AENGM3100 Research Project

TACTICAL EXPLOITATION AND DEFENCE OF AIRCRAFT FLIGHT MANAGEMENT AND COMMUNICATIONS SYSTEMS

Department of Aerospace Engineering, University of Bristol, Queen's Building, University Walk, Bristol. BS8 1TR. UK.

May 29, 2015

Author:
Joseph Greenwood

Supervisors:
Dr. Daniel Page
Dr. Thomas Richardson

1 Abstract

This project demonstrates vulnerabilities in, and proposes defenses for, the Automated Dependent Surveillance-Broadcast system. These vulnerabilities allow false traffic to be injected with minimal skill, cost and attribution into the Traffic Collision Avoidance System aboard aircraft and thereby reduce the safety of air traffic. A partially complete offensive platform was developed using a HackRF software defined radio and false traffic injection was demonstrated using a custom python library and ZeroMQ sockets. A number of defensive prototypes were developed and tested in software including symmetric cryptography (using both a raw block cipher and authenticated encryption mode) and asymmetric cryptography (via elliptic curve digital signature verification). Whilst each of these defenses prevented injection, a combination of techniques would be required to practically mitigate the risk to air traffic, using both technical and policy defenses to provide confidentiality, integrity and authentication.

Keywords: *Avionics, Security, Exploitation, Cyber, Aircraft*

2 Introduction

2.1 Systems

This project concerns two primary systems; Automated Dependant Surveillance - Broadcast (ADS-B) and Traffic Collision Avoidance System (TCAS). Both of these systems are installed on medium to large civil and military aircraft, and are currently critical to the safe operation of an airframe.

2.1.1 ADS-B

ADS-B is a cooperative aircraft surveillance technology in which an aircraft determines it's position and state using the Global Positioning System (GPS), and then periodically broadcasts this, enabling tracking of that aircraft. Figure 1 below shows the ADS-B concept.

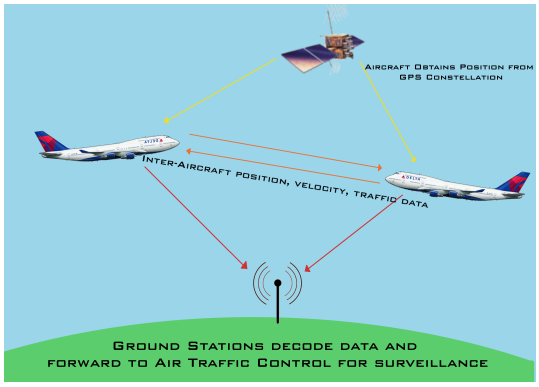

Figure 1: Diagram of ADS-B system

The system was introduced in 2007 under a Federal Aviation Authority (FAA) mandate[1], and is intended to replace radar as the primary surveillance mechanism in the United States Next Generation Air Traffic System (NextGen), and the European Single European Sky ATM Research (SESAR) projects. The system is mandatory in portions of Australian airspace, and will become so for certain aircraft types in Europe and the United States in 2017[2] and 2020[3] respectively.

The ADS-B system is used for multiple mission services, including flight information, terrain data, weather information and traffic reports. ADS-B is an extension of the Mode-S transponder protocol, and is sometimes referred to as 'extended squitter', due to messages that are twice as long as standard Mode-S (112 bits compared to 56 bits). The radios transmit primarily on 1090MHz using Pulse Position Modulation (PPM) encoding, but there is a separate system on 900MHz for General Aviation using Universal Access Transceivers. The average range of ADS-B transmissions is around 300 Nautical Miles (NM).

2.1.2 TCAS

TCAS is an aircraft collision avoidance system which started development in 1978 following a collision between a light aircraft and an airliner over San Diego[4]. The system monitors the area around it for Mode-S TCAS equipped aircraft, and issues warnings to pilots based on traffic.

Using multiple data feeds (including ADS-B), TCAS builds a 3 dimensional picture of the airspace around it, factoring in traffic range, bearing and altitude. It then uses this in conjunction with its own speed, bearing and altitude to determine if a risk of collision exists. Should an aircraft intrude on the 'protected area' around each TCAS equipped aircraft, then the system will compute and negotiate (for co-operative aircraft) mutual avoiding action. If the intruder is within 40s of collision, a 'Traffic Advisory' (or TA) is issued[4]. This does not require action, but informs the pilot of the risk of collision. Should the intruder be within 25s of collision, then a 'Resolution Advisory' (or RA) is issued. This requires the pilot to follow an avoiding action. RAs have a very high priority in the pilot's workflow, requiring immediate avoiding action within 2.5s. They also have priority over Air Traffic Control issued instructions[5]. Figure 2 below shows the protected areas around an aircraft. Figure 3 below shows the different warnings that are displayed to the pilot visually. TAs are displayed as orange circles, whilst RAs are displayed in solid red. These are accompanied by audible warnings, such as *'Climb, Climb"* or *'Descend, Descend"*

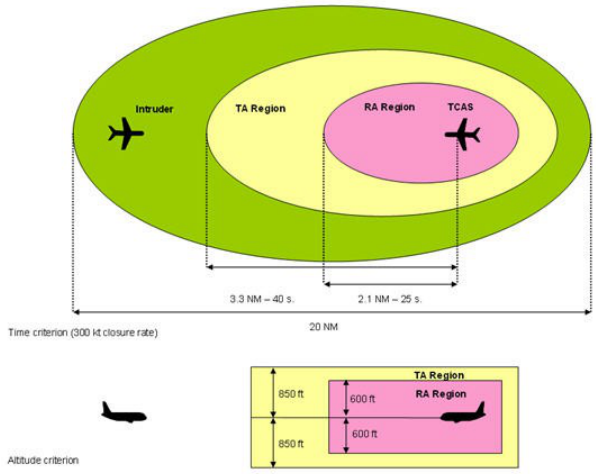

Figure 2: Protected areas around a TCAS equipped aircraft

Figure 3: TCAS display symbology

2.2 False Target Injection

Neither ADS-B nor TCAS has authentication, authorisation or encryption on any of its messages. This has led to multiple on-line trackers for aircraft[6], and an associated risk of airline surveillance. ADS-B messages can also be easily generated by software defined radios, as has been demonstrated previously by both the United States Air Force[7] and the wider hacker community[8][9]. This raises the possibility of a malicious attacker injecting aircraft into the Air Traffic Control picture. Whilst ground stations have methods of cross correlating signals using multiple

antennae, aircraft receiving ADS-B 'In' (i.e. using ADS-B to provide traffic warnings) do not have this option and are therefore vulnerable to injected 'ghost' aircraft. The impact of these attacks has been widely considered for ground infrastructure [10], where the risk can be reduced through correlating multiple sensors (i.e. Radar).

However the impact on individual aircraft and the interface to TCAS has been neglected, despite the catastrophic effects of a failure in collision avoidance. The injection of false targets into the air traffic network that aircraft can see, but that air traffic control cannot would wreak havoc in high density traffic areas. Distracting a pilot during a critical phase of flight could have fatal consequences. Despite the critical nature of the systems involved there are currently no defenses against this type of attack. In addition, there is no forensic capability to determine who launched an attack, or from where. This lack of attribution, combined with a low skill level required to launch an attack makes it attractive to a wide range of adversaries, from 'bedroom hackers' pulling a practical stunt, to organised terrorist groups, and even state-sponsored agencies.

2.3 TCAS Exploitation Pathway and Key Contributions

The aim of this research is twofold; to trigger a malicious resolution advisory by injecting ADS-B traffic, and to propose and demonstrate airborne mitigations against a malicious adversary. Neither of these objectives has been considered before. Figure 4 below shows the exploitation pathway to achieve this first objective.

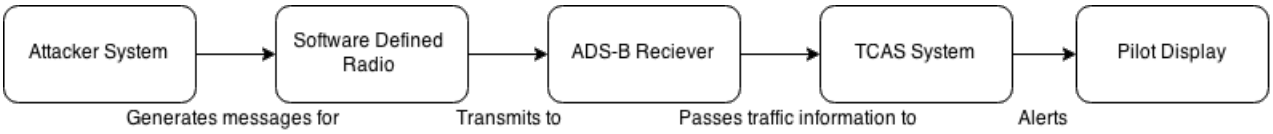

Figure 4: TCAS Exploit Pathway

A novel Python library will be created to generate arbitrary ADS-B messages to be transmitted using a software defined radio. To fulfil the second objective, a number of bespoke defensive methods have been prototyped in order to defend against this attack; the merits of each of these will be critically assessed.

- Symmetric encryption via a raw block cipher
- Symmetric authentication via a keyed Hash Message Authentication Code
- Symmetric authentication and encryption via an authenticated mode cipher
- Asymmetric authentication via elliptic curve digital signatures

The source code for all of the prototypes is available in the Appendices.

3 Methodology

3.1 ADS-B Testbed

Commercial ADS-B hardware can cost thousands of pounds per installation. In order to keep project costs down, software defined radios were used to simulate an integrated ADS-B and TCAS system. The radio used for receiving was an RTL2832U with a R820T2 chip, normally used to recieve digital TV. These radios can be tuned from 24 – 1766 MHz, and can provide raw I/Q samples over USB. This allows raw signals to be captured and processed in software, rather than hardware. These radios are available for around £10 online[11].

In order to do this processing, the GNURadio software development toolkit[12] provides pre-built signal processing blocks. It is regularily used in industry and academia as an alternative to expensive hardware, or to develop novel radio applications [13]. In this instance, it was used as the platform for a software Mode-S receiver, GR-AIR-MODES[14]. This system is written in Python and receives and parses Mode-S messages from multiple sources. Its results have previously been shown to be analogous to a hardware receiver[7]. Both GNURadio version 3.7.8 and GR-AIR-MODES were compiled from source from their respective GitHub repositories on a Kali Linux[15] penetration testing system.

In order to visualise successful injection and to integrate a TCAS system the open source flight simulator 'Flight-Gear' was used. This simulator contains a TCAS system that conforms to the TCAS II Version 7 standard[16]. This integrates with the simulated ADS-B receiver to provide traffic reports. In order to run the latest version of the simulator, supporting TCAS and ADS-B, the simulator was compiled from source[17].The final system is shown below in Figure 5.

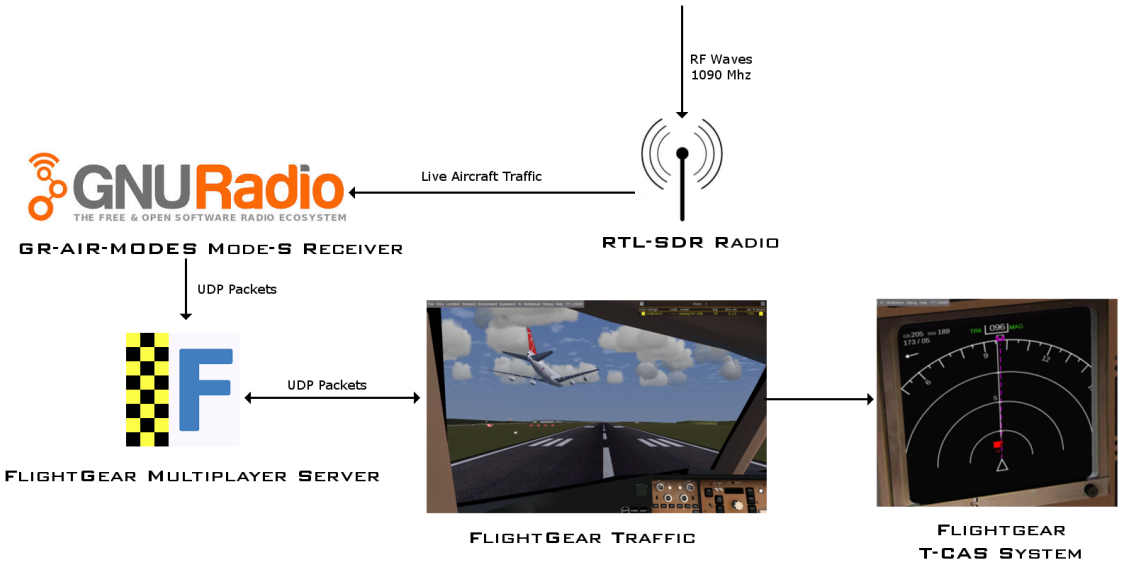

Figure 5: Diagram of ADS-B and TCAS Software testbed

This system was verified by listening to live ADS-B traffic in the Bristol area. Figure 6 below shows a virtual Easyjet flight EZY61GR mirroring its real equivalent.

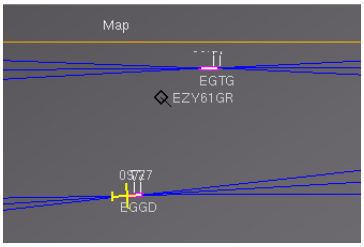

Figure 6: Easyjet flight EZY61GR displayed in the simulator

3.2 ADS-B Message Generation

ADS-B messages are Mode-S messages with an overall length of 112 bits and the structure below in Figure 7.

Figure 7: ADS-B Message structure

The downlink field (DF) of 5 bits describes the type of Mode-S message. In this case, the type number is 17 (for ADS-B). The aircraft address is the 24 bit hexadecimal representation of the unique ICAO address given to each aircraft. This is followed by the 56 bits of ADS-B data itself, and a 24 bit parity check that uses a CRC algorithm over the entire message. There are three key types of ADS-B message that we are concerned with, each referred to by the associated register in receiver memory.

3.2.1 Extended Squitter Airborne Position - BDS05

These messages have a format type code (FTC) of 0x5 and describe the position and altitude of the aircraft. The data structure is shown below in Figure 8.

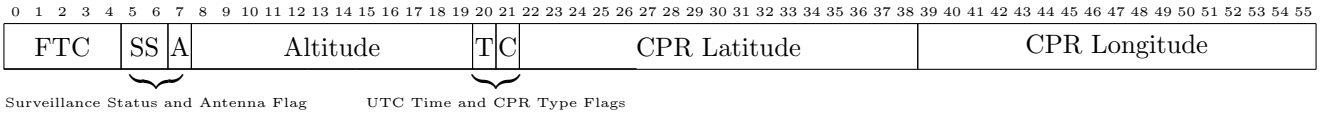

Figure 8: BDS05 Message Structure

The latitude and longitude of the aircraft are encoded using compact position reporting. This reduces the number of bits required to encode a latitude and longitude by referencing a location to a specific zone in the world[7]. In order to fully resolve the location of the aircraft, an 'odd' message and an 'even' message are transmitted, these are decoded together, and hence are the only stateful message in the ADS-B protocol.

3.2.2 Extended Squitter Identification and Category - BDS08

With an FTC of 0x8, these messages link the ICAO address to an eight character callsign and an aircraft category. This enables air traffic control to monitor aircraft using human-readable names. The aircraft category includes values such as 'Small', 'Large' and 'Parachutist'. Figure 9 below shows the message structure.

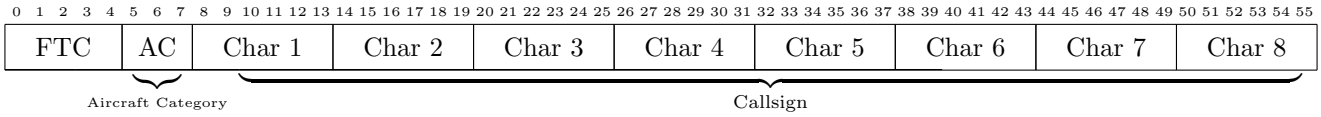

Figure 9: BDS08 Message Structure

3.2.3 Extended Squitter Airborne Velocity - BDS09

This message has a FTC of 0x9, and describes the aircraft velocity in north/south, east/west and vertical components. These messages allow aircraft to build a better picture of traffic movements.

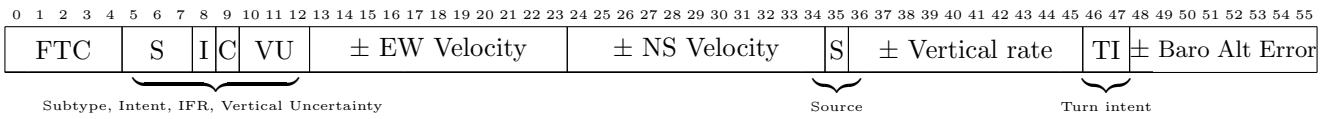

Figure 10: BDS09 Message structure

3.2.4 Python Library

The generation of all of the above messages is handled by a custom Python library. This library exposes the 'ADSBMessage' class, which is initialised below:

Listing 1: LibADSB usage

```
>> bds05data = [alt, latitude, longitude, cpr_type]
>> bds_code = 0x05
>> meven = libadsb.ADSBMessage(id, bds_code, bds05data)
>> print meven.hex
8dbeef114e3788aaaa53e929bc9b
```

This enables external applications to generate and use ADS-B messages. The library is available at Appendix A.

3.3 RF Transmissions

In order to inject false traffic into the ADS-B network, the method used by Capt. Magazu (USAF) in his 2012 thesis[7] was used. This uses a C program to generate ADS-B messages and encode them using PPM. This is then written to disk as a binary file. A set of GNURadio blocks are then used to decode the binary file and transmit using a software defined radio. In this instance, the radio used was a HackRF from Great Scott Gadgets[18], capable of transmission from 10MHz to 6GHz, and with a retail price of around £300.

In order to avoid interference with active aircraft, both the ADS-B testbed and the HackRF were tuned to transmit and receive on 433MHz, in the unlicensed Industrial Scientific and Medical (ISM) band. The GNURadio block diagram is shown below in Figure 11.

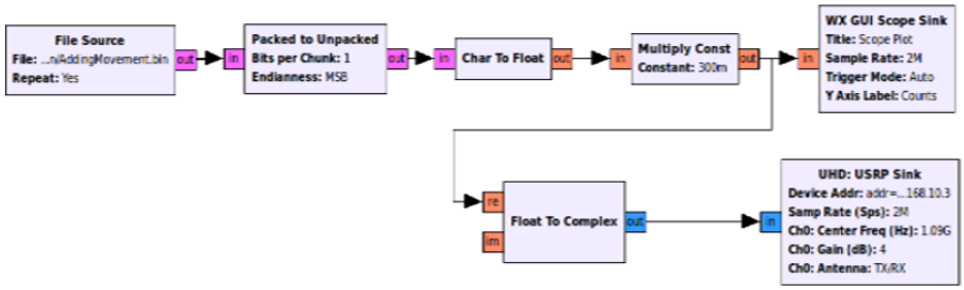

Figure 11: USAF exploitation GNURadio Block Diagram[7]

This did not trigger the ADS-B testbed. In the block diagram above, the zero line of the complex to floating point conversion block is not connected. As the software defined radio used in the referenced paper was an Ettus Radios USRP, rather than a HackRF, this was tied at zero in case the HackRF required an explicit zero. Figure 12 below shows the new block diagram.

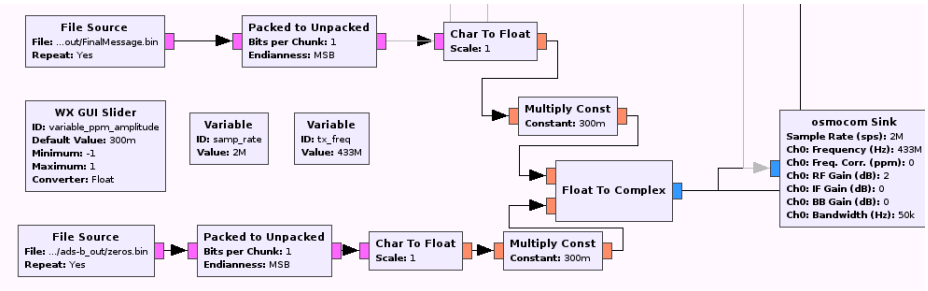

Figure 12: Modified GNURadio flow diagram with clamped zero

As this did not trigger the testbed, the raw output from the RTL-SDR radio was recorded and compared to the transmitted waveform. Figure 13 below shows the transmitted waveform on the left, compared to the received waveform on the right.

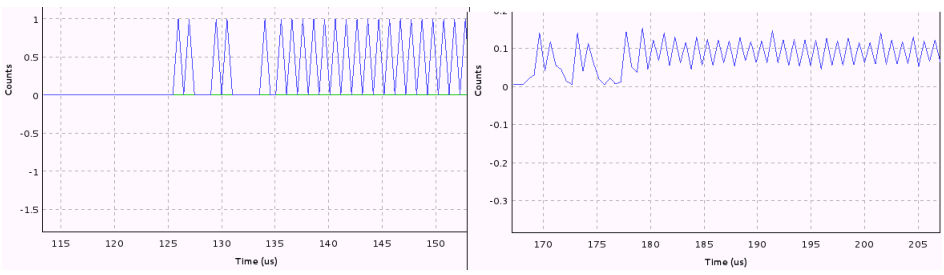

Figure 13: Transmitted ADS-B waveform on the left, received on the right

Both of these waveforms are similar and fit the requirement for an ADS-B transmission (0.5μs pulses, total length of 120μs, with an 8μs preamble). In an attempt to determine why the waveform was not recognised, a band of the spectrum around 1090MHz was recorded using the HackRF as complex samples for analysis, using the bash script below.

Listing 2: HackRF Receive script

```
root@Moriarty{~/Masters_Project}:cat rx.sh
hackrf_transfer -r $1 -f 1090020000 -s 8000000 -g 50
```

The offset of 20kHz removes the effect of the HackRF's DC interference, and $1 represents the first argument passed to the script, which will be the recorded filename. This file was then retransmitted at 433MHz using the following script.

Listing 3: HackRF Transmit script

```
root@Moriarty{~/Masters_Project}:cat tx.sh
hackrf_transfer -t $1 -f 433020000 -s 8000000 -x 20
```

With the testbed tuned to the same frequency, no response was observed. The noise level in the 433MHz band is higher than the 1090MHz band, but even with extra noise filters in the testbed, no response was observed.

ADS-B injection using software defined radios, and the HackRF specifically, has been demonstrated previously[7][8][19]. This leads to the conclusion that there is a key element missing from this injection system. Given that even re-playing captured ADS-B traffic failed to succeed, a likely solution is that the sample rates of the transmitting and receiving stations are mis-matched for the operating frequency. Whilst multiple attempts to re-sample the signal were made, this did not alter the outcome. It is possible that this issue may resolve itself during transmission at 1090MHz, rather than at 433MHz.

3.4 Simulated RF Transmission - ZeroMQ

The GR-AIR-MODES receiver includes an option to accept input on a ZeroMQ socket. ZeroMQ is a networking framework designed to send atomic messages over a variety of transport layers. In this case, the transport layer is TCP/IP. The ADS-B receiver is run using the following command-line arguments:

```
modes_rx -s osmocom -r 2000000 -T 20 -a tcp://127.0.0.1:1337 -m 127.0.0.1:5000
```

This initialises the receiver to use the osmocom driver, reading samples from the RTL-SDR radio, at a sample rate of 2M samples per second, and a noise threshold of 20dB. In addition, it will connect to the ZeroMQ socket at address 127.0.0.1, port 1337, and add any messages published via that socket to the message queue. This enables the collation of multiple sensors information. The '-m' argument is the FlightGear multiplayer server to send data to.

Python ZeroMQ bindings were used to create this publishing socket. An example snippet is shown in Listing 4.

Listing 4: ZeroMQ Message injection snippet

```
>> import zmq
>> context = zmq.Context() # Create a ZMQ Context
>> socket = context.socket(zmq.PUB) # Set type Publisher
>> socket.bind("tcp://*:%s" % 1337) # Bind to localhost at port 1337
>> # Generate Even Position message
>> message= libadsb.ADSBMessage(ident,0x05,[10000,51.1300,2.733, 0])
>>  # Format message for transmission
>> messagedata = '%s 000000 0.0003365123994 0000000000"%message.hex
>> topic = 'dl_data" # This is the topic that GR-AIR-MODES subscribes to
>> socket.send_multipart((topic, messagedata))  # Send the data
```

3.5 Exploitation

3.5.1 Traffic Injection System

A traffic injection console program was written in Python to simplify the execution of the attacks. This program requires no knowledge of the underlying protocols or message formats to run, enabling a low-skilled user to easily generate traffic. Messages are generated according to supplied parameters for the 'ghost' aircraft, and then transmitted using the ZeroMQ interface. By using a threaded sending architecture, the message parameters can be changed on the fly from the console. Figure 14 and Figure 15 below show the message parameters and the help screen respectively.

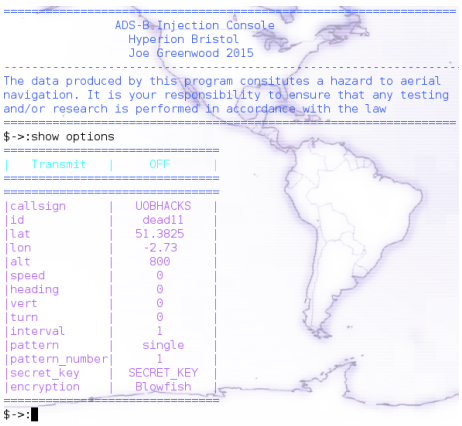

Figure 14: Traffic injection console program

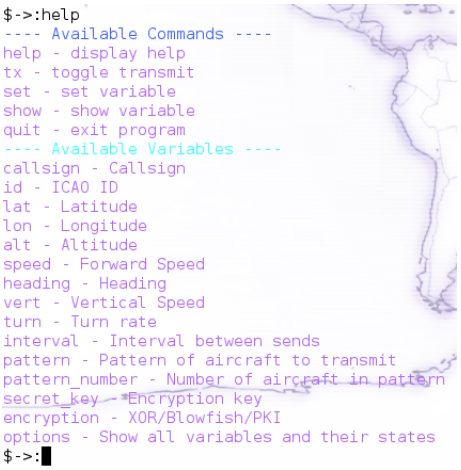

Figure 15: List of injection program commands

The flow diagram for the injection scheme is shown in Figure 16, and the full injector is available at Appendix B.

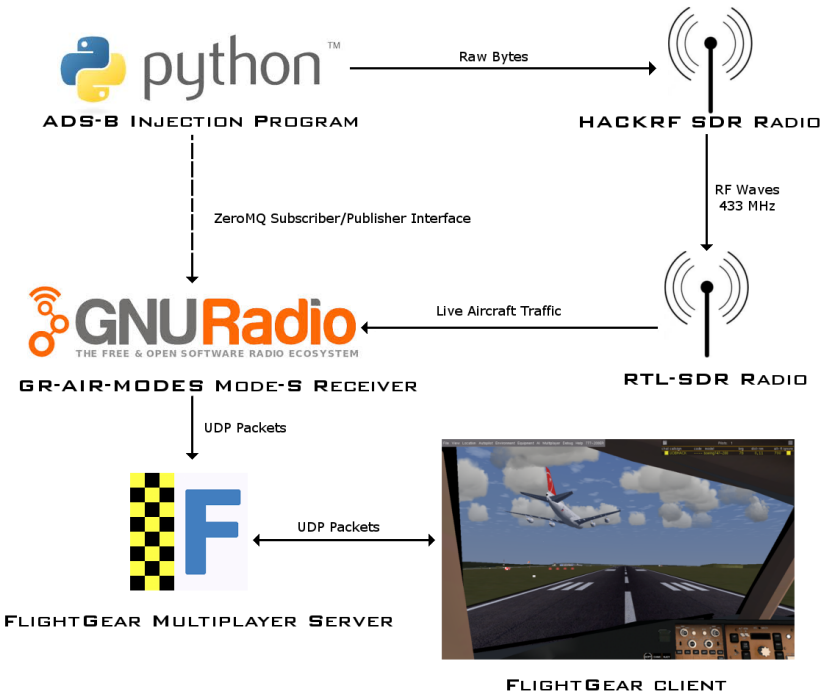

Figure 16: Information flow diagram for injector system - dashed line for software bridge

3.6 Technical Defenses

The aims of the defenses detailed in the following section are threefold; to prevent a malicious adversary from injecting false traffic into the network (authentication), to prevent illicit modification of data (integrity) and to prevent an adversary from reading information from the network (confidentiality), with the priority being authentication. The ideal 'silver bullet' defence would require no changes to existing transmitter and receiver systems, save the addition of a 'security black box' that would handle all security functions, whilst also minimising the time delay added to signals, and without reducing the number of aircraft able to operate in the band. These are conflicting requirements, and therefore none of the prototypes demonstrated here fit this description. Each solution has its own advantages and disadvantages, which will be discussed later.

3.6.1 Symmetric Encryption - AES CTR

By hooking the message parsing function in GR-AIR-MODES, an Advanced Encryption Standard (AES) decryption function was able to be applied to incoming Type 17 messages (ADS-B). Due to the small size of the message (80 bits to be encrypted) AES was used in Counter (CTR) mode as a stream cipher rather than as a block cipher. This was paired with an encryption function in the ADS-B library. The library was extended to allow an encryption scheme and secret key to be specified. It was assumed that the counter would be generated deterministically by both sender and receiver, and that the shared key would be distributed to all privileged parties. The decryption code is shown below in Listing 5.

Listing 5: AES CTR Decryption code

```
def aes_decrypt(data, key):
  #Decrypt a message using AES CTR with fixed counter
  data = hex(data)[2:-1]
  payload = data[2:-6]
  h = SHA256.new()
  h.update(key)
  aeskey = h.digest()
  cipher = AES.AESCipher(aeskey, AES.MODE_CTR, counter=lambda: '\x01'*16)
  # This counter would be deterministic, but is fixed for now.
  ciphertext = str(bytearray.fromhex(payload))
  plaintext_payload = cipher.decrypt(ciphertext)
  # Reconstruct hex packet
  plain_hex = data[:2] + plaintext_payload.encode("hex") + data[-6:]
  result = int(plain_hex,16)
  return result
```

3.6.2 Symmetric Authentication - AEAD

Authenticated Encryption with Associated Data (AEAD) is an encryption mode that provides intrinsic encryption and authentication and integrity checks. These schemes would appear to satisfy all of the objectives of our 'silver bullet', but the extremely short length of the ADS-B message means that there is no way to fit encrypted content and the associated authentication and integrity data in one packet. This necessitates the splitting of encrypted ciphertext into a number of separate messages.

Type 24 messages are defined as 'Extended Length Messages' in the Mode-S protocol. These allow messages to be chained together to provide longer messages, up to 1280 bits in length. A custom protocol was designed using the 80 bits available in each Type 24 message. The first four bits are a sequence number, giving the order of the corresponding packets. Following this is the ICAO address of the sending aircraft, to allow messages to be grouped by sender. This is followed by the CRC of the message that was previously encrypted. This ties the tag messages to one particular payload, allowing it to be subsequently decrypted and verified. The structure is shown below in Figure 17.

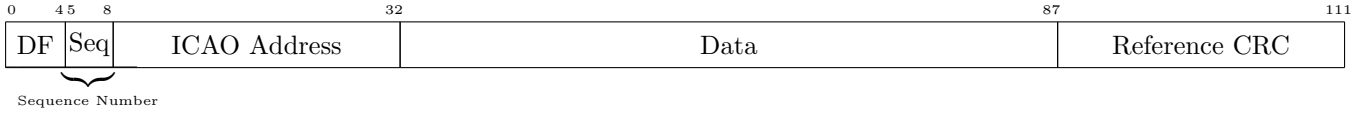

Figure 17: Type 24 AEAD Protocol

In order to decode the messages, a stateful decoder was added to the GR-AIR-MODES parsing system. This decoder hooks ADS-B messages and Type 24 messages, combines and decrypts them, before publishing them to the internal message queue to be decoded as ADS-B messages (providing they pass validation). This decoder is shown below in Figure 18.

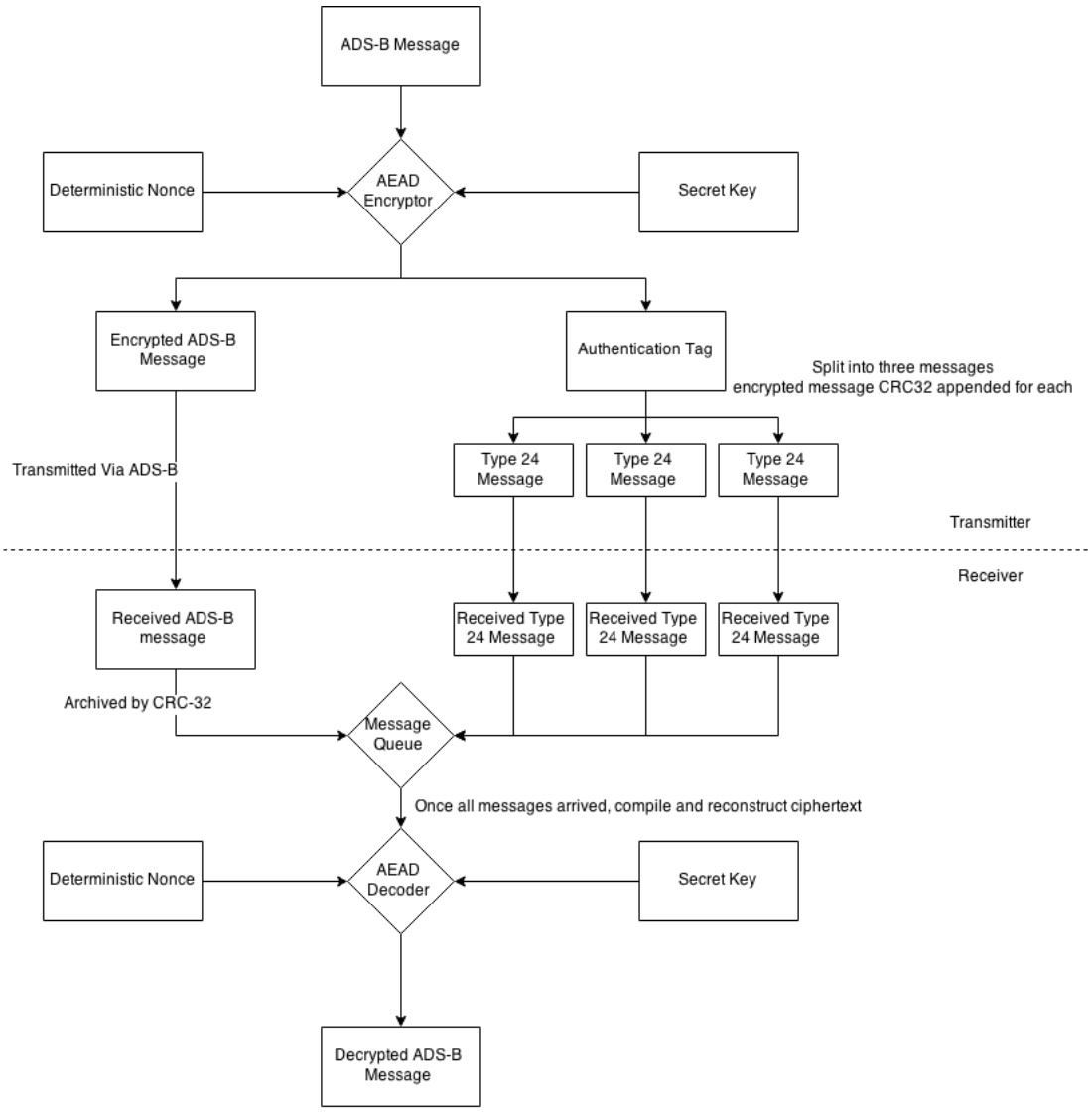

Figure 18: Stateful AEAD decoder system

For this system, the pynacl package was used to provide NaCl, the 'Networking and Cryptography Library' [20] for encryption and decryption. A Salsa20 stream cipher was used with a Poly1305 Message Authentication Code.

3.6.3 Hash Message Authentication Code - HMAC MD5

A keyed Hash Message Authentication Code (HMAC) is a method of verifying both the integrity of the message, and authenticating the sender. By using a shared secret key, both parties can independently verify that the message has not been altered in transit, and that the sender has the same shared secret key.

In order to implement this scheme, a similar stateful decoder was used to the AEAD scheme. The HMAC generated by HMAC-MD5 is the same length as that of the AEAD scheme, so the same Type 24 communications channel was used. Rather than decrypting the message at the receiving end, the HMAC function was applied to the message

payload data, and the resulting digest compared to the received digest (recompiled from three Type 24 messages). The HMAC was generated at the transmitting end in LibADSB by the function below in Listing 6, with a similar function at the receiving end.

Listing 6: HMAC Creation code

```
def hmac_create(self, key):
        # Format key in correct form
        h = SHA256.new()
        h.update(key)
        key = h.digest()
        # Create hmac
        h = hmac.new(key) # Defaults to md5 as hashing algorithm
        # Format data into required form
        data = self.hex[2:-6]
        data = int(data,16)
        # Create HMAC
        h.update(str(data))
        return h.digest() # Returns 128-bit digest
```

3.6.4 Asymmetric Authentication - ECDSA

The essence of public-key cryptography is that an entity has two keys; public and private. The public key is distributed as widely as possible. If the entity wishes to sign some data, they apply their private key to the data. The public key can then be used to verify that it was indeed the entities corresponding private key that signed the data. Similarly, data can be signed or encrypted with the public key, and only the corresponding private key can verify or decrypt that data[21]. This differs from symmetric encryption, where both parties share the same secret key for encryption and decryption.

Elliptic Curve Digital Signing Algorithms (ECDSA) use elliptic curve cryptography to implement an asymmetric public key signing scheme[22]. Elliptic curve cryptography is based on the algebraic structure of elliptic curves over finite fields (such as Galois fields). This makes the keys considerably smaller than for an equivalent non-elliptic asymmetric scheme (such as RSA or DSA).

This type of scheme is well suited to an authorisation and integrity problem such as in a broadcast network. In order to implement it for ADS-B, the payloads for a complete set of state messages (two BDS05 Position, one BDS08 Ident and one BDS09 Velocity) are appended together and signed using the aircraft private key. This signature (384 bits) is then split into seven Type-24 Extended Length Messages in a similar manner to the AEAD scheme. These are then transmitted alongside the original messages. The receiver receives all of the messages and adds the ADS-B messages to a queue, pending verification. The signature is then assembled from the Type 24 messages and verified using the published aircraft public key. If the signature is valid, then the messages are accepted, otherwise they are dropped. Figure 19 below shows the ECDSA system.

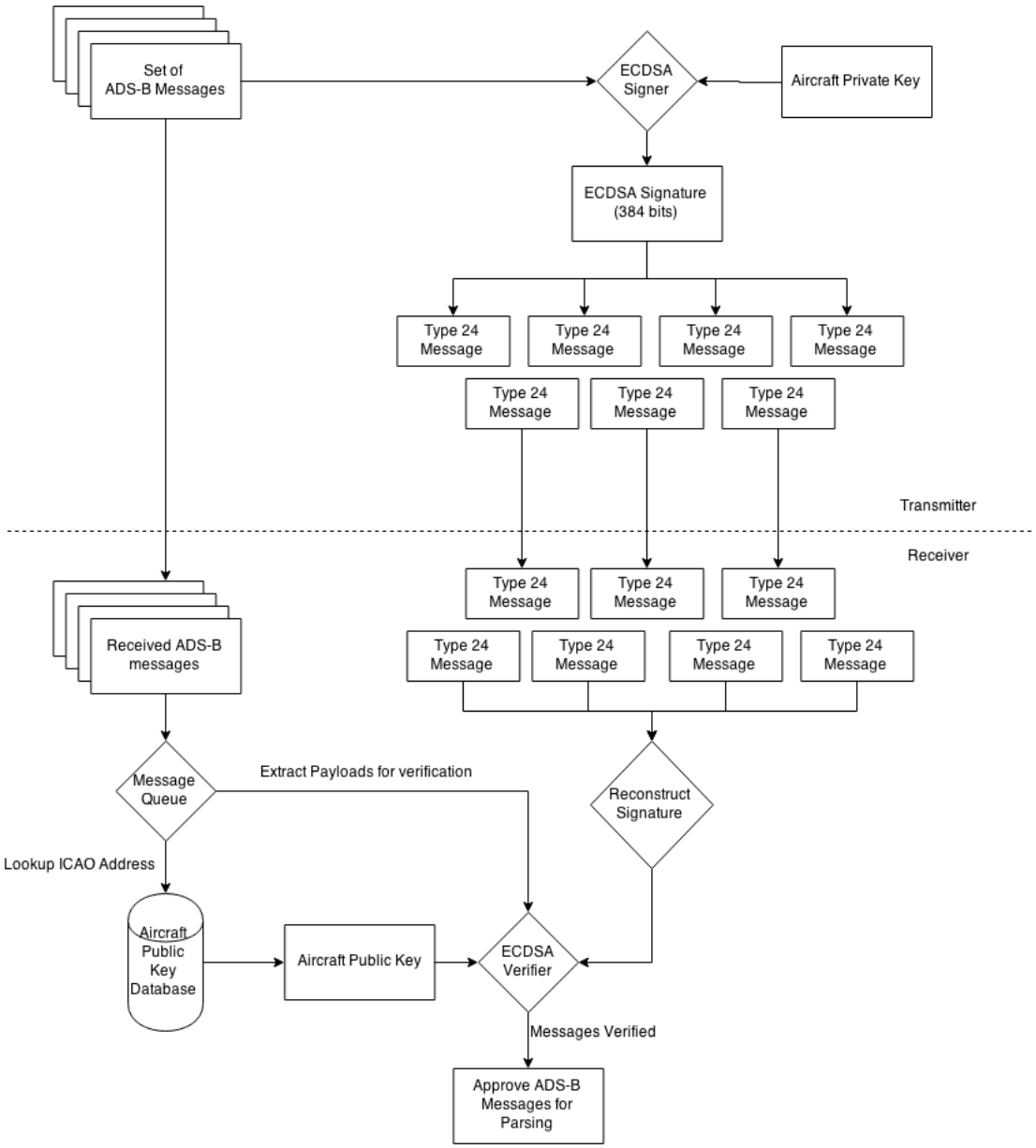

Figure 19: Stateful ECDSA message verification scheme

4 Results

4.1 Traffic Injection

4.1.1 Targeted Attack

Using the injection program, arbitrary traffic was able to be injected into the ADS-B testbed. Figure 20 shows a single aircraft with callsign 'UOBHACK' injected, with speed, altitude and position chosen to present a false collision risk. This aircraft was then responsible for a TCAS RA commanding the pilot to climb, as shown in Figure 21.

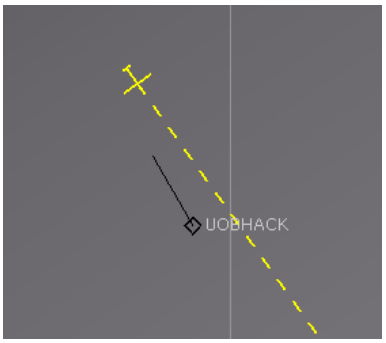

Figure 20: Aircraft 'UOBHACK' injected into the system

Figure 21: TCAS RA caused by the injection of aircraft 'UOBHACK'

4.1.2 Mass Attack

In addition to injecting one singular 'ghost' aircraft, it is possible to inject an arbitrary number of aircraft in arbitrary locations. The injection program was modified to inject a swarm of randomly placed aircraft around a central latitude and longitude. This resulted in a large number of RAs being issued simultaneously. At times these were conflicting and rapidly changed direction between climb and descending. Were a pilot to encounter multiple conflicting RAs, the workload in the cockpit would increase massively, and it would leave the aircraft unaware of any real collision warnings. Figure 22 below shows the TCAS display during this injection.

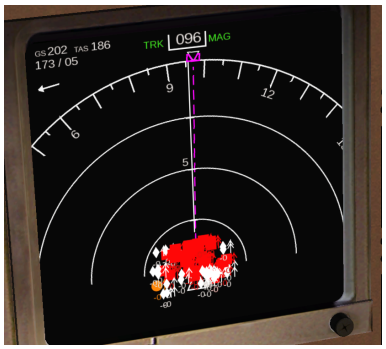

Figure 22: Collection of TCAS RAs due to an injected swarm of aircraft

4.2 Defenses

The source code for all of the defenses are listed in the appendices. Appendix B contains the transmission system, and Appendix C contains the modifications to GR-AIR-MODES.

4.2.1 AES Encryption

When traffic was injected that did not have the correct encryption or obfuscation key, it was decoded as garbage data by GR-AIR-MODES, and therefore rejected. There was a negligible increase in message decoding time, as shown below in Figure 23.

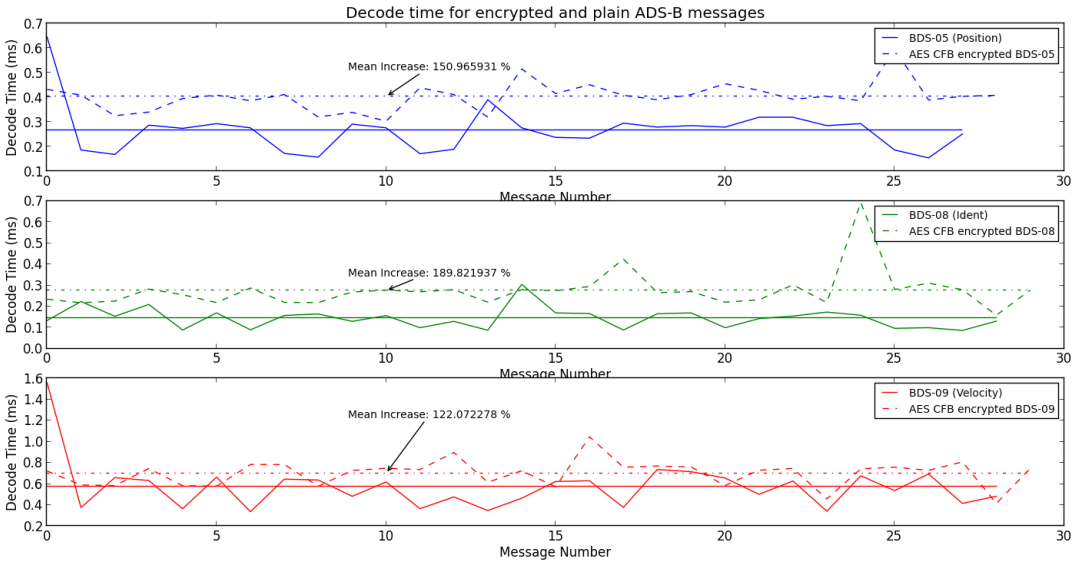

Figure 23: Increase in message decoding time due to AES Encryption

4.2.2 ECDSA Authentication , AEAD Encryption and HMAC-MD5

The AEAD, ECDSA and HMAC schemes prevent the injection of false traffic. Any traffic that fails the verification stages is logged and dropped. The ECDSA and HMAC schemes do not encrypt the data itself, so anyone can still eavesdrop on the ADS-B traffic.

All of these schemes have a time penalty and a bandwidth penalty over the plain ADS-B system. In terms of bandwidth, the ECDSA scheme requires seven extra messages for every set of five ADS-B messages, resulting in a 58.3% decrease in the maximum number of users. The AEAD and HMAC schemes require three messages for every single ADS-B message, reducing the maximum number of users by 75%. It is difficult to measure the increase in time, as the first ADS-B message will be delayed until all of the subsequent messages have been received, compiled and verified. It is certain however that this will be considerably slower than purely encrypting the messages.

5 Discussion

5.1 Exploitation Impact

This project has demonstrated the impact of false traffic injection on the ADS-B and TCAS systems. TCAS is a critical system aboard aircraft, with a failure rate of around 0.000097 per flight sector[23]. Assuming the RF transmission problems are resolved it is trivial for an attacker to use a cheap laptop and HackRF system to render this system inoperable, and thereby cause failure of a critical section of aircraft avionics. This in turn massively increases the risk of collision for an aircraft, and increases the crew workload. This highlights the enormous

asymmetry of digital security, whereby a critical system costing tens of thousands of pounds is rendered useless by £400 worth of investment. This level of investment is easily within the reach of the average hacker, let alone a well-funded terrorist organisation. The attack can be executed from anywhere within 300NM of the target aircraft, with little to no attribution possible.

5.2 Defenses

5.2.1 Symmetric Encryption

Unlike a simple obfuscation scheme, it is difficult to recover the secret key when using AES. However, there is still no integrity checking on the messages. Senders could potentially be authenticated through their possession of the secret key. This was the proposed solution for the military ADS-B system using Type 19 messages[24]. This would enable the sender to produce valid decodable messages, which all follow a similar pattern. By applying heuristic analysis to the sequence of decoded messages produced by a sender, it could be determined whether they are valid. This would massively increase the complexity of an already complex safety critical system however, and still leaves the possibility that an attacker could inundate the system with false garbage traffic.

The other main problem with this scheme is the distribution of secret keys. In order for the system to work every aircraft must have a copy of the secret key whilst also keeping it secret from adversaries. In addition, the key should ideally be changed regularly, to reduce the impact of a disclosed key. This becomes a mammoth task considering Airbus alone had 15,372 aircraft orders in total as of 31st March 2015[25].

5.2.2 Symmetric Authentication and HMAC

The AEAD scheme solves a number of problems with pure symmetric encryption. As with the HMAC scheme, it is no longer possible to modify and replay traffic, as these messages would fail validation. It is also trivial to add a time stamp to the system, meaning that previously transmitted messages cannot be replayed at a later date. This system also provides confidentiality of the data, meaning that aircraft could not be tracked using ADS-B. Both of these schemes also provide explicit authentication (over symmetric encryption), as the fact that the message validates proves that the sender has the secret key. The HMAC scheme also has the advantage of being separate from the main ADS-B message flow, so a 'bolt-on' piece of hardware could provide message validation.

However, these schemes also share the problem of key distribution with pure symmetric encryption. In addition for the symmetric schemes used here, the nonce value used must remain unique in order to ensure the security of the system. This becomes a challenge when the system is used in a broadcast network with a large data rate and thousands of aircraft across the world. The use of a Pseudo-Random Number Generator (PRNG) seeded using the ICAO address of the transmitting aircraft or a counter with the ICAO address as the first 3 bytes provides sufficient assurance that the nonces remain unique. With a set of two BDS05, one BDS08 and one BDS09 message as the minimum required information to determine an aircraft state results in a total of nine messages transmitted per second (BDS05: 2/sec, BDS08: 1/sec, BDS09: 2/sec). Assuming every aircraft has a unique ICAO address,

that leaves a total of 21 bytes left for the counter section (The AEAD system has a 24 byte nonce, and the ICAO address is 3 bytes). This results in 3.74×10^{50} possible unique nonces per aircraft, and a time of 1.317×10^{42} years continuous transmission until a nonce is re-used.

As mentioned previously, the major disadvantage of these systems is the requirement to transmit three Type 24 messages for every one ADS-B message. This is due to the limited space available in an ADS-B message, which is already at the limits of the limited space in Mode-S. This small size makes it very difficult to apply state of the art cryptographic primitives. Modern block ciphers for instance generally use a block size of 128 bits, which is longer than the 80 bits in the data portion of an ADS-B message. This precludes the use of AEAD schemes such as AES Galois Counter Mode.

For the HMAC scheme it is possible to truncate the validation hash, but this comes at the expense of security. If the ICAO identifier was removed from the Type 24 messages in the HMAC scheme, and the digest was truncated to 80 bits, then each ADS-B message could be validated by one Type 24 message. This provides the least reduction in bandwidth for all of the authentication schemes demonstrated (50%). However, this reduction to 80 bits of digest reduces the number of possible digests by 2.81×10^{14}, making it much easier for an attacker to generate a message with a matching hash (a so-called hash 'collision'). Given that MD5 (with a size of 128 bits) is currently regarded as a 'broken' hash[26], reducing the number of bits below this provides very little security at all.

5.2.3 Asymmetric Authentication

The use of asymmetric authentication and signing has a number of advantages over the previously mentioned schemes. Signing provides both integrity of the message and authentication of the sender. As with AEAD, a time stamp would be used to prevent message replay attacks. Public key cryptography also makes the issue of key distribution considerably easier. The only secret is the private key, which is specific to an aircraft and can therefore be burned into a specific hardware device. The public key is then distributed as widely as possible to all stations, potentially via a secure out of band channel, such as an airport ground network, or a 2/3G network. This shifts the security onus away from the aircraft and onto the ground networks. Whilst in some cases this may be an advantage, a ground network suffers from all of the problems associated with traditional digital security.As with the HMAC scheme, the other main advantage of this scheme is that it requires no changes to existing ADS-B hardware or software. The only addition is a separate signing unit at the transmission end, and a verification unit at the receiver. These can be implemented as separate hardware units and 'bolted on' to the existing ADS-B architecture.

The major drawback however is the reduction in capacity that this system produces. By requiring seven signing messages for each set of five ADS-B messages, the amount of bandwidth required for each aircraft increases dramatically. This could cause large problems in high-density traffic areas such as Heathrow, which are already operating at near maximum capacity. The other main issue is the time delay associated with verifying messages. For an aircraft cruising at 945km/h, a delay of one second means the aircraft is 262.5m closer to a potential collision. This

scheme also does not solve the issue of data confidentiality.

5.2.4 Distance Bounding

Other potential solutions involve the use of a distance bounding protocol. These allow a verifier to establish an upper bound of the physical distance of a broadcasting station by timing the response to a challenge. In the context of ADS-B, this would enable the aircraft to determine if a transmitting station was in fact where they claimed to be. Currently any attacker within transmission range (300NM) of the aircraft could claim to be in a close collision state. By establishing the range of the transmitter the aircraft can satisfy itself that no risk of collision exists.

Practical examples of these protocols have been implemented using RFID cards over very short distances[27]. The short distances require a very high degree of accuracy to time the response, making processor calculation times and issue. This would be reduced in a system such as ADS-B, where the distances are orders of magnitude higher. However, systems such as this are still open to some exploitation, as the attacker could still inject traffic anywhere within a range circle around the aircraft. They are also subject to vulnerabilities in themselves, as demonstrated by the University of Cambridge[28].

5.2.5 Policy Protections

There are also a number of defenses that are not technical in nature. TCAS RAs currently have a higher priority than Air Traffic Control instructions. This is because the TCAS system on board the aircraft may have a better view of the situation than a ground based surveillance system. However, when arbitrary alerts can be triggered this becomes a serious vulnerability. One solution to this would be to implement a series of code words to be used by Air Traffic Control to indicate that an aircraft is a 'ghost'. Ground stations can use techniques such as multilateration to determine where the transmitter is. Relaying this information to the air crew would reduce the risk associated.

However, training pilots to ignore TCAS warnings in some circumstances may have a negative effect on their adherence to valid TCAS warnings. Previous research into this area suggests that there are still many instances where pilots ignore a RA, or take incorrect avoiding action[5]. The risk of an aircraft collision due to a 'ghost' aircraft needs to be weighed against the risk of a collision due to pilot error.

5.3 Combined Defenses and implementation

A key principle of digital security is of 'defence in depth'. Rather than relying on one defensive layer, it is best practice to use multiple layers in the event that one fails. This approach can be applied to ADS-B by introducing a combination of the above schemes.

As an example, the ECDSA scheme could be used to distribute keys for use with the symmetric encryption scheme. This is a standard cryptography primitive. This would provide both integrity and authorisation for the data

(through the ECDSA signatures), whilst also providing confidentiality for the data. In addition, a distance bounding protocol could be implemented by signing timestamps that are then transmitted. The receiving station could then verify that the time stamp was sent by a valid station, and compare this to the time of reception to work out a range to the station. This combination would provide multiple layers of protection over the current system.

The cost and effort of implementing defenses for ADS-B would be enormous. Due to the limitations of a legacy system, any solution would have to be built using bespoke hardware, software and cryptography, rather than using an off-the-shelf solution. The added difficulty of developing and certifying safety critical software makes implementing solutions that are currently commonplace in the digital security environment very difficult. In addition, the system would have to provide security for many years into the future, anticipating advances in security and cryptography.

6 Conclusion

ADS-B as it currently stands is a flawed system. Providing no security whatsoever it allows malicious adversaries, with a nominal investment and negligible risk, to degrade the safety of air traffic. This project aimed to demonstrate the ease with which an adversary could inject false traffic into the ADS-B system. An RF injection prototype was developed, but this failed to trigger a test bed operating on 433MHz. Future work would couple the Python ADS-B message generation library with a HackRF transmitting on the Mode-S band at 1090MHz.
The secondary aim of this project was to develop and demonstrate defenses against false traffic injection. These defenses are summarised below:

- Symmetric encryption provides confidentiality, but no guarantee of integrity, or intrinsic authentication. It did however have the least impact on bandwidth.
- A keyed HMAC system provides authentication and integrity, but no confidentiality. This method also has a bandwidth reduction of between 75% and 50%, depending on the truncation (and therefore strength) of the hashing algorithm.
- An AEAD system provides authentication, confidentiality and integrity for messages. However, this system requires a 75% reduction in bandwidth.
- An ECDSA system provides authentication and integrity of data, with a minimum interference with the ADS-B protocol. It also solves the key issue with key distribution. However, it still entails a reduction in bandwidth of %, and does not provide confidentiality.

The ideal solution would provide confidentiality, authentication and integrity of data, whilst also requiring a minimum of change to the current system. By taking a layered approach, all of these properties could be implemented using a combination of the schemes prototyped in this project. Future work would look at the feasibility of implementing these in lower level languages, and ultimately in hardware.

References

[1] Federal Aviation Administration, *Automatic Dependent Surveillance – Broadcast (ADS-B) Out performance requirements to support Air Traffic Control (ATC) service*, 2007.

[2] COMMISSION IMPLEMENTING REGULATION (EU) No 1207/2011, *Laying down requirements for the performance and the interoperability of surveill ance for the single European sky*, Official Journal of the European Union, November 2011.

[3] Federal Aviation Administration, *Automatic Dependent Surveillance— Broadcast (ADS–B) Out Performance Requirements To Support Air Traffic Control (ATC) Service; Final Rule*, 14 CFR Part 91, 2010.

[4] Federal Aviation Administration, *Introduction to TCAS II Version 7.1*, 2011.

[5] IFALPA, Air Traffic Services Briefing Leaflet 12ATSBL05, *Dealing with TCAS RA reversals*, November 2011.

[6] Example of a flight tracker: http://www.flightradar24.com/, Accessed 21/04/2015.

[7] Domenic Magazu III, Captain (USAF), *Exploiting the Automatic Dependent Surveillance-Broadcast System Via False Target Injection*, Air Force Insitute of Technology, 2012.

[8] Brad "RenderMan" Haines, *Hackers+Aircraft, No Good Can Come of This*, DEFCON 20, 2012. *https://www.youtube.com/watch?v=CXv1j3GbgLk*, Accessed 21/04/2015.

[9] Andrei Costin, Aurelien Francillon, *Ghost in the Air(Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices*, Blackhat USA, 2012.

[10] Donald McCallie, Major, USAF, *Exploring Potential ADS-B Vulnerabilities In The FAA's NEXTGEN Air Transportation System*, Air Force Institute of Technology, 2011.

[11] RTL-SDR Radio available from Amazon, *http://preview.tinyurl.com/nafotct*, Accessed 21/04/2015.

[12] GNURadio website, *http://gnuradio.org/*, Accessed 21/04/2015.

[13] Jens Elsner, Martin Braun, Stefan Nagel, Kshama Nagaraj and Friedrich Jondral, *Wireless Networks In-the-Loop: Software Radio as the Enabler*, Software Defined Radio Forum Technical Conference, Washington DC, 2009.

[14] GR-AIR-MODES Github Repository, *https://github.com/bistromath/gr-air-modes*, Accessed 21/04/2015.

[15] Kali Linux Website, *https://www.kali.org/*, Accessed 21/04/2015.

[16] US Department of Transportation, Federal Aviation Administration, *Introduction to TCAS II Version 7*, 2000.

[17] Compiling Flightgear from Source, *http://wiki.flightgear.org/Building_FlightGear_-_Linux*, Accessed 21/04/2015.

[18] HackRF Radio, *https://greatscottgadgets.com/hackrf/*, Accessed 21/04/2015.

[19] Legally dubious Chinese demonstration, https://www.youtube.com/watch?v=xpLDqBkUiKc, Accessed 21/04/2015.

[20] Daniel J. Bernstein, Tanja Lange and Peter Schwabe, *The security impact of a new cryptographic library*, Pages 159–176 in Proceedings of LatinCrypt 2012, 2012.

[21] Whitfield Diffie and Martin E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, VOL. IT-22, NO. 6, NOVEMBER 1976.

[22] Don Johnson, Alfred Menezes and Scott Vanstone, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, Certicom Research, Canada, Dept. of Combinatorics & Optimization, University of Waterloo, Canada, 2001.

[23] Robert Hemm and Andrew Busickempt, *Safety Analysis of the Separation Assurance Function in Today's National Airspace System* 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO), 2009.

[24] 2ndLt John R. Jochum, *Encrypted Mode Select ADS-B for Tactical Military Situational Awareness*, Massachusetts Institute of Technology, 2001.

[25] Number of Airbus orders, *http://www.airbus.com/company/market/orders-deliveries/*, Accessed 22/04/2015.

[26] Xiaoyun Wang and Hongbo Yu *How to Break MD5 and Other Hash Functions*, Shandong University, Jinan 250100, China, 2006.

[27] Gerhard P. Hancke and Markus G. Kuhn, *An RFID Distance Bounding Protocol*, Cambridge University, 2005.

[28] Gerhard P. Hancke and Markus G. Kuhn, *Attacks on Time-of-Flight Distance Bounding Channels*, Cambridge University, 2008.

Appendices

A Libadsb - Python ADS-B Message Library

This file has been removed at the request of the University of Bristol.

Please contact jg1558@my.bristol.ac.uk if you would like to make a request for it.

B Python traffic injection console

This file has been removed at the request of the University of Bristol.

Please contact jg1558@my.bristol.ac.uk if you would like to make a request for it.

C GR-AIR-MODES modifications

C.1 Additions to parse.py

This file has been removed at the request of the University of Bristol.

Please contact jg1558@my.bristol.ac.uk if you would like to make a request for it.

C.2 Additions to flightgear.py

This file has been removed at the request of the University of Bristol.

Please contact jg1558@my.bristol.ac.uk if you would like to make a request for it.

C.3 Additions to modes_rx.py

This file has been removed at the request of the University of Bristol.

Please contact jg1558@my.bristol.ac.uk if you would like to make a request for it.