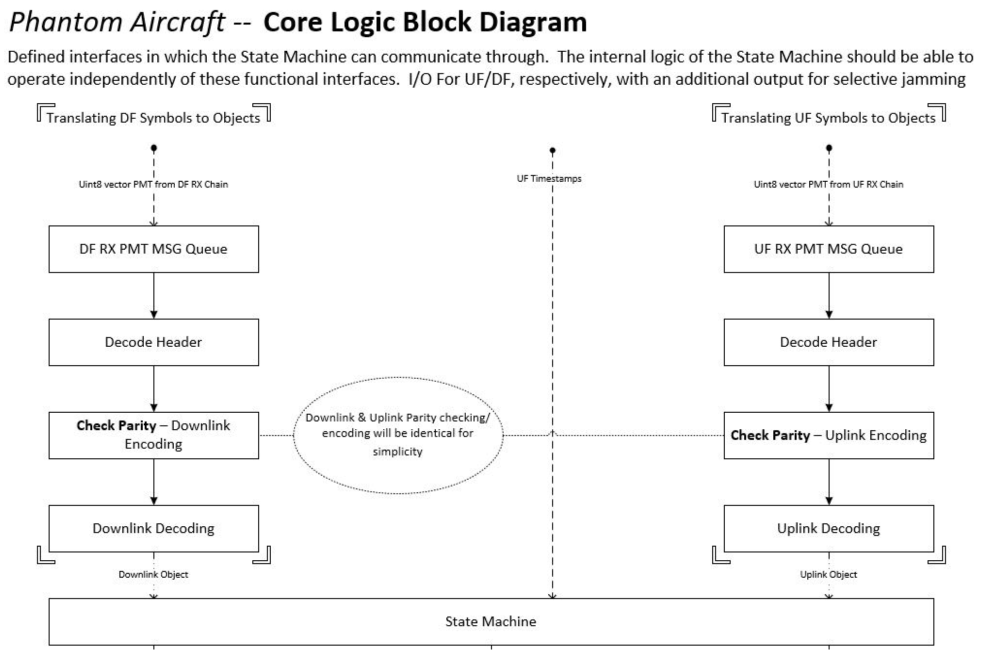

Phantom Aircraft -- Core Logic Block Diagram
Defined interfaces in which the State Machine can communicate through. The internal logic of the State Machine should be able to operate independently of these functional interfaces. I/O For UF/DF, respectively, with an additional output for selective jamming
Translating DF Symbols to Objects
Translating UF Symbols to Objects
Uint8 vector PMT from DF RX Chain
UF Timestamps
Uint8 vector PMT from UF RX Chain
DF RX PMT MSG Queue
UF RX PMT MSG Queue
Decode Header
Decode Header
Check Parity – Downlink Encoding
Downlink & Uplink Parity checking/ encoding will be identical for simplicity
Check Parity – Uplink Encoding
Downlink Decoding
Uplink Decoding
Downlink Object
Uplink Object
State Machine

Figure 6.6: Subroutines of the core logic on the incoming message interfaces for delivering payloads to the state machine.

the `Core Logic` functions. The module provides classes for each message type as well as useful helping functions.

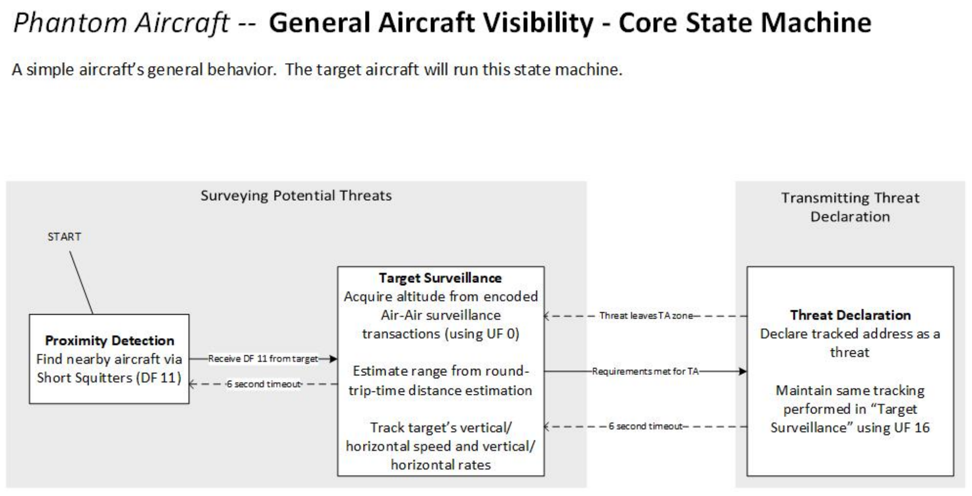

Phantom Aircraft -- General Aircraft Visibility - Core State Machine
A simple aircraft's general behavior. The target aircraft will run this state machine.
Surveying Potential Threats
Transmitting Threat Declaration
START
Proximity Detection
Find nearby aircraft via Short Squitters (DF 11)
Receive DF 11 from target
6 second timeout
Target Surveillance
Acquire altitude from encoded Air-Air surveillance transactions (using UF 0)
Estimate range from round-trip-time distance estimation
Track target's vertical/ horizontal speed and vertical/ horizontal rates
Threat leaves TA zone
Requirements met for TA
6 second timeout
Threat Declaration
Declare tracked address as a threat
Maintain same tracking performed in "Target Surveillance" using UF 16

Figure 6.7: State machine representation of a basic aircraft that looks for squitters and begins surveillance.

The `state machine` blocks can be "swapped" out to change the behavior of a phantom aircraft. For this proof-of-concept, a simplified state-machine operation used in this design is shown in Figure 6.7. These aircraft simply detect nearby DF 11 messages and begin surveillance of the aircraft that transmitted the message.

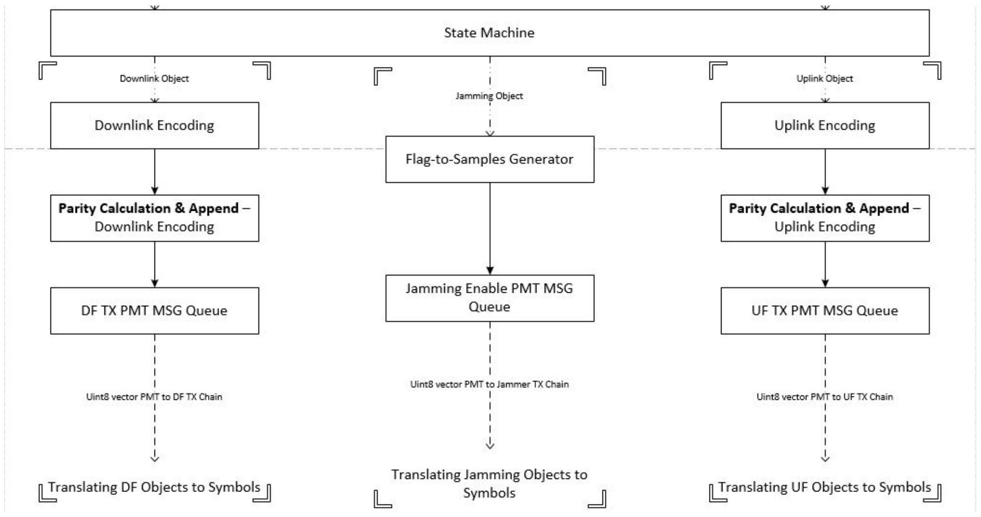

State Machine
Downlink Object
Jamming Object
Uplink Object
Downlink Encoding
Flag-to-Samples Generator
Uplink Encoding
Parity Calculation & Append – Downlink Encoding
Parity Calculation & Append – Uplink Encoding
DF TX PMT MSG Queue
Jamming Enable PMT MSG Queue
UF TX PMT MSG Queue
Uint8 vector PMT to DF TX Chain
Uint8 vector PMT to Jammer TX Chain
Uint8 vector PMT to UF TX Chain
Translating DF Objects to Symbols
Translating Jamming Objects to Symbols
Translating UF Objects to Symbols

Figure 6.8: Subroutines of the core logic on the outgoing message interfaces for transmitting payloads from the state machine.

Figure 6.8 shows the conversion of objects to vectors that the modulators can understand. The classes for `downlink` and `uplink` objects automatically perform the construction of the vector and the parity check. The only requirement of the user is to call the automatic conversion function, and then, they place the result of that routine into the output message queue.

6.3 Hardware

In this section, the hardware used in this implementation is shown. The PC specifications and the SDR platforms are outlined. Then, a diagram of the hardware components' connections is given.

USRP hardware requires significant processing power in general, especially at high sample rates. The PC in this setup uses an Intel **Core i7-6800K**; it has six cores at a 3.4 GHz clock rate. Combined with 16 GB of RAM, this PC provides adequate flexibility in processing two USRPs simultaneously.

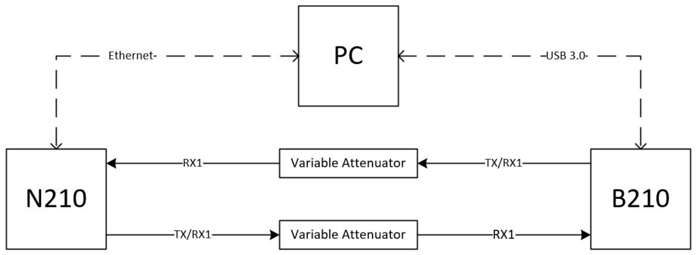


Figure 6.9: Experimental hardware setup for HIL testing.

Two USRPs act like a pair of communicating TCAS. An Ettus Research **B210** is the transmitting USRP, and an Ettus Research **N210** is the receiving USRP [45, 46]. Both USRPs have programmable FPGAs which enables better performance if software alone is insufficient for some function. The N210 only has a single pair of SMA-bulkheads which must be accounted for in a full test of TCAS; this can be solved by either introducing power combiners or by switching SDR platform (e.g. using an identical B210).

The full hardware setup is shown as a block diagram in Figure 6.9. The two USRPs communicate over SMA cables through a variable attenuator. The attenuator prevents the receiver's analog to digital converter (ADC) from being saturated. The N210 communicates with the PC over 10 Gbps Ethernet, and the B210 communicates over a USB 3.0 port.

6.4 Flowgraphs

GNU Radio Companion (GRC) is a graphical application packaged with GNU Radio. It allows for quick conversion of block diagrams into software applications because GNU Radio relies on the abstraction of a **flowgraph** for rapid development [20].

The GNU Radio ecosystem often divides its processing functions into **blocks**. In GNU Radio, a block is an independent piece of arbitrary code that has its own thread. GNU Radio has a built-in scheduler to determine which block gets to execute at a given time. All blocks are contained in a **flowgraph** which defines how the blocks are connected as well as other system-level parameters.

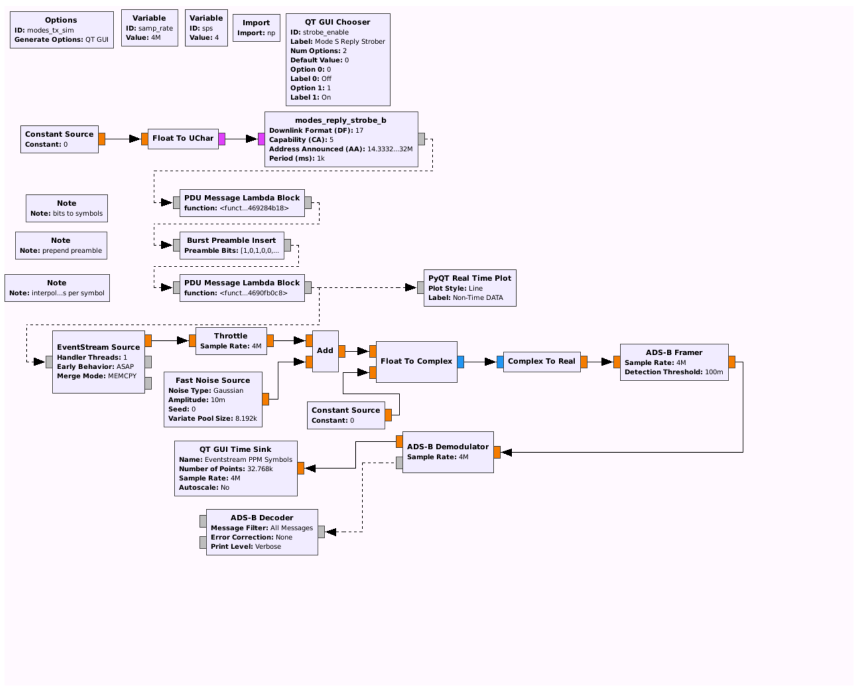
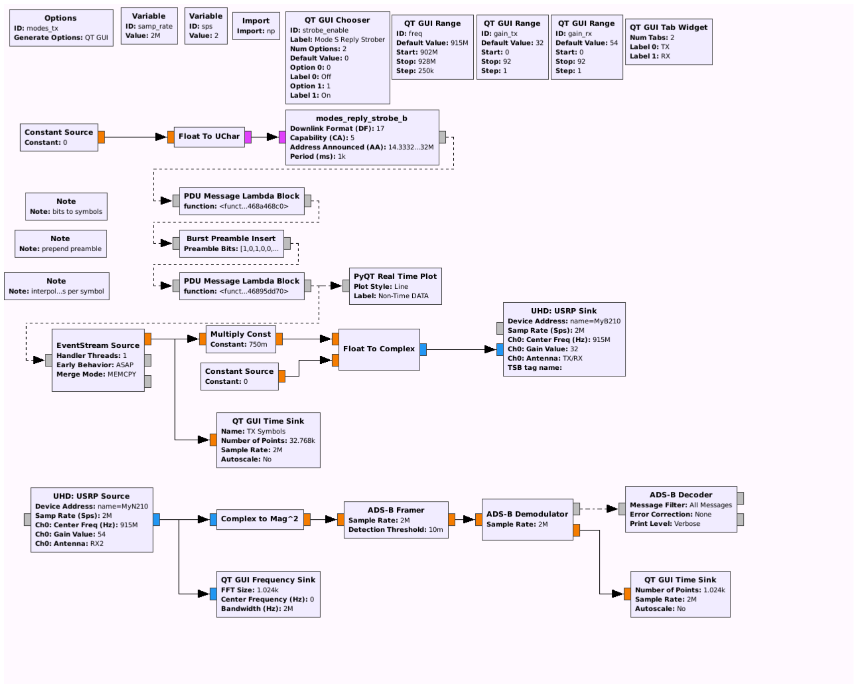


Figure 6.10: A flowgraph for testing the communication interfaces for reply messages without USRP hardware.

Figure 6.10 is a flowgraph for the TX and RX interfaces of reply messages; it implements Figures 6.2 and 6.3. The reply strobe block periodically sends a 112 bit data message to the beginning of the DF TX interface. The TX interface is directly connected to the RX interface, and the `ADS-B Decoder` is utilized to verify that message integrity is maintained. Additive White Gaussian Noise (AWGN) is added into the data stream to simulate the effects of a noisy channel.

The same flowgraph implementation is then adapted to use USRP hardware to transmit and


Options
ID: modes_tx
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: sps
Value: 2

Import
Import: np

QT GUI Chooser
ID: strobe_enable
Label: Mode S Reply Strober
Num Options: 2
Default Value: 0
Option 0: 0
Label 0: Off
Option 1: 1
Label 1: On

QT GUI Range
ID: freq
Default Value: 915M
Start: 902M
Stop: 928M
Step: 250k

QT GUI Range
ID: gain_tx
Default Value: 32
Start: 0
Stop: 92
Step: 1

QT GUI Range
ID: gain_rx
Default Value: 54
Start: 0
Stop: 92
Step: 1

QT GUI Tab Widget
Num Tabs: 2
Label 0: TX
Label 1: RX

Constant Source
Constant: 0

Float To UChar

modes_reply_strobe_b
Downlink Format (DF): 17
Capability (CA): 5
Address Announced (AA): 14.3332...32M
Period (ms): 1k

Note
Note: bits to symbols

PDU Message Lambda Block
function: <funct...468a468c0>

Note
Note: prepend preamble

Burst Preamble Insert
Preamble Bits: [1,0,1,0,0,...

Note
Note: interpol...s per symbol

PDU Message Lambda Block
function: <funct...46895dd70>

PyQT Real Time Plot
Plot Style: Line
Label: Non-Time DATA

EventStream Source
Handler Threads: 1
Early Behavior: ASAP
Merge Mode: MEMCPY

Multiply Const
Constant: 750m

Constant Source
Constant: 0

Float To Complex

UHD: USRP Sink
Device Address: name=MyB210
Samp Rate (Sps): 2M
Ch0: Center Freq (Hz): 915M
Ch0: Gain Value: 32
Ch0: Antenna: TX/RX
TSB tag name:

QT GUI Time Sink
Name: TX Symbols
Number of Points: 32.768k
Sample Rate: 2M
Autoscale: No

UHD: USRP Source
Device Address: name=MyN210
Samp Rate (Sps): 2M
Ch0: Center Freq (Hz): 915M
Ch0: Gain Value: 54
Ch0: Antenna: RX2

Complex to Mag^2

ADS-B Framer
Sample Rate: 2M
Detection Threshold: 10m

ADS-B Demodulator
Sample Rate: 2M

ADS-B Decoder
Message Filter: All Messages
Error Correction: None
Print Level: Verbose

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 2M

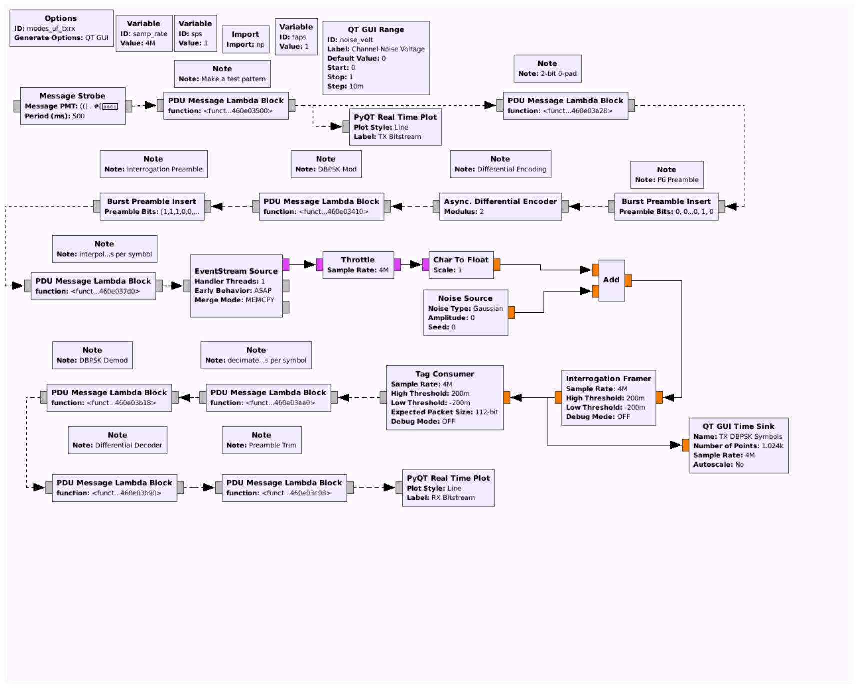
QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 2M
Autoscale: No

Figure 6.11: A flowgraph for testing the communication interfaces for reply messages over USRP hardware.

receive an identical payload to the one used in Figure 6.10. This flowgraph enables insights into how well two USRPs perform together to emulate phantom aircraft generation. Again, the `ADS-B Framer`, `Demodulator`, and `Decoder` are used to verify data integrity over the channel. This hardware test also allows for estimating the amount of messages dropped over the medium.

Likewise, Figure 6.12 is a flowgraph for the TX and RX interfaces of interrogations; it implements Figures 6.4 and 6.5. Similar to the DF case, this case transmits a periodic interrogation. The TX and RX interfaces are directly connected with AWGN to simulate a noisy channel.

Combining Figures 6.10 and 6.12, it is now possible to implement a flowgraph of two general phantom aircraft. Figure 6.13 shows how the two aircraft cores fit into a looping flowgraph. With the addition of a periodically transmitted squitter external to the cores, the top core



Options
ID: modes_uf_txrx
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 4M

Variable
ID: sps
Value: 1

Import
Import: np

Variable
ID: taps
Value: 1

QT GUI Range
ID: noise_volt
Label: Channel Noise Voltage
Default Value: 0
Start: 0
Stop: 1
Step: 10m

Note
Note: Make a test pattern

Note
Note: 2-bit 0-pad

Message Strobe
Message PMT: (() . #[0001]
Period (ms): 500

PDU Message Lambda Block
function: <funct...460e03500>

PyQT Real Time Plot
Plot Style: Line
Label: TX Bitstream

PDU Message Lambda Block
function: <funct...460e03a28>

Note
Note: Interrogation Preamble

Note
Note: DBPSK Mod

Note
Note: Differential Encoding

Note
Note: P6 Preamble

Burst Preamble Insert
Preamble Bits: [1,1,1,0,0,...

PDU Message Lambda Block
function: <funct...460e03410>

Async. Differential Encoder
Modulus: 2

Burst Preamble Insert
Preamble Bits: 0, 0...0, 1, 0

Note
Note: interpol...s per symbol

PDU Message Lambda Block
function: <funct...460e037d0>

EventStream Source
Handler Threads: 1
Early Behavior: ASAP
Merge Mode: MEMCPY

Throttle
Sample Rate: 4M

Char To Float
Scale: 1

Add

Noise Source
Noise Type: Gaussian
Amplitude: 0
Seed: 0

Note
Note: DBPSK Demod

Note
Note: decimate...s per symbol

PDU Message Lambda Block
function: <funct...460e03b18>

PDU Message Lambda Block
function: <funct...460e03aa0>

Tag Consumer
Sample Rate: 4M
High Threshold: 200m
Low Threshold: -200m
Expected Packet Size: 112-bit
Debug Mode: OFF

Interrogation Framer
Sample Rate: 4M
High Threshold: 200m
Low Threshold: -200m
Debug Mode: OFF

QT GUI Time Sink
Name: TX DBPSK Symbols
Number of Points: 1.024k
Sample Rate: 4M
Autoscale: No

Note
Note: Differential Decoder

Note
Note: Preamble Trim

PDU Message Lambda Block
function: <funct...460e03b90>

PDU Message Lambda Block
function: <funct...460e03c08>

PyQT Real Time Plot
Plot Style: Line
Label: RX Bitstream

Figure 6.12: A flowgraph for testing the communication interfaces for interrogation messages without USRP hardware.

can store the address transmitted in the squitter and begin interrogations of that address. When the AA field is equal to the bottom core's address, the bottom core will recognize the interrogations and reply accordingly.

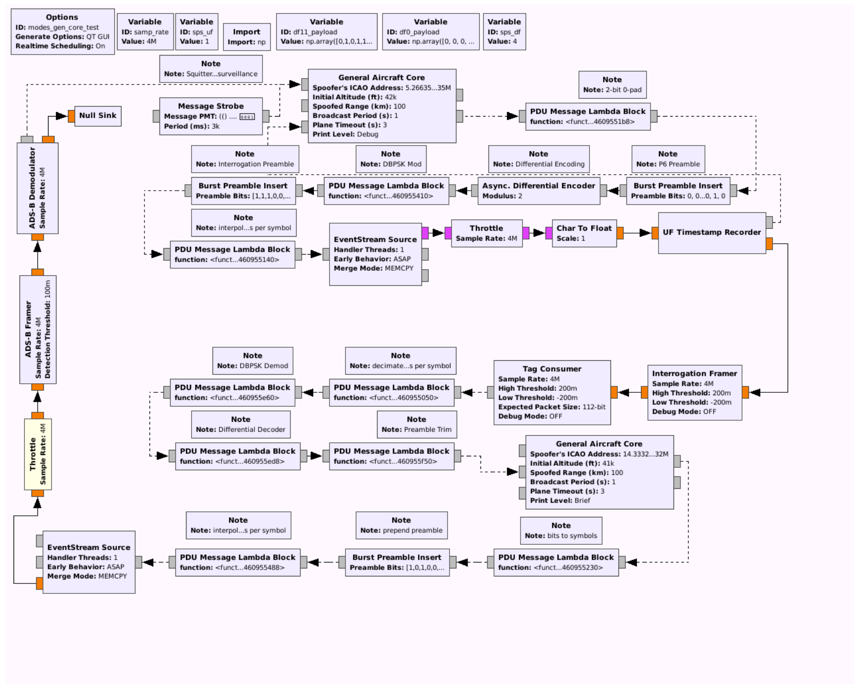


Figure 6.13: Two general aircraft cores communicating to each other through all four interfaces without USRP hardware.

Chapter 7

Attack Design Verification

Verification is ensuring that a design meets its requirements. The design in Chapter 6 must be verified, and Section 6.1 defines the requirements of which to verify against. If the design works as expected, then the generation of phantom aircraft is sufficiently verified. While the direct testing approach would be to generate aircraft on a real TCAS, real TCAS hardware is not trivial to acquire and setup. Therefore, verification is completed when two simplified aircraft cores described in Chapter 6 can track each other at zero speed through the medium of Mode S messages. In this chapter, the verification goals are made explicit, and the results of the verification are explored.

7.1 Verification Goals

In this section, verification points are outlined. This section demonstrates the flowgraphs which perform the functions of a simplified aircraft generator. While insufficient for the phantom aircraft attack of Chapter 5, generating and maintaining a false track is essential for the phantom aircraft attack regardless.

The systems under test are the flowgraphs developed in Section 6.4. A series of unit tests an integration tests through GNU Radio are performed to accomplish the tasks outlined in Chapter 6. The goals of the verification are to demonstrate the following:

1. Arbitrary Mode S reply messages (see Figure 2.12) can be crafted to fool the `gr-adsb` receiver without USRP hardware in-the-loop [29]). This is verified by the flowgraph in Figure 6.10.

2. The `gr-adsb` receiver's performance can be characterized via a packet-loss calculation with USRP hardware in-the-loop (see Figure 6.9). This is verified by the flowgraph in Figure 6.11.

3. Arbitrary Mode S interrogation messages (see Figure 2.10) can be crafted to fool a custom Mode S interrogation receiver (see Figure 6.5) without USRP hardware in-the-loop. This is verified by the flowgraph in Figure 6.12.

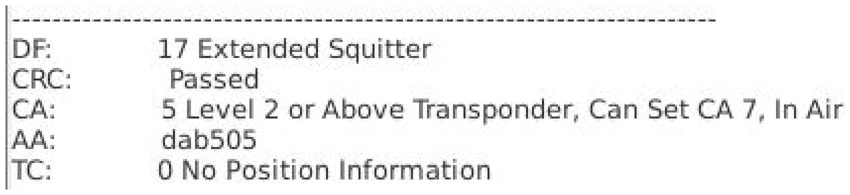
4. All four message modulators and demodulators can deliver payloads to the `core logic`. The core logic performance is indicative of the state machine functionality of Figure 6.7. No USRP hardware is used in-the-loop. This is verified by the flowgraph in Figure 6.13.

For each point of verification, the flowgraph is executed for 10 minutes with a fixed surveillance period of one second such that approximately 600 rounds of interrogations and replies occur for each item. In the Section 7.2, the verification procedures are discussed in detail; Chapter 8 provides insights on these results from a design, security, and future works perspective.

7.2 Verification Results

While this section is mainly concerned with integration testing between custom blocks and other OOT modules, all custom written blocks are unit tested through GNU Radio's `gr_unittest` framework before integration within GRC [22]. Unit testing through `gr_unittest` allows for flowgraph behavior without the GUI environment to quickly verify a block's atomic functionality.

7.2.1 Mode S Reply Functional Verification



```
------------------------------------------------------------------
DF:         17 Extended Squitter
CRC:         Passed
CA:         5 Level 2 or Above Transponder, Can Set CA 7, In Air
AA:         dab505
TC:         0 No Position Information
```

Figure 7.1: Console output of the `ADS-B Decoder` [29] demonstrating that the received DF 17 message is formatted as expected.

The first verification performed is on the flowgraph shown in Figure 6.10. This flowgraph implements the block diagrams of Figures 6.2 and 6.3 without using USRP hardware. For this verification, the essential functionality to observe is that a crafted payload message is modulated, demodulated, and decoded properly. The message is also converted from asynchronous data into streamed data and back again such that synchronization is also tested.

This flowgraph uses a custom *strobe* block that periodically transmits a reply message once a second. The chosen message is a Mode S extended squitter [35] which has the following parameters:

- A header indicating a `DF` of `17`
- A CA set to `5`
- An AA set to the hexadecimal value `0xdab505`
- No ADS-B data (i.e. 56 zeros)

- A 24-bit CRC value

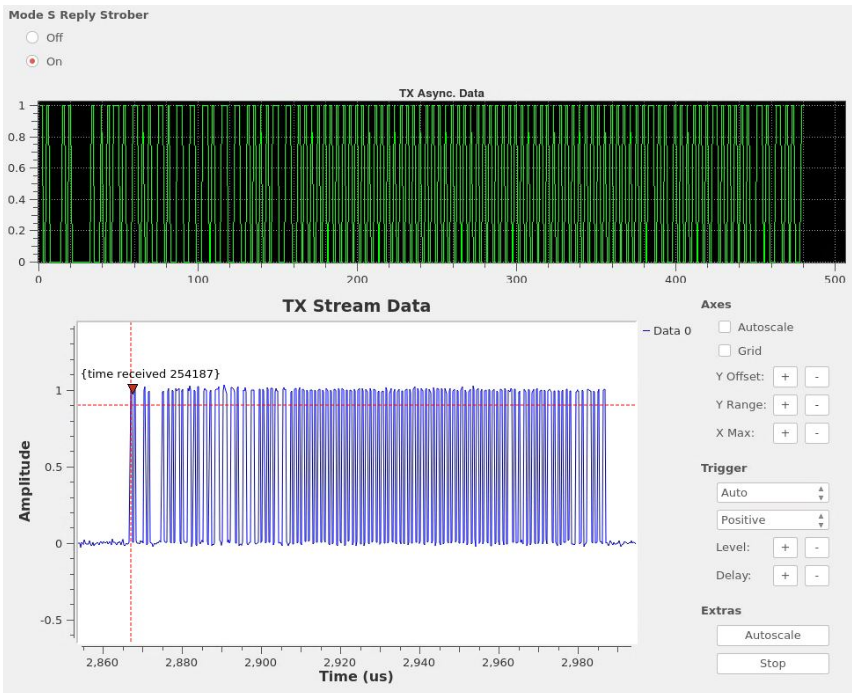


Figure 7.2: Real-time graphs of Figure 6.10 showing the DF 17 message received (and decoded in Figure 7.1).

The flowgraph is ran at a **2 mega-samples per second** rate through a simulated AWGN channel. The message is received as expected as demonstrated by the decoder output of Figure 7.1. Two graphs are also produced: a graph of the transmitted data and a graph of the `ADS-B Framer` locating the start of the preamble. These graphs are shown in Figure 7.2.

7.2.2 Mode S Reply Performance Characterization

The second verification is to characterize the performance of the USRP setup shown in Figure 6.9. Specifically, an examination of this software's susceptibility to **packet loss** in its current state is taken. The examination is performed with the flowgraph of Figure 6.11, and with it, inferences can be made on where to improve the phantom aircraft's performance.

The effect on packet loss in this system is observed as a function of transmission gain. The N210's receiver gain is fixed at 31 dB based on its WBX daughter-board specifications [47]. The B210's initial transmission gain is chosen to be more than half of its specified TX capabilities [45], and the variable attenuator is fixed at 30 dB.

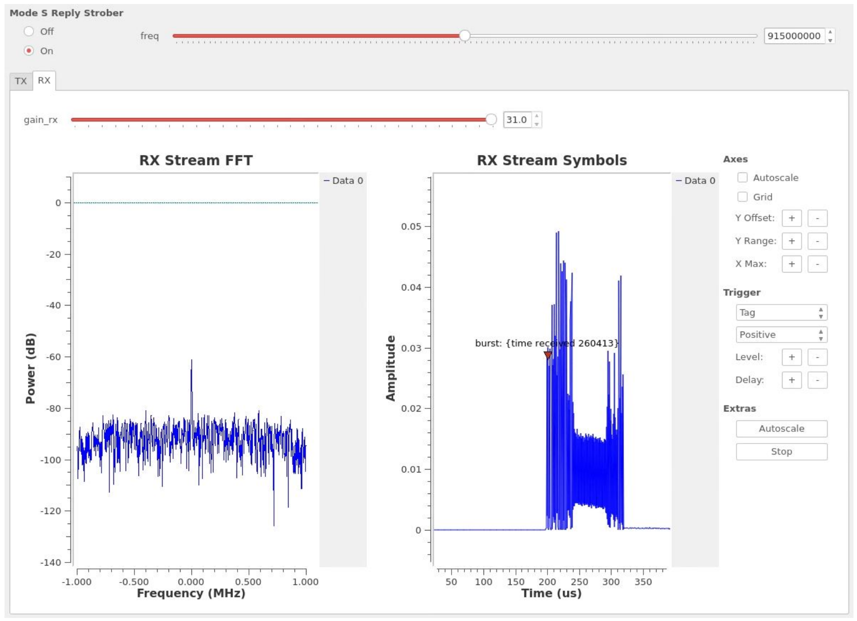

Mode S Reply Strober
Off [✓ On]
freq 915000000
TX RX
gain_rx 31.0

Figure 7.3: Real-time and FFT plot of the receiving USRP that is synchronizing a DF 17 message.

Then, the USRP flowgraph of Figure 6.11 is ran at a 2 mega-samples per second rate. The Mode S replies transmit once per second over the cable on the 915 MHz frequency band. Figure 7.3 shows a real-time plot of the Mode S reply being recognized by the `ADS-B Framer` at run-time. The particular case shown in that figure uses a transmission gain of 46 dB and a reception gain of 31 dB.

Six different 10-minute test runs are then performed. For each test, the TX gain is noted. Then the amount of messages sent and messages received are counted. A received message is defined as a message that **passes its CRC check**; messages that are missed entirely, are the wrong format, or do not pass the CRC check are counted as a dropped packet. No

Test Run	TX Gain (dB)	RX Gain (dB)	Messages Sent	Messages Received	Packet Loss
1	46	31	628	265	57.80%
2	44	31	610	222	63.61%
3	42	31	611	113	81.51%
4	48	31	614	248	59.61%
5	50	31	628	237	62.26%
6	52	31	609	131	78.49%

Table 7.1: Tests performed over the USRP hardware as shown in Figure 6.9 and their respective results.

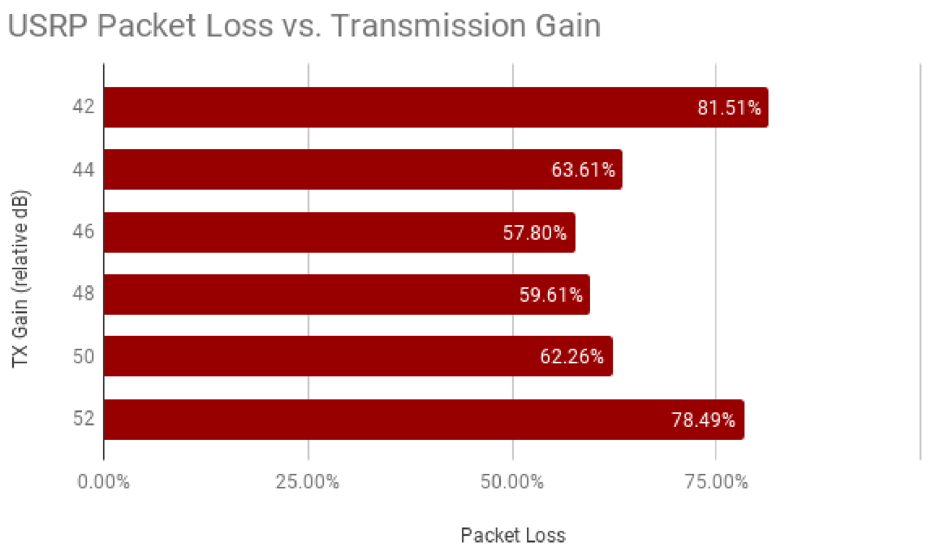


Figure 7.4: Graph of percent packet loss versus TX gain.

distinction is made between these three modes of packet loss.

Table 7.1 summarizes each test case and their respective packet loss performance. The data of Table 7.1 is also realized in a graph shown in Figure 7.4. Even the best performing case still has a packet-loss factor nearly 60%. While this is a significant problem for hardware testing, this thesis's main focus is its emulation of TCAS functionality. Therefore, the remainder of the tests ignore the use of USRP hardware.

7.2.3 Mode S Interrogation Functional Verification

This verification is similar to the verification performed on the Mode S reply flowgraph. The flowgraph in question is shown in Figure 6.12; it implements the block diagrams of Figures 6.4 and 6.5 without using USRP hardware.

However, unlike the testing of Mode S replies, an arbitrary payload of **112 bits** is modulated and demodulated to conform an expected Mode S interrogation. The message is converted from asynchronous data into streamed data and back again through a simulated AWGN channel. This method tests the Mode S interrogation receiver implementation as well as its robustness to AWGN noise.

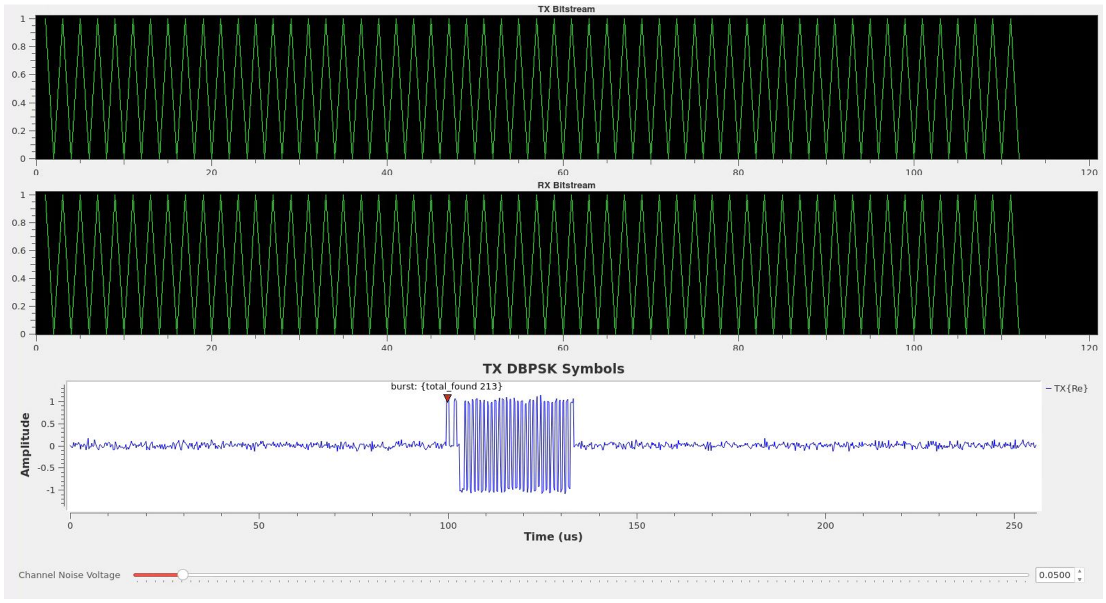


Figure 7.5: Three real-time plots of the interrogation interface test. The top plot shows the asynchronous TX bit stream, the middle plot shows the asynchronous RX bit stream, and the lower plot shows the real-time plot of the `framer` marking the start of the message.

Plots of the real-time data are compared to verify data integrity. These plots are shown in Figure 7.5. The top plot shows the asynchronous data transmitted before modulation, and the middle plot shows the asynchronous data received after demodulation. The lower plot shows the real-time data stream where the `framer` has identified the start of the Mode S interrogation preamble.

7.2.4 Core Logic Functional Verification

This verification intends to demonstrate that the core logic can behave like the state machine shown in Figure 6.7. As is discussed in Chapter 6, demonstrating that all three states are reachable in this software-only environment should be sufficient for proving the base-level functionality of this phantom aircraft generator.

The flowgraph being verified is shown in Figure 6.13; it is designed such that one aircraft (the top core) will declare the other aircraft a threat when the declaration conditions are met. Also, all four communication interfaces are used, so this flowgraph therefore acts like a sufficient integration test for *all* communication interfaces running simultaneously.

The top aircraft core is referred to as **Aircraft One**, and the bottom aircraft core is referred to as **Aircraft Two**. Aircraft One has the parameters shown in Table 7.2. Likewise, Aircraft Two has the parameters shown in Table 7.3.

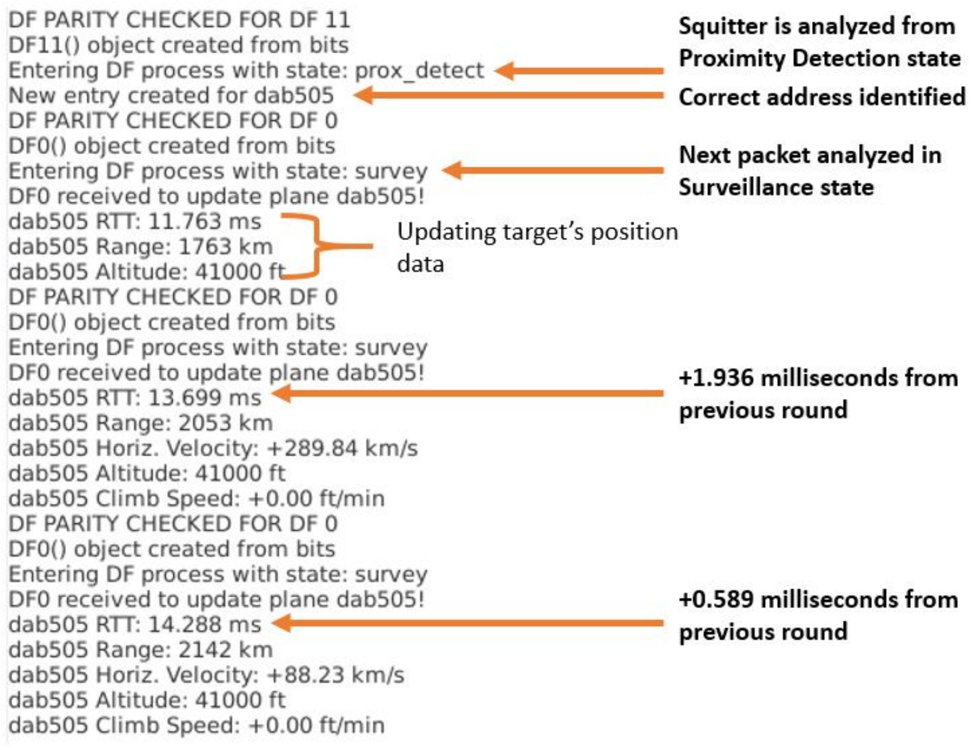
Item	**Value**	**Description**
ICAO Address	0x505bad	This aircraft's ICAO address
Altitude	42,000 ft	Initial spoofed altitude
Range	*Not applicable*	Adds an artificial delay before replying to appear a greater distance away
Surveillance Period	1 second	Aircraft is interrogated once per this period
Intruder Timeout	3 seconds	If an aircraft is not responsive for this many seconds, its entry is removed.
Print Level	Debug	Prints all internal processing information

Table 7.2: Parameters for the core logic of Aircraft One

Since neither aircraft transmits a squitter, an additional periodic transmitter is added to Aircraft One's reply DF RX interface to initialize the encounter. However, although altitude

Item	Value	Description
ICAO Address	0xdab505	This aircraft's ICAO address
Altitude	41,000 ft	Initial spoofed altitude
Range	0 km	Adds an artificial delay before replying to appear a greater distance away
Surveillance Period	*Not applicable*	Aircraft is interrogated once per this period
Intruder Timeout	*Not applicable*	If an aircraft is not responsive for this many seconds, its entry is removed.
Print Level	Brief	Prints no internal processing information

Table 7.3: Parameters for the core logic of Aircraft Two



```
DF PARITY CHECKED FOR DF 11
DF11() object created from bits
Entering DF process with state: prox_detect
New entry created for dab505
DF PARITY CHECKED FOR DF 0
DF0() object created from bits
Entering DF process with state: survey
DF0 received to update plane dab505!
dab505 RTT: 11.763 ms
dab505 Range: 1763 km
dab505 Altitude: 41000 ft
DF PARITY CHECKED FOR DF 0
DF0() object created from bits
Entering DF process with state: survey
DF0 received to update plane dab505!
dab505 RTT: 13.699 ms
dab505 Range: 2053 km
dab505 Horiz. Velocity: +289.84 km/s
dab505 Altitude: 41000 ft
dab505 Climb Speed: +0.00 ft/min
DF PARITY CHECKED FOR DF 0
DF0() object created from bits
Entering DF process with state: survey
DF0 received to update plane dab505!
dab505 RTT: 14.288 ms
dab505 Range: 2142 km
dab505 Horiz. Velocity: +88.23 km/s
dab505 Altitude: 41000 ft
dab505 Climb Speed: +0.00 ft/min
```

Squitter is analyzed from Proximity Detection state
Correct address identified
Next packet analyzed in Surveillance state
Updating target's position data
+1.936 milliseconds from previous round
+0.589 milliseconds from previous round

Figure 7.6: Imprecise RTT measurements from the design under test (see Figure 6.13).

information is as expected, the computational overhead is too great and imprecise for range estimation.

The imprecise ranging behavior is shown in the output of the system in Figure 7.6. The state transitions from `Proximity Detection` into `Surveillance`, and simulated range is estimated. However, the RTT varies on the order of milliseconds. In order to meet the resolution requirements of 15 meters or better [34, 35], the ranging estimation requires a RTT measurement precision on the order of $10^{-}8$. Therefore, further investigation into improving the RTT estimation in this system is required.

Chapter 8

Discussion

A discussion of various implications from this thesis is presented in this chapter. In particular, a discussion on the phantom aircraft generator's performance improvement opportunities, potential attack countermeasures, and future work opportunities are presented.

There are three discussions with distinct themes to each. First, a discussion on the design and results presented in Chapters 5, 6, and 7 are done in Section 8.1. Second, the inferences drawn from this thesis on TCAS from a security perspective are made explicit in Section 8.2. Third and finally, the opportunities for future work are discussed in Section 8.3.

8.1 Design and Implementation

This discussion concerns the design and testing of the system proposed in Chapters 5, 6 and 7. These chapters propose a method for attacking a TCAS encounter. The proposed attack is called the **Phantom Aircraft** attack where the attacker creates the ideal conditions for TCAS inducing a mid-air collision. The attack implementation is a simplified foundation of the Phantom Aircraft where the intended outcome is the generation of non-malicious false aircraft that a TCAS would be tricked into tracking.

The first test performed in Section 7.2.1 aims to verify that the integration of Mode S reply modulation and demodulation in GNU Radio accurately reflects the behavior of real Mode

S reply transmission and reception. In this particular test, a single type of Mode S reply is used to fool the `ADS-B Decoder`; it is reasonable to assume that any other message can be crafted to fool the decoder. Therefore, since the OOT `ADS-B Decoder` could recognize the crafted reply message and any changes to its parameters, the function of Mode S reply modulation is verified (see Figure 7.2).

The second test performed in Section 7.2.2 aims to use similar software as that used in the first test. However, the simulated noise channel is exchanged with real USRP hardware that communicates over a wire. The purpose of this test is to verify how feasible it would be to test a phantom aircraft attack over two USRPs.

The results of the second test are concerning with respect to the feasibility of hardware-in-the-loop testing with the current hardware setup (see Figure 6.9). The packet loss of the system was investigated as a function of transmission gain - the basis being that there is an optimal balance between gain and signal distortion that must be met. However, the best performing case of the tests performed still showed a packet loss of about 57%.

For designing an optimal TCAS, the intuition would be to immediately investigate solutions for improving the packet-loss performance. However, when performing an attack, it is important to consider the cost of optimizations; attacks can still be carried out in non-ideal circumstances because the overall attack only has to work *once* in practice.

Information on the packet-loss performance of TCAS hardware could provide insights into how well the attack setup must perform. However, as shown in Table 3.1, the safety study only considers the risk ratio of the probability that TCAS drops a track on a particular aircraft [40]. It remains unclear whether such a low risk ratio is attributed to minimal packet loss or the tracking algorithm's robustness to high-loss environments. Therefore, further investigation into TCAS's packet loss performance should be conducted in order to influence the hardware-in-the-loop test design.

Similar to the first test, the test of Section 7.2.3 has the same goal but within the context of interrogation interfaces. Rather than limit the test to the transmission of Mode S interrogations, an arbitrary payload is used to verify that data integrity is maintained through the modulation and demodulation processes. Since the data integrity is maintained through a transmission and reception, Mode S interrogation modulation and demodulation is deemed sufficiently functional (see Figure 7.5).

The fourth and final test is shown in Section 7.2.4. However, the results of that verification remain inconclusive because of the imprecise estimation of RTT. During the design of the range estimator, it was assumed that the software could be sufficiently precise enough to make accurate RTT estimations; this assumption is shown to be untrue, however. The software cannot precisely estimate RTT in its current state, and other analytical approaches have shown this approach to be problematic [66].

Similar to the discussion on packet loss, it remains unclear how precise these estimations need to be in order for a successful attack to occur. While it might be unnecessary for an attacker to meet the 15 meter resolution requirement of TCAS, the attacker's timestamp precision must be on the order of hundreds of nanoseconds in order to realize a range resolution of hundreds of meters.

Coarse analysis reveals little about the performance of the spoofing system. Two different coarse analysis techniques are used: system-wide profiling with `htop`, and application profiling with built-in Python profiling.

For system-wide profiling, `htop` is used to verify that the hardware is not being saturated. When the spoofer is running, `htop` shows an average load of less than 1.5 for a 12 core CPU; this is a significantly lower load than what the CPU can handle. However, the spoofer is not under constant load, and, therefore, the statistics are not indicative of CPU load during short bursts of data.

Since `htop` profiling remains inconclusive, another coarse analysis is performed using built-in Python performance profiling tools. However, because the GNU Radio scheduler is not Python-based, it remains invisible to the Python profiling software. Tests of various lengths on the spoofer are performed, but the only difference in execution time is for how long the main Python script executes. There are no increases to total function calls across these tests, and a majority of functions are invoked only once.

Because the coarse analysis techniques are both inconclusive, fine analysis tools may be necessary in order to accurately profile the software. GNU Radio provides two tools which could achieve this: performance counters and `ControlPort` [67]. Both of these tools are designed to give the developer better insights into the performance of a particular block in a GNU Radio flowgraph.

However, these performance tools seem to be incompatible with the spoofer of this project. First, performance counters are statistics designed for streams, not packets. Statistics like CPU time are relative to the execution of the block when it uses routines controlled by the GNU Radio scheduler, which asynchronous-only blocks do not have.

Furthermore, ControlPort tools are not enabled by default and are not part of the latest *stable* build of GNU Radio. A custom build of GNU Radio with other external packages is required to enable ControlPort profiling. However, even if ControlPort is functional, it remains unclear as to how much conclusive data it can produce for the spoofer flowgraph because it only supports the probing of streamed data.

Although performance profiling remains inconclusive, there may still be opportunities to re-design a better performing GNU Radio spoofer. In order for a practical phantom aircraft attack to occur, some optimization is necessary. There are four possible approaches that could be taken to improve the ranging performance: First, identifying and mitigating the highest overhead in the current software could remove imprecision. Second, creating a different architecture approach that does not use asynchronous messages would enable the use of more GNU Radio performance profiling tools. Third, a novel approach to costly algorithms like correlation could be utilized [5]. Fourth, RTT estimation can be implemented on the SDR FPGA as a co-processor, thereby bypassing the scarce real-time guarantees that GNU Radio provides [66].

In summary, the implementation of a non-malicious *phantom aircraft* generator is not fully realized yet. There is opportunity to implement a more sophisticated spoofer with greater real-time constraints by developing modules that more closely follow the GNU Radio idiosyncrasies, removing expensive message interfacing from the architecture, and possibly implementing a high-accuracy time-stamping in the SDR's FPGA. However, in the software's current state, other simpler attacks are realizable because they do not rely on the attacker's ability to create an accurate range estimate like the DoS attacks explained in Chapter 4. Nevertheless, a sophisticated attack like the malicious *Phantom Aircraft* requires additional work to realize on an SDR.

8.2 Security Implications

In this discussion, inferences on TCAS's security are presented. In summation, TCAS is not found to be a secure system. The difficulties had in attacking TCAS can be attributed to its nature as a highly complex system that is difficult to learn about - a phenomenon colloquially known as "security through obscurity" [27]. However, obscure systems only have time as its defense; those with the time and resources to attack a complicated system will make it work eventually. Therefore, it is imperative that those who can defend these systems to use this time to their own advantage.

To start, much of the analysis on the security of TCAS done in Chapters 3 and 4 is revisited, and the concepts behind the attack design in Chapters 5, 6, and 7 are explored to infer how vulnerable TCAS is to external manipulation. Then, potential countermeasures to the ideal *Phantom Aircraft* attack are explored.

First, the analysis of Chapters 3 and 4 should provide ample evidence to suggest that TCAS is not a secure system. Although TCAS has ample robustness to failures in a cooperative environment, it is evident from CIA analysis and modeling adversarial failure that many of these safety features and studies do not account for an adversary in its design. More work on redesigning critical components of TCAS is required to secure the weaknesses identified in this thesis's analysis.

Second, although the sophisticated *Phantom Aircraft* attack is not yet fully realized, there are much simpler attacks an adversary could perform right now. This basic implementation provides ample features to carry out DoS-style attacks on TCAS. As explained in Chapter 4, an attacker can simply exhaust the entire 24 bit ICAO address space to keep all nearby TCAS busy with acquiring false aircraft, thereby bypassing the need for precise range estimation.

However, let us assume that the ideal *Phantom Aircraft* attack is possible for sake of proposing countermeasures. Specifically, the style of defense to focus on is physical-layer defense because these defenses would require few to no changes to the Mode S protocol. Two styles of physical-layer defenses are identified which could be feasible for implementation in TCAS: **distance bounding** and **fingerprinting**. In distance bounding protocols, an upper-bound is placed on the distance between the parties in the network [43, 57]. The advantage to this defense is that the attacker **must** be within this distance to an aircraft's TCAS in order for its communications to be accepted; this limits the range window in which the attacker can operate. However, from a safety-critical system perspective, distance bounding may be infeasible to implement because it would affect the performance of intruder surveillance. An analysis on the trade-offs between the size of the distance bound and its impact on risk ratio would provide valuable insight.

In fingerprinting protocols, a responding waveform itself is analyzed for hardware "fingerprints" [41, 62]. In other words, only responses proven to be originating from authentic TCAS hardware would be accepted as a response. Ideally, this would thwart the attack in the sense that it could no longer be performed using SDR; an attacker would somehow need to carry out their attack using real TCAS hardware. However, fingerprinting techniques are often sensitive to environmental factors which may require frequent re-training of the fingerprint subroutines. Guarantees on minimum *false negatives* - the rejection of authentic TCAS signals - would be required for TCAS to maintain its safety-critical requirements.

In summary, the analyses of this thesis show TCAS is reasonably vulnerable to attack right now, and, although a sophisticated *Phantom Aircraft* attack is not fully realized against TCAS, attackers do have other options for attacking TCAS. More security analyses are required on TCAS in order to better understand how attacks and defenses could affect its performance.

8.3 Future Works

There are many opportunities for future work on security with respect to TCAS. Any solutions to TCAS's security would require a redesign of the system because of how specialized it is. Therefore, there are currently three main areas of improvement in which this section identifies: new analytical **safety studies**, improved or other TCAS **attack implementations**, and TCAS **defense**.

First, updated TCAS safety studies could be conducted to reanalyze TCAS from a non-cooperative perspective. In other words, a revision of the TCAS safety study [40] - with emphasis on a new FTA that considers adversarial failures - could provide more insights into how an attacker could affect TCAS in its current state. This future work is the least costly of the three as it would require no experimentation on the authors' part.

Second, more attacks could be demonstrated against TCAS. In general, successful attacks should motivate defense development. Especially in the context of safety-critical systems, any vulnerability that compromises the safety of on-board personnel and passengers should be a major motivator of change. The attacks could have novel approaches, or they could be improvements on the *Phantom Aircraft* attack.

Third and finally, demonstrating functional defenses are as important as demonstrating attacks. Further research into the countermeasures discussed in Section 8.2 is necessary with respect to implementation on a TCAS. These countermeasures are robust to the ideal *Phantom Aircraft* attacker and therefore warrant exploration. There is also opportunity for novel defenses for TCAS and for defending against less sophisticated attackers.

With any further TCAS research opportunity, it is also essential to develop a platform in which implementations can be tested on real TCAS hardware. TCAS cannot operate independently from its inputs; it has subroutines which disable the TCAS in the event that inputs are failing or nonexistent [15, 34, 35]. Therefore, designing a test bench in which TCAS can operate on real or simulated inputs would be invaluable to security work about it.

Chapter 9

Conclusion

An exploration on TCAS's vulnerabilities to exploitation is presented in this thesis. Overall, a wide breadth of analysis is done with respect to TCAS's security with the intention to motivate academic and industry-led research into the security of airborne collision avoidance. TCAS must have stronger security guarantees in this world of constant cyber-threat given its duty as a safety-critical system.

This thesis is an early effort to view TCAS from an adversarial perspective. To accomplish this goal, the following actions are taken. First, a TCAS safety study is analyzed from an adversarial perspective to quantify the effect of attacks on overall NMAC risk ratio. Second, attacks on TCAS are explored through the model of an *attack tree*. Third, an attack is chosen, and a threat model is defined to relay the attacker's goals and capabilities. Fourth, an implementation of a threat using GNU Radio is presented, and critical components of the implementation are tested. Fifth and finally, a discussion on various results are presented; the topics of these discussions are on the verification results produced, the inferred security implications, and the future work opportunities produced from this thesis.

Bibliography

[1] Jean Baker. Plane talk. https://web.archive.org/web/20010425005541/http://www.stuckmic.com/editorials/jeanbaker/planetalk.html, 1999. Accessed 2019-04-01.

[2] James Barron. Park slope plane crash | a collision in the clouds. https://cityroom.blogs.nytimes.com/2010/12/12/park-slope-plane-crash-the-lede-all/, 2010. Accessed 2019-04-01.

[3] P. Berthier, J. M. Fernandez, and J. Robert. Sat : Security in the air using tesla. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–10, Sep. 2017. doi: 10.1109/DASC.2017.8102003.

[4] J. Bowen. Formal methods in safety-critical standards. In *Proceedings 1993 Software Engineering Standards Symposium*, pages 168–177, Aug 1993. doi: 10.1109/SESS.1993.263953.

[5] R. Calvo-Palomino, F. Ricciato, B. Repas, D. Giustiniano, and V. Lenders. Nanosecond-precision time-of-arrival estimation for aircraft signals with low-cost sdr receivers. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 272–277, April 2018. doi: 10.1109/IPSN.2018.00055.

[6] Emily Chang, Roger Hu, Danny Lai, Richard Li, Quincy Scott, and Tina Tyan. The story of mode s: An air traffic control data-link technology. Technical report, Massachusetts Institute of Technology (MIT), 2000. Accessed 2019-04-01.

[7] Microprocessor Standards Committee. Ieee standard for binary floating-point arithmetic. Technical report, IEEE, 1985. Accessed 2019-04-06.

[8] The MITRE Corporation. Traffic alert and collision avoidance system. https://web.archive.org/web/20040803075017/http://www.caasd.org/proj/tcas/, 2004. Accessed 2019-04-05.

[9] Alexis England. 60 years ago, 2 planes collided over the grand canyon and it changed the world. https://www.azcentral.com/story/news/local/arizona-history/2016/06/30/60-years-ago-2-planes-collided-over-grand-canyon/86529858/, June 2016. Accessed 2019-03-28.

[10] EUROCONTROL. Acas ii equipage requirements. https://web.archive.org/web/20100421004042/http://www.eurocontrol.int/msa/public/standard_page/ACAS_Equipage_Requirements.html, March 2010. Accessed 2019-03-26.

[11] *ACAS Guide: Airborne Collision Avoidance Systems (incorporating TCAS II versions 7.0 & 7.1 and introduction to ACAS X).* EUROCONTROL, May 2016. Accessed 2019-04-03.

[12] Federal Aviation Administration (FAA). A brief history of the faa. https://www.faa.gov/about/history/brief_history/#origins, January 2017. Accessed 2019-03-28.

[13] *Pilot/Controller Glossary (P/CG).* Federal Aviation Administration (FAA), September 2009. Accessed 2019-04-05.

[14] *Advisory Circular 120-55C: Air Carrier Operational Approval and Use of TCAS II.* Federal Aviation Administration (FAA), February 2011. Accessed 2019-04-03.

[15] *Introduction to TCAS II Version 7.1.* Federal Aviation Administration (FAA), February 2011. Accessed 2019-03-28.

[16] Thomas K. Ferrell and Uma D. Ferrell. *RTCA DO-178B/EUROCAE ED-12B*, chapter 27. CRC Press, 2001.

[17] Laura Fitzgibbons. Checksum. https://searchsecurity.techtarget.com/definition/checksum, 2011. Accessed 2019-04-04.

[18] Laura Fitzgibbons. Security tip (st04-015): Understanding denial-of-service attacks. https://www.us-cert.gov/ncas/tips/ST04-015, 2018. Accessed 2019-04-04.

[19] FlightAware. Flightaware - flight tracker / flight status / flight tracking. https://flightaware.com/, 2019. Accessed 2019-03-31.

[20] The GNU Radio Foundation. Gnu radio - the free & open source radio ecosystem. https://www.gnuradio.org/, 2019. Accessed 2019-04-09.

[21] Y. Fujiwara. Self-synchronizing pulse position modulation with error tolerance. *IEEE Transactions on Information Theory*, 59(9):5352–5362, Sep. 2013. ISSN 0018-9448. doi: 10.1109/TIT.2013.2262094.

[22] *GNU Radio*. The GNU Radio Foundation, 2019. Accessed 2019-04-15.

[23] Eric Griffith. Two-factor authentication: Who has it and how to set it up. https://www.pcmag.com/feature/358289/two-factor-authentication-who-has-it-and-how-to-set-it-up, 2019. Accessed 2019-04-04.

[24] Tim Grobaty. Cerritos plane crash 30 years ago: "you either died or you didn't". https://www.presstelegram.com/2016/08/30/cerritos-plane-crash-30-years-ago-you-either-died-or-you-didnt/, 2017. Accessed 2019-04-01.

[25] Kanika Grover, Alvin Lim, and Qing Yang. Jamming and anti-jamming techniques in wireless networks: a survey. *International Journal of Ad Hoc and Ubiquitous Computing*, 17(4):197–215, 2014.

[26] Matthew Haughn and Stan Gibilisco. Confidentiality, integrity, and availability (cia triad). https://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA, June 2013. Accessed 2019-04-04.

[27] David Hayes. Security through obscurity is not security at all. https://wpshout.com/security-through-obscurity/, October 2017. Accessed 2019-04-23.

[28] C.M. Holloway. Towards understanding the do-178c / ed-12c assurance case. Technical report, NASA, 2012. Accessed 2019-04-09.

[29] Matt Hostetter. gr-adsb: Gnu radio oot module for demodulating and decoding ads-b packets. https://github.com/mhostetter/gr-adsb, July 2018. Accessed 2019-04-11.

[30] P. Hunt. *CRC CALCULATION FOR MODE-S TRANSPONDERS*. EUROCONTROL, November 1994. Accessed 2019-04-02.

[31] Clifton A. Ericson II. Fault tree analysis. Technical report, University of Central Florida, 1999. Accessed 2019-04-04.

[32] Christopher Ingraham. The safest — and deadliest — ways to travel. https://www.washingtonpost.com/news/wonk/wp/2015/05/14/the-safest-and-deadliest-ways-to-travel/?noredirect=on&utm_term=.fbd6dab7ea76, May 2015. Accessed 2019-03-26.

[33] National Instruments. Ettus research - the leader in software defined radio (sdr). https://www.ettus.com/, 2019. Accessed 2019-04-09.

[34] *Airborne Collision Avoidance System (ACAS) Manual.* International Civil Aviation Organization (ICAO), first edition, 2006. Accessed 2019-04-02.

[35] *Aeronautical Telecommunications: Annex 10 to the Convention on International Civil Aviation, Volume IV Surveillance and Collision Avoidance Systems.* International Civil Aviation Organization (ICAO), fourth edition, July 2007. Accessed 2019-04-02.

[36] *ARTICLE 1 - Terms and definitions.* International Telecommunication Union (ITU), 2009. Accessed 2019-04-16.

[37] Chong Hee Kim, Gildas Avoine, Francois Koeune, Francois-Xavier Standaert, and Olivier Pereira. The swiss-knife rfid distance bounding protocol. In *Information Security and Cryptology – ICISC 2008*, pages 98–115, 2008.

[38] Mykel J. Kochenderfer, Jessica E. Holland, and James P. Chryssanthacopoulos. Next-generation airborne collision avoidance system. *Lincoln Laboratory Journal*, 19(1):17–33, 2012. Access 2019-04-01.

[39] Kaspersky Lab. What is data encryption? https://usa.kaspersky.com/resource-center/definitions/encryption, 2017. Accessed 2019-04-04.

[40] John E. Lebron. System safety study of minimum tcas ii. Technical report, The MITRE Corporation, McLean, Virginia 22102, December 1983. DOT/FAA/PM-83/36.

[41] M. Leonardi and D. D. Fausto. Secondary surveillance radar transponders classification by rf fingerprinting. In *2018 19th International Radar Symposium (IRS)*, pages 1–10, June 2018. doi: 10.23919/IRS.2018.8448244.

[42] Aaron Margosis. Problems of privilege: Find and fix lua bugs. Technical report, Microsoft Corporation, 2016. Accessed 2019-04-04.

[43] Catherine Meadows, Radha Poovendran, Dusko Pavlovic, LiWu Chang, and Paul Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In *Secure localization and time synchronization for wireless sensor and ad hoc networks*, pages 279–298. Springer, 2007.

[44] Hugh Morris. How many planes are there in the world right now? https://www.telegraph.co.uk/travel/travel-truths/how-many-planes-are-there-in-the-world/, August 2017. Accessed 2019-03-26.

[45] *USRP Hardware Driver and USRP Manual: USRP B2x0 Series*. National Instruments, 2018. Accessed 2019-04-13.

[46] *USRP Hardware Driver and USRP Manual: USRP2 and N2x0 Series*. National Instruments, 2018. Accessed 2019-04-13.

[47] *WBX - Ettus Knowledge Base*. National Instruments, 2019. Accessed 2019-04-15.

[48] The OpenSky Network. Opensky network: Free ads-b and mode s data for research. https://opensky-network.org/, 2019. Accessed 2019-03-31.

[49] Distribution Officer. Secondary surveillance radar mode s advisory circular. Technical report, International Civil Aviation Organization (ICAO), 1000 Sherbrooke Street West, Suite 400, Montreal, Quebec, Canada, 1983. CIRCULAR 174-AN/110.

[50] Tim O'Shea. gr-pyqt: pyqt based plotters intended for plotting bursted events in gnu radio. https://github.com/osh/gr-pyqt, July 2016. Accessed 2019-04-11.

[51] Tim O'Shea. gr-eventstream. https://github.com/osh/gr-eventstream, April 2017. Accessed 2019-04-11.

[52] Tim O'Shea and Kiran Karra. gr-burst: Burst psk modem building blocks for gnu radio. https://github.com/gr-vt/gr-burst, October 2016. Accessed 2019-04-11.

[53] Tim O'Shea and Kiran Karra. gr-mapper: Symbol to bit mapping and demapping blocks for gnu radio. https://github.com/gr-vt/gr-mapper, October 2018. Accessed 2019-04-11.

[54] *Python Anonymous/Lambda Function*. Parewa Labs, 2018. Accessed 2019-04-11.

[55] Bruce Schneier. Attack trees. https://www.schneier.com/academic/archives/1999/12/attack_trees.html, 1999. Accessed 2019-04-04.

[56] R. Shirey. Rfc 4949: Internet security glossary, version 2. Technical report, Network Working Group, 2007. Accessed 2019-04-04.

[57] D. Singelee and B. Preneel. Location verification using secure distance bounding protocols. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, pages 7 pp.–840, Nov 2005. doi: 10.1109/MAHSS.2005.1542879.

[58] SKYbrary. Airborne collision avoidance system (acas). Technical report, EUROCONTROL, 2019. Accessed 2019-04-01.

[59] SKYbrary. Minimum operational performance standards. Technical report, EUROCONTROL, 2019. Accessed 2019-04-03.

[60] SKYbrary. Ra downlink. Technical report, EUROCONTROL, 2019. Accessed 2019-04-01.

[61] Ian Sommerville. Critical systems. http://iansommerville.com/software-engineering-book/web/critical-systems/, 2019. Accessed 2019-04-03.

[62] Martin Strohmeier and Ivan Martinovic. On passive data link layer fingerprinting of aircraft transponders. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, CPS-SPC '15, pages 1–9, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3827-1. doi: 10.1145/2808705.2808712. URL http://doi.acm.org/10.1145/2808705.2808712.

[63] Symantec. What is a man-in-the-middle attack? https://us.norton.com/internetsecurity-wifi-what-is-a-man-in-the-middle-attack.html, 2018. Accessed 2019-04-04.

[64] *What is social engineering? Tips to help avoid becoming a victim.* Symantec Corporation, 2018. Accessed 2019-04-07.

[65] H. Tanaka, L. T. Fan, F. S. Lai, and K. Toguchi. Fault-tree analysis by fuzzy probability. *IEEE Transactions on Reliability*, R-32(5):453–457, Dec 1983. ISSN 0018-9529. doi: 10.1109/TR.1983.5221727.

[66] N. B. Truong, Y. Suh, and C. Yu. Latency analysis in gnu radio/usrp-based software radio platforms. In *MILCOM 2013 - 2013 IEEE Military Communications Conference*, pages 305–310, Nov 2013. doi: 10.1109/MILCOM.2013.60.

[67] Thomas W. Rondeau, Tim O'Shea, and Nathan Goergen. Inspecting gnu radio applications with controlport and performance counters. In *ACM SIGCOMM 2013 Proceedings of the second workshop on Software radio implementation forum*, 08 2013. ISBN 978-1-4503-2181-5. doi: 10.1145/2491246.2491259.

[68] Dave Warburton and Dan Goldman. Medical device safety system design: A systematic approach. https://www.mddionline.com/medical-device-safety-system-design-systematic-approach, 2019. Accessed 2019-04-03.

[69] H. Yang, M. Yao, Z. Xu, and B. Liu. Lhcsas: A lightweight and highly-compatible solution for ads-b security. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–7, Dec 2017. doi: 10.1109/GLOCOM.2017.8254500.

Appendices

Appendix A

TCAS Advisory Definition Tables

Own Altitude	**SL**	**Tau (s)**		**TVTHR (s)**	**DMOD (NM)**		**ZTHR (ft)**		**ALIM (ft)**
		TA	**RA**	**RA**	**TA**	**RA**	**TA**	**RA**	**RA**
<1,000	2	20	-	-	0.3	-	850	-	-
1,000-2,350	3	25	15	15	0.33	0.2	850	600	300
2,350-5,000	4	30	20	18	0.48	0.35	850	600	300
5,000-10,000	5	40	25	20	0.75	0.55	850	600	350
10,000-20,000	6	45	30	22	1.0	0.8	850	600	400
20,000-42,000	7	48	35	25	1.3	1.1	850	700	600
>42,000	7	48	35	25	1.3	1.1	1,200	800	700

Table A.1: Definitions of Sensitivity Level (SL), alarm thresholds (Tau, DMOD, ZTHR), and minimum safe vertical separation (ALIM) [11, 15].

RA Type	**Upward Sense**		**Downward Sense**	
	RA	**Required Vertical Rate (fpm)**	**RA**	**Required Vertical Rate (fpm)**
Positive (Corrective)	Climb	1,500 to 2,000	Descend	-1,500 to -2,000
Positive (Corrective)	Crossing Climb	1,500 to 2,000	Crossing Descend	-1,500 to -2,000
Positive (Corrective)	Crossing Maintain Climb	1,500 to 2,000	Crossing Maintain Descend	-1,500 to -2,000
Positive (Corrective)	Maintain Climb	1,500 to 2,000	Maintain Descend	-1,500 to -2,000
Negative (Corrective)	Reduce Descent	0	Reduce Climb	0
Negative (Preventative)	Do Not Descend	>0	Do Not Climb	<0
Negative (Preventative)	Do Not Descend > 500 fpm	>-500	Do Not Climb > 500 fpm	<500
Negative (Preventative)	Do Not Descend > 1,000 fpm	>-1,000	Do Not Climb > 1,000 fpm	<1,000
Negative (Preventative)	Do Not Descend > 2,000 fpm	>-2,000	Do Not Climb > 2,000 fpm	<2,000

Table A.2: Possible RAs against one threat for TCAS Version 7.1 [15].

TCAS Advisory	**Annunciation**
Traffic Advisory (TA)	“Traffic, traffic!”
Climb	“Climb, climb!”
Descend	“Descend, descend!”
Altitude Crossing Climb	“Climb, crossing climb; climb, crossing climb!”
Altitude Crossing Descend	“Descend, crossing descend; descend, crossing descend!”
RA Reversal to Climb	“Climb, climb now; climb, climb now!”
RA Reversal to Descend	“Descend, descend now; descend, descend now!”
Increase Climb	“Increase climb; increase climb!”
Increase Descend	“Increase descent; increase descent!”
Maintain Rate	“Maintain vertical speed, maintain!”
Altitude Crossing, Maintain Rate, Climb or Descent	“Maintain vertical speed, crossing maintain!”
Preventative	“Monitor vertical speed.”
RA Resolved	“Clear of conflict!”

Table A.3: Aural announcements TCAS reports to pilots during an advisory [15].

Appendix B

Expected Mode S Attack Messages

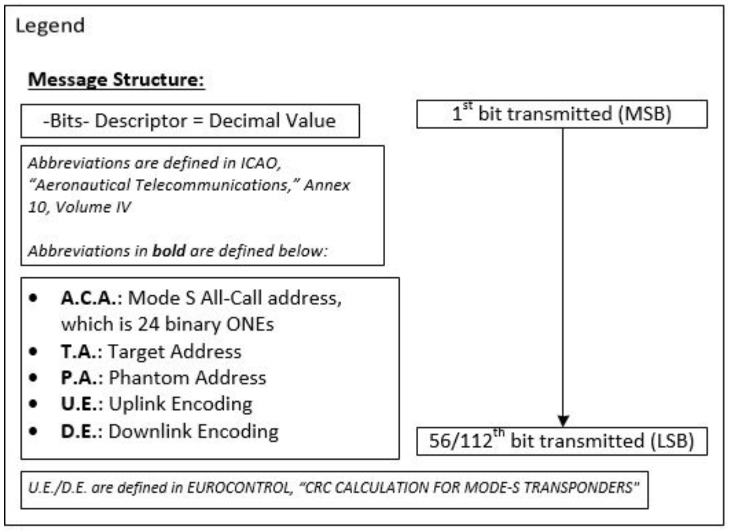

Legend
Message Structure:
-Bits- Descriptor = Decimal Value
Abbreviations are defined in ICAO, "Aeronautical Telecommunications," Annex 10, Volume IV
Abbreviations in bold are defined below:
A.C.A.: Mode S All-Call address, which is 24 binary ONEs
T.A.: Target Address
P.A.: Phantom Address
U.E.: Uplink Encoding
D.E.: Downlink Encoding
1st bit transmitted (MSB)
56/112th bit transmitted (LSB)
U.E./D.E. are defined in EUROCONTROL, "CRC CALCULATION FOR MODE-S TRANSPONDERS"

Figure B.1: Legend with respect to the message diagrams of Appendix B.

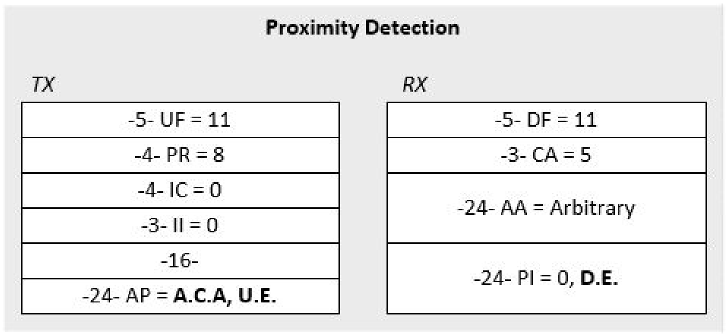
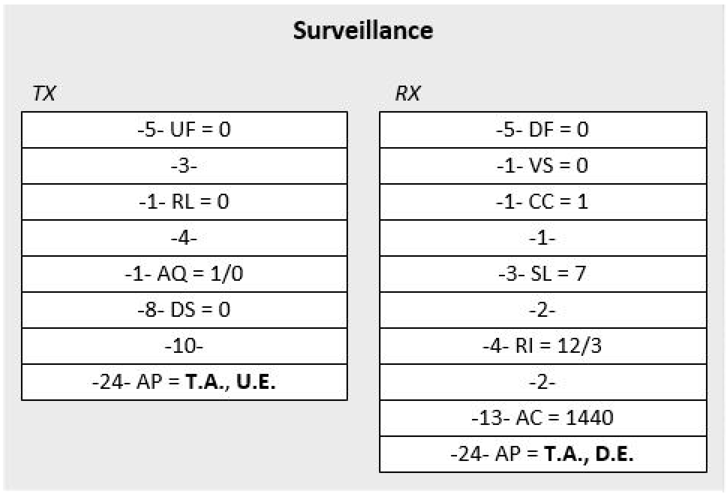


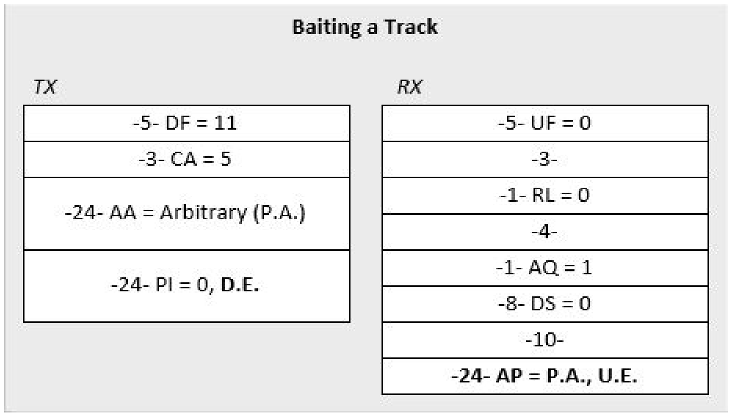
Figure B.2: Message transmitted and expected response for detecting nearby aircraft.



Surveillance

TX	*RX*
-5- UF = 0	-5- DF = 0
-3-	-1- VS = 0
-1- RL = 0	-1- CC = 1
-4-	-1-
-1- AQ = 1/0	-3- SL = 7
-8- DS = 0	-2-
-10-	-4- RI = 12/3
-24- AP = **T.A., U.E.**	-2-
	-13- AC = 1440
	-24- AP = **T.A., D.E.**

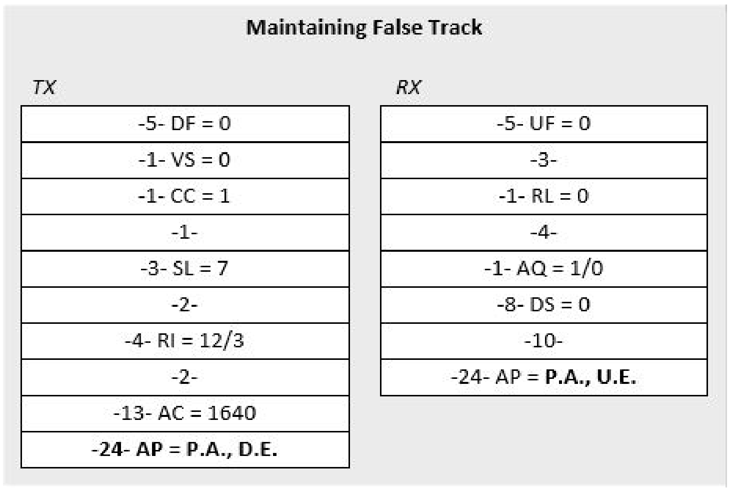
Figure B.3: Message transmitted and expected response for surveying a particular Target Aircraft.



Baiting a Track

TX	*RX*
-5- DF = 11	-5- UF = 0
-3- CA = 5	-3-
-24- AA = Arbitrary (P.A.)	-1- RL = 0
	-4-
-24- PI = 0, **D.E.**	-1- AQ = 1
	-8- DS = 0
	-10-
	-24- AP = P.A., U.E.

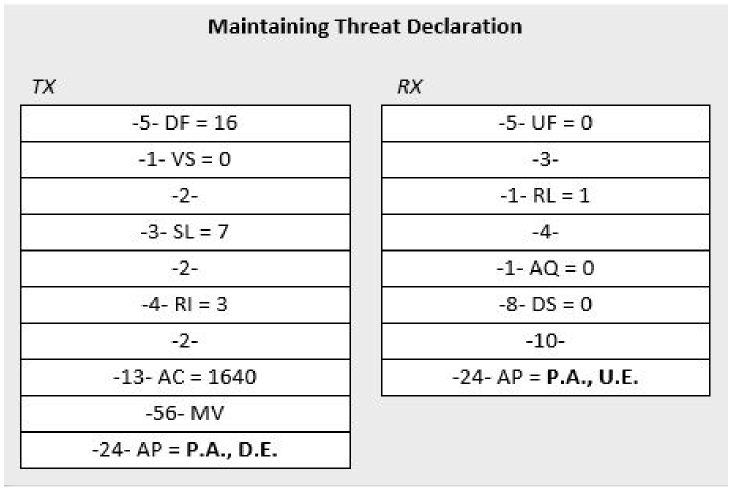
Figure B.4: Message transmitted and expected response for baiting interrogations from nearby aircraft.



Maintaining False Track

TX	*RX*
-5- DF = 0	-5- UF = 0
-1- VS = 0	-3-
-1- CC = 1	-1- RL = 0
-1-	-4-
-3- SL = 7	-1- AQ = 1/0
-2-	-8- DS = 0
-4- RI = 12/3	-10-
-2-	-24- AP = **P.A., U.E.**
-13- AC = 1640	
-24- AP = P.A., D.E.	

Figure B.5: Message transmitted and expected response when phantom aircraft is under normal surveillance from another aircraft.



Maintaining Threat Declaration

TX	*RX*
-5- DF = 16	-5- UF = 0
-1- VS = 0	-3-
-2-	-1- RL = 1
-3- SL = 7	-4-
-2-	-1- AQ = 0
-4- RI = 3	-8- DS = 0
-2-	-10-
-13- AC = 1640	-24- AP = **P.A., U.E.**
-56- MV	
-24- AP = **P.A., D.E.**	

Figure B.6: Message transmitted and expected response when phantom aircraft is under surveillance during a RA encounter.

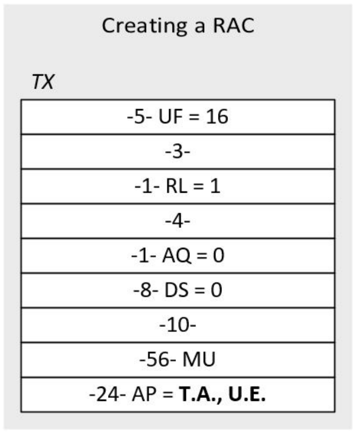


Figure B.7: UF 16 message that delivers a RAC to a target

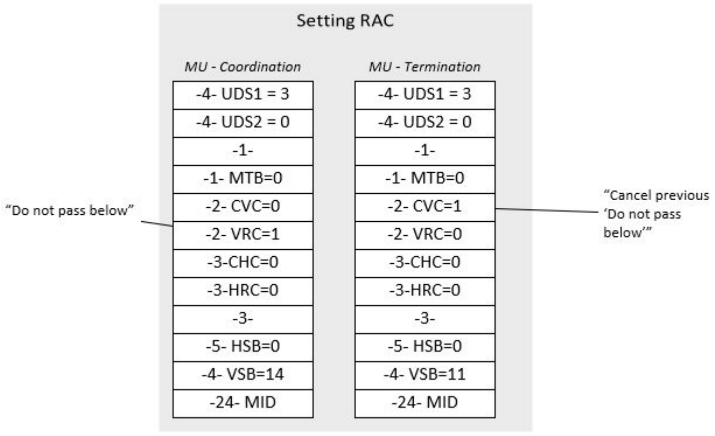


Figure B.8: Example of a RAC that creates or cancels a "Do not pass below" rule.

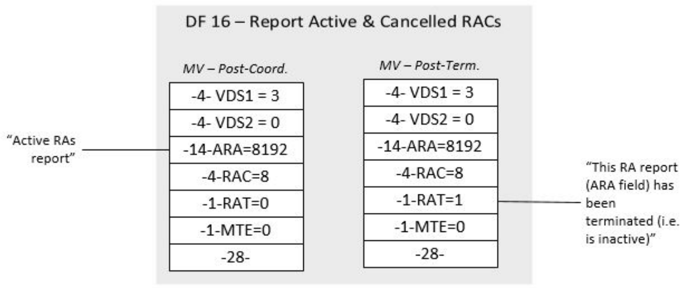


Figure B.9: Example output of a transponder interrogated with UF 0, AQ 0, and RL 1.