ENHANCING THE INTEGRITY OF AUTOMATIC DEPENDENT SURVEILLANCE-BROADCAST SYSTEMS USING FORMAT-PRESERVING ENCRYPTION: AN EMBEDDED SYSTEM SOLUTION

AMÉLIORATION DE L'INTÉGRITÉ DES SYSTÈMES DE SURVEILLANCE DÉPENDANTE AUTOMATIQUE PAR DIFFUSION AU MOYEN D'UN CRYPTAGE PRÉSERVANT LE FORMAT : UNE SOLUTION DE SYSTÈME EMBARQUÉ

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Hasan Ali, BEng
Captain

In Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science in Electrical and Computer Engineering

March 2025

Abstract

The increasing adoption of Automatic Dependent Surveillance-Broadcast (ADS-B) has contributed to more efficient air traffic management by allowing aircraft to autonomously broadcast their flight parameters to Air Traffic Control (ATC) [1]. Despite these benefits, ADS-B remains vulnerable to attacks that threaten data integrity, such as message injection, modification, and deletion [2]. While various mitigation techniques exist, many either require significant modification to existing infrastructure or major changes to the current ADS-B protocol, failing to protect the integrity of ADS-B in a practical way.

This thesis introduces a novel solution for enhancing ADS-B integrity using Format-Preserving Encryption (FPE) implemented on a low-cost embedded system. By installing an in-line device between the aircraft's ADS-B equipment and the ADS-B antenna, ADS-B messages are encrypted before being broadcast using an FPE algorithm, which preserves the format and length of ADS-B messages. Consequently, once the encrypted ADS-B messages are decrypted, they remain compatible with existing decoding tools used by ATC. Three FPE algorithms were implemented in this thesis: FF3, FFX, and AES-CTR. The embedded computer and the three FPE algorithms were also evaluated for encryption and decryption times, standard deviation of encryption times, CPU and memory usage, and thermal performance.

All three FPE algorithms successfully preserved the ADS-B message structure which resulted in correctly decoded messages once decrypted by the receiver. Simulation of message injection and modification attacks demonstrated that non-encrypted or tampered messages failed to decrypt and were therefore rejected. The findings confirm that FPE can be practically applied to protect the integrity of ADS-B communications with minimal disruption to the current infrastructure.

Contents

List of Tables

List of Figures

List of Acronyms

ACARS	Aircraft Communications Addressing and Reporting System
ADC	Analogue-to-Digital
ADS-B	Automatic Dependent Surveillance-Broadcast
AES	Advanced Encryption Standard
AES-CTR	Advanced Encryption Standard in Counter Mode
AOA	Angle of Arrival
AoIP	ACARS over IP
ATC	Air Traffic Control
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DAC	Digital-to-Analogue
EPA	Extended Projection Algorithms
FAA	Federal Aviation Administration
FMS	Flight Management System
FPE	Format Preserving Encryption
GNSS	Global Navigation Satellite System
ICAO	International Civil Aviation Organization
IP	Internet Protocol
IV	Initialization Vector
KMD	Key Management and Distribution
LPE	Length-Preserving Encryption
MAC	Message Authentication Code
ME	Message Extended Squitter
NIST	National Institute of Standards and Technology
NPE	Number-Preserving Encryption

PI	Parity/Interrogator
RF	Radio Frequency
RP	Raspberry Pi
SATCOM	Satellite Communication
SDR	Software-Defined Radio
SIBE	Staged Identity-Based Encryption
SIR	Signal-to-Interference-Ratio
SVD	Singular Value Decompression
TCP	Transmission Control Protocol
TDOA	Time Difference of Arrival
USRP	Universal Software Radio Peripheral
VHF	Very High Frequency
VM	Virtual Machine
VPN	Virtual Private Network

1 Introduction

The integration of Automatic Dependent Surveillance-Broadcast (ADS-B) technology for aviation tracking has transformed Air Traffic Control (ATC) and management by allowing aircraft to automatically transmit their location and flight data [1]. This technology has been crucial in the transition from ground-based radar systems to a satellite-based system, enhancing real-time visibility, airspace efficiency, and safety [3]. The technology's importance is further underscored by mandates from aviation authorities, such as Transport Canada and the Federal Aviation Administration (FAA), requiring aircraft to be equipped with ADS-B Out in certain airspace classes [4, 5]. The introduced mandates will undoubtedly result in an ever-increasing reliance on ADS-B, making it a critical part of modern aviation. The global push for ADS-B reflects its significant role in achieving next-generation air traffic management objectives, including real-time aircraft tracking, reduced fuel consumption, and lower carbon emissions [3].

Despite its efforts to improve the safety and efficiency of flight, ADS-B is not immune to cyberattacks. Its unencrypted signals are vulnerable to a variety of security threats, including eavesdropping, jamming, and message manipulation attacks, which could be exploited to compromise ATC operations [2]. Given the critical nature of ADS-B data for ATC and the potential consequences of compromised ADS-B integrity, developing robust security measures is essential to prevent potential exploitation that could endanger safety, especially as the reliance on ADS-B grows due to mandated adoption.

This thesis addresses the integrity issue in ADS-B communications by proposing a practical solution: the use of Format-Preserving Encryption (FPE) algorithms implemented on an embedded system. By encrypting critical portions of ADS-B messages without altering the message format, the solution preserves compatibility with existing ADS-B infrastructure while enhancing data integrity. The solution presented here is evaluated through a series of performance tests, including measures of encryption and decryption times, standard deviation of encryption times, Central Processing Unit (CPU) and memory usage, and thermal behaviour monitoring. This approach aims to strike a balance between enhanced integrity whilst requiring minimal modifications to existing systems.

1.1 Problem Overview

Although ADS-B greatly improves the efficiency and safety of aviation, it lacks integrity protection mechanisms. As outlined in [2], the ADS-B protocol relies on the transmission of unencrypted messages over 1090 MHz. This makes eavesdropping trivial by allowing any receiver within range to capture and decode these messages using one of the many open-source ADS-B decoder tools available online and in the market. This nature of openness is likely why ADS-B was not designed with encryption in mind. Consequently, apart from eavesdropping, a malicious actor could modify existing ADS-B messages while in transit by changing the aircraft speed, altitude, or position shown in the message resulting in inaccurate aircraft data being received by ATC [2]. They could also inject a fake ADS-B message resulting in a non-existing aircraft appearing on ATC screens. Both these scenarios pose significant safety risks in air traffic management.

The vulnerabilities of ADS-B are not limited to risks in the safety of air travel, but they could also result in disruptions to air operations caused by the injected or modified messages which can lead to delays, diversion, and increased workload for ATC. Additionally, these vulnerabilities could have an economic impact due to delays and inefficiencies resulting in financial losses for airlines and regulatory bodies. Given the critical role of ADS-B in air traffic management, addressing these vulnerabilities is essential. Enhancing the integrity of ADS-B without compromising system compatibility is a key challenge that this thesis seeks to address.

1.2 Thesis Aim

The aim of this thesis is to propose a solution that enhances the integrity of ADS-B messages using FPE while respecting the structure and transmission constraints of the ADS-B protocol, and ensuring it can be implemented on an embedded computer. To achieve this, the solution utilizes an embedded computer that can be installed on an aircraft equipped with ADS-B Out. The embedded computer captures the ADS-B messages before they are broadcast by the antenna and applies encryption using an FPE algorithm. It then transmits the encrypted messages to ATC which will decrypt them. Using an FPE algorithm will ensure that the format of the ciphertext remains the same as that of the plaintext. This guarantees that the encrypted ADS-B messages remain compatible with the current ADS-B infrastructure.

Three FPE algorithms are implemented and evaluated in this study: FF3, FFX, and AES-CTR. Each algorithm is assessed based on its compatibility with the ADS-B format, as well as its performance impact on the embedded system. Performance metrics were measured and include encryption and decryption times, encryption time standard deviation, CPU and memory usage, and the thermal behaviour of the embedded computer. These metrics provide a comprehensive understanding of the trade-offs associated with each FPE algorithm as well as an assessment of the selected embedded system as a suitable encryption system.

In addition to performance evaluation, the thesis also provides an analysis of the security characteristics of the selected FPE algorithms. Several journal and conference proceeding articles outlined the advantages and vulnerabilities of the three FPE algorithms used in this thesis. These papers will be referenced to highlight the differences between the FPE algorithms from a security perspective. The thesis also discusses the key management and distribution (KMD) challenge and suggests potential mitigating solutions. One proposed solution is the use of a hybrid encryption model which uses asymmetric encryption for key exchange and symmetric encryption using an FPE algorithm for message encryption as potential future work. Additionally, the thesis explores the feasibility of leveraging existing aviation communication systems for secure key distribution.

1.3 Organization of Thesis

The remainder of this thesis is organized as follows: Chapter 2 provides a comprehensive background on the ADS-B protocol, the ADS-B mandate, and a literature review of some of the existing ADS-B security vulnerabilities and mitigation techniques. It also introduces FPE and discusses its relevance to enhancing ADS-B integrity. Chapter 3 details the methodology used in this research, including the system design, the implementation of each FPE algorithm and the performance evaluation criteria. Chapter 4 presents the results of the performance evaluation and includes a conceptual analysis of the security characteristics of the FPE algorithms. This chapter also discusses KMD challenges and provides insights into the feasibility of the proposed solution. Finally, Chapter 5 offers recommendations for future research, summarizes the findings, and highlights the contributions of the proposed solution.

2 Background

Modern aviation has seen a transformation thanks to the ADS-B system, which provides real-time tracking of aircraft, enhancing safety and operational efficiency [3]. This chapter examines the technical structure of the ADS-B protocol, identifies some of its vulnerabilities and explores some mitigation techniques proposed by researchers. A detailed discussion of FPE is also provided, emphasizing its applicability in safeguarding the integrity of ADS-B.

2.1 ADS-B System Overview

ADS-B is an aircraft surveillance technology that utilizes Global Navigation Satellite Systems (GNSS) along with on-board avionics and ground or space-based systems for the accurate and real-time relay of flight data between aircraft and ATC [6]. Flight data includes aircraft identification, position, altitude, and velocity. ADS-B is divided into two key components: ADS-B Out, which is installed onboard the aircraft and autonomously transmits its flight data, and ADS-B In, which receives this data using ground-based receivers, space-based receivers, or other aircraft [6]. Unlike a transponder, which utilizes a secondary surveillance radar that requires an interrogation pulse to transmit aircraft data, ADS-B Out transmits aircraft data autonomously without an interrogation request or intervention from the flight crew, hence the term 'Automatic' in ADS-B [1]. Similarly, ADS-B depends on the aircraft's sensors and systems to transmit its velocity, altitude, and position [1]. Figure 1 shows a functional overview of the ADS-B Out system when fitted on an aircraft [6]. It demonstrates that the 'ADS-B Equipment' on the aircraft uses the data captured from other aircraft systems (GNSS, air data, etc.) and then broadcasts this information via the antenna to an ADS-B In receiver, whether ground or space-based. This data is used by ATC for airspace control. Although not depicted in Figure 1, the data can also be received by other aircraft equipped with ADS-B In.

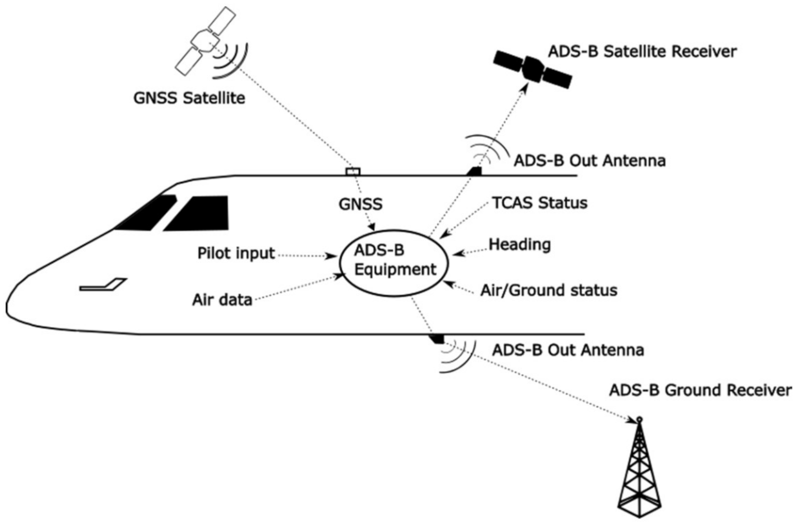


Figure 1: Overview of ADS-B Out system [6]

ADS-B transmits messages over 1090 MHz and its range depends on several factors such as the curvature of the earth, the ADS-B out transmit power, and the antenna characteristics. For a typical ADS-B Out system with a ground-based receiver, the average range is around 400 km, assuming the aircraft and receiver are within line of sight [7]. When the receiver is space-based, the range and coverage area are much greater, which is one of the benefits of switching to satellite-based ADS-B receivers. An ADS-B message consists of the frame format shown in Figure 2 [8]. An ADS-B Out frame is comprised of two primary sections: an 8 µs preamble section, used for synchronization, and a 112 µs payload section, for a total of 120 µs per message [8]. Since data in ADS-B Out is transmitted at a rate of 1 Mbps, this equals 120 bits per message. J. Sun recorded the different segments of the payload section in [7] as shown in Table 1.

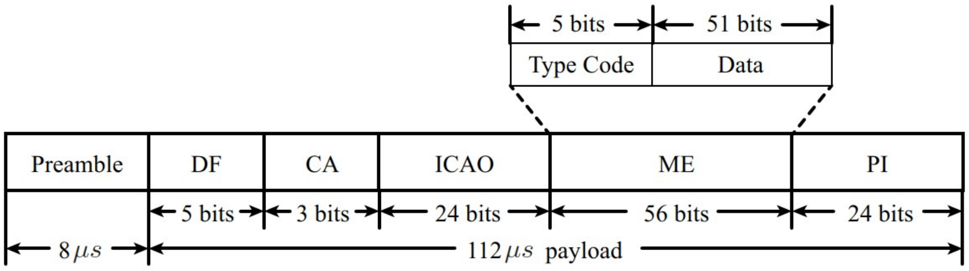
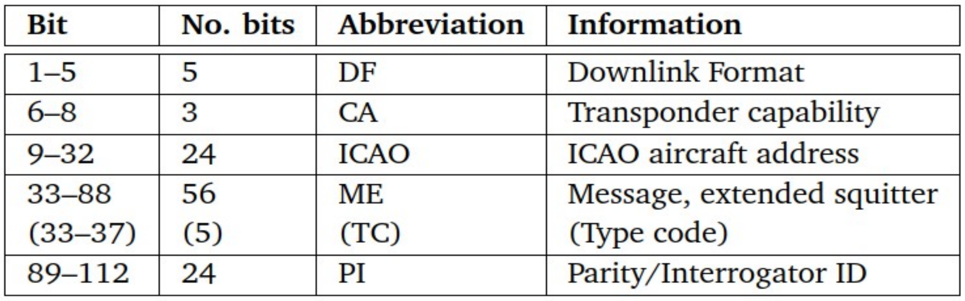


Figure 2: ADS-B Out message frame format [8]

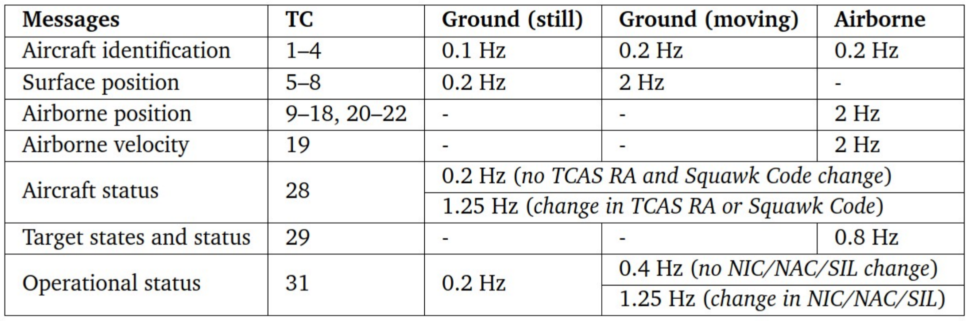
Table 1: ADS-B message payload structure [7]



Bit	No. bits	Abbreviation	Information
1–5	5	DF	Downlink Format
6–8	3	CA	Transponder capability
9–32	24	ICAO	ICAO aircraft address
33–88 (33–37)	56 (5)	ME (TC)	Message, extended squitter (Type code)
89–112	24	PI	Parity/Interrogator ID

Arguably, the most critical segment of the ADS-B frame is the Message Extended Squitter (ME) block. The ME block stores the flight-specific data like aircraft position, velocity, or altitude [7]. Due to its limited size of 56 bits (split into 5 bits of Type Code and 51 bits of Data), only one message type can be transmitted at a time. For example, the aircraft velocity will be sent in a single ADS-B message, and the aircraft identification will be sent in another message. The 5 bits of Type Code will determine which message type is being sent. J. Sun recorded the different ADS-B message types and their codes in Table 2 [7].

Table 2: ADS-B message types and transmission rates [7]



Messages	TC	Ground (still)	Ground (moving)	Airborne
Aircraft identification	1–4	0.1 Hz	0.2 Hz	0.2 Hz
Surface position	5–8	0.2 Hz	2 Hz	-
Airborne position	9–18, 20–22	-	-	2 Hz
Airborne velocity	19	-	-	2 Hz
Aircraft status	28	0.2 Hz (*no TCAS RA and Squawk Code change*)		
		1.25 Hz (*change in TCAS RA or Squawk Code*)		
Target states and status	29	-	-	0.8 Hz
Operational status	31	0.2 Hz	0.4 Hz (*no NIC/NAC/SIL change*)	
			1.25 Hz (*change in NIC/NAC/SIL*)	

Table 2 shows the type codes for the various ADS-B message types. Each message type requires a single frame to be sent by ADS-B Out, with the exception of airborne position and surface position messages [7]. These position messages require two separate frames – referred to as even and odd messages – due to the size constraints of the ME block. For these messages, a flag bit within the ME block indicates whether the message is even or odd. All this is processed and transmitted autonomously by ADS-B Out and the pairs of messages get decoded by the receiver. Table 2 also shows the transmission rates for each message type. For example, ADS-B Out automatically transmits airborne position messages every 0.5 seconds (2 Hz), while aircraft identification messages are transmitted every 5 seconds (0.2 Hz), if the aircraft is airborne, and every 10 seconds (0.1 Hz) if the aircraft is stationary on the ground. Each ADS-B message, regardless of its Type Code, always contains the aircraft's unique 24-bit ICAO address, as shown in Table 1. This allows the receiver to match all ADS-B messages to the correct aircraft even though aircraft identification messages are transmitted less frequently.

Another critical segment of the ADS-B message is the Parity/Interrogator (PI) block. The PI block is 24 bits long and is responsible for error detection, primarily through a Cyclic Redundancy Check (CRC), though other types of error detection exist depending on the version of ADS-B used [7]. ADS-B Out applies CRC by performing calculations on the bits of the ADS-B message excluding the PI segment, and the resulting remainder is placed in the PI block before transmission. The receiver will then perform the CRC calculation on the entire ADS-B message, including the PI segment. Since the PI block already contains the remainder from the initial CRC computation, a correctly received message will always yield a remainder of zero when recalculated at the receiver. This confirms that the ADS-B message was received without error. Otherwise, if any bit gets corrupted during transmission, the recalculated CRC will result in a nonzero remainder, indicating an error, and the message is, therefore, discarded [7].

Many decoders exist to interpret ADS-B messages. One such decoder is Dump1090, a widely recognized open-source ADS-B decoder which is available on GitHub [7]. This tool converts ADS-B messages formatted in hexadecimal into a human-readable format. For example, a raw ME segment of 202CC371C32CE0 (shown in hex) is decoded to KLM1023 (aircraft call sign) using Dump1090 [7].

2.2 ADS-B Mandate

In February 2022, the Canadian air navigation service provider, NAV Canada, mandated the requirement for aircraft to be fitted with an ADS-B Out capability in a phased approach [4]. This mandate currently applies to aircraft flying in Class A Airspace (aircraft flying over 18,000 ft) and Class B Airspace (aircraft flying over 12,000 ft), and will apply in Classes C, D, and E no sooner than 2028. ADS-B has many benefits when compared to primary/secondary radar systems:

- **Safety:** Aircraft equipped with ADS-B equipment provide ATC with an increased area of surveillance coverage, thus enhancing ATC's situational awareness [4]. This is also applicable to pilots in other aircraft fitted with an ADS-B In system. Additionally, ADS-B offers better tracking of aircraft in distress, which improves search and rescue response times [4].
- **Efficiency:** Due to ADS-B's improved accuracy compared to primary and secondary radar, ATC can maintain aircraft separation more efficiently [1]. Additionally, the increased positional accuracy can offer more efficient

flight routing which often uses direct and shorter routes, resulting in a reduction of greenhouse carbon emissions [4].

- **Cost:** ADS-B can greatly reduce the cost of maintenance of ATC radar equipment and offers a longer service life [9]. Additionally, fuel savings are also a benefit of ADS-B due to the ability of providing shorter and more direct routes [1].

Transport Canada is not the only regulatory body to impose such a mandate. In fact, the FAA in the United States mandated ADS-B Out for most controlled airspace classes starting in January 2020 [10]. Similarly, the European Union Aviation Safety Agency released a similar mandate requiring ADS-B Out compliance starting December 2020 [11]. These efforts align with the International Civil Aviation Organization's (ICAO) vision of seamless air operations around the globe [12]

2.3 Security Vulnerabilities in ADS-B

Numerous studies have focused on various attack vectors targeting ADS-B systems. Each attack tackles one or more of the three main computer security goals: Confidentiality, Integrity, and Availability [13]. This section will provide a brief overview of key attacks and vulnerabilities that exist in ADS-B. A detailed survey of existing ADS-B vulnerabilities and mitigation techniques was conducted in [2].

2.3.1 Eavesdropping

Eavesdropping tackles the confidentiality of ADS-B by exploiting the unencrypted nature of ADS-B messages. It involves the interception of ADS-B messages, allowing an adversary to gather sensitive flight information such as aircraft position, speed, and identity [9]. An eavesdropping attack can be achieved by using a Software Defined Radio (SDR) and the appropriate antenna for 1090 MHz. Then, an ADS-B decoder, such as dump1090, can be used to decode received ADS-B messages.

2.3.2 Jamming

Jamming, in the context of ADS-B systems, refers to the deliberate transmission of radio frequency (RF) signals that interfere with the normal operation of ADS-B, effectively preventing the reception of legitimate ADS-B messages [9]. Unlike eavesdropping, which mainly targets the confidentiality pillar of ADS-B, jamming actively targets the availability pillar by denying its service. Leonardi et al. demonstrated that jamming using a low-cost ground-based transceiver is effective in diminishing the reception of ADS-B messages [14]. They transmitted RF signals at 1090 MHz at a power of 20 dBm with an emulated distance of 1 km from the ADS-B receiver and were able to reduce the coverage of the receiver from 400 km to 35 km [14]. Their results are summarized in Table 3.

In Table 3, the jammer outputs a signal at a constant power of 20 dBm while varying the distance from the jammer to the ADS-B receiver (indicated as 'Simulated range of the jammer'). Prx is the power of the jamming signal that is received at the ADS-B. The results show that the closer the jammer is to the ADS-B receiver, the greater its effect in diminishing the maximum range of ADS-B Out [14]. Therefore, ADS-B coverage can be further diminished using jammers with higher output power.

Table 3: Summary of ADS-B jamming at different received jammer power and jammer distance to ADS-B receiver at a constant jammer transmitted power of 20 dBm [14]

Prx (dBm)	Simulated range of the jammer (Km)	Maximum range with jammer (Km)
-90	6.92	218.78
-87	4.90	154.88
-83	3.09	97.72
-80	2.19	69.18
-77	1.55	48.98
-74	1.10	34.67

2.3.3 Message Manipulation

Message manipulation refers to attacks where adversaries insert (message injection), alter (message modification), or delete (message deletion) ADS-B messages, thereby compromising the system's integrity [9, 15]. The complexity of these attacks varies. However, all message manipulation types pose significant safety risks, including loss of situational awareness for both aircrew and ATC, potential for mid-air collisions due to missing aircraft, or disruption of air traffic management due to unnecessary diversion. This section will explore the three types of message manipulation attacks.

2.3.3.1 Message Injection

Message injection, also referred to as spoofing, is an attack that involves transmitting fabricated ADS-B messages to simulate non-existent aircraft, thereby compromising air traffic data [15]. This attack creates a "ghost aircraft", undermining the integrity of ATC systems by leading to false air traffic data. Schäfer et al. illustrated the effects of message injection using their experimental setup shown in Figure 3. Their setup consisted of the attacker's machine (Linux) that targeted the SBS-3, a commercially-available ADS-B receiver that receives real aircraft ADS-B Out messages via the connected Antenna and displays the input in a radar-visualization software on the Windows machine. The attacker generates fake, but correctly formatted, ADS-B out messages using the Linux machine, which are transmitted using the Universal Software Radio Peripheral (USRP), an off-the-shelf SDR. Schäfer et al. physically connected the USRP (A) to the SBS-3 receiver. A real attacker would likely not have this opportunity and would need to have a setup similar to USRP (M) shown in Figure 3.

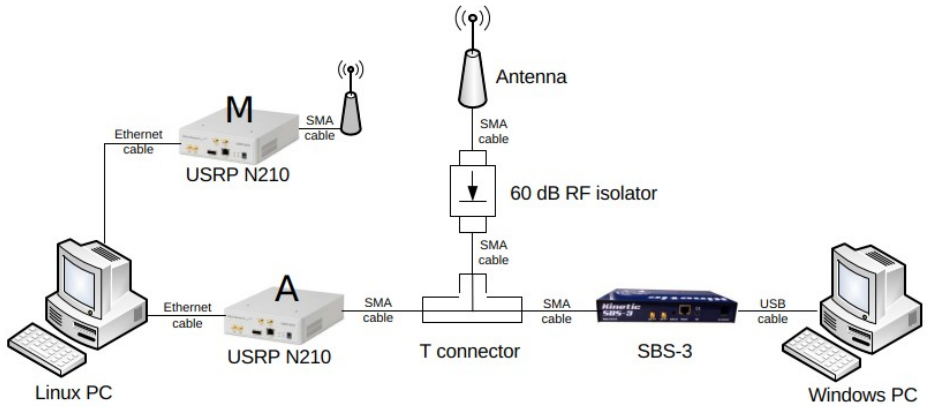


Figure 3: ADS-B message injection setup [15]

Schäfer et al. performed two message injection experiments: a single ghost aircraft injection (injecting one ADS-B message) and ghost aircraft flooding (injecting multiple ADS-B messages) [15]. Both experiments were successful in displaying the ghost(s) aircraft on the radar-visualization software. The results of the single ghost aircraft injection are found in Figure 4 which shows the injected ghost aircraft as C0FFEE, while the ghost aircraft flooding can be seen in Figure 5 which shows the effectiveness of flooding ghost aircraft on the radar-visualization software. Both attacks pose significant risks to air traffic safety and have the potential to cause confusion among ATC and aircrew, forcing unnecessary diversions to avoid collisions, and increasing the workload in the aircraft unnecessarily. Additionally, they can undermine confidence in the ADS-B system affecting its reliability and future adoption in airspace management. Although the two injection attacks are performed in the same manner, they differ by targeting different computer security pillars; a single-injected ghost aircraft targets the integrity of ADS-B, while flooding many ghost aircraft also targets the availability of ADS-B in addition to its integrity.

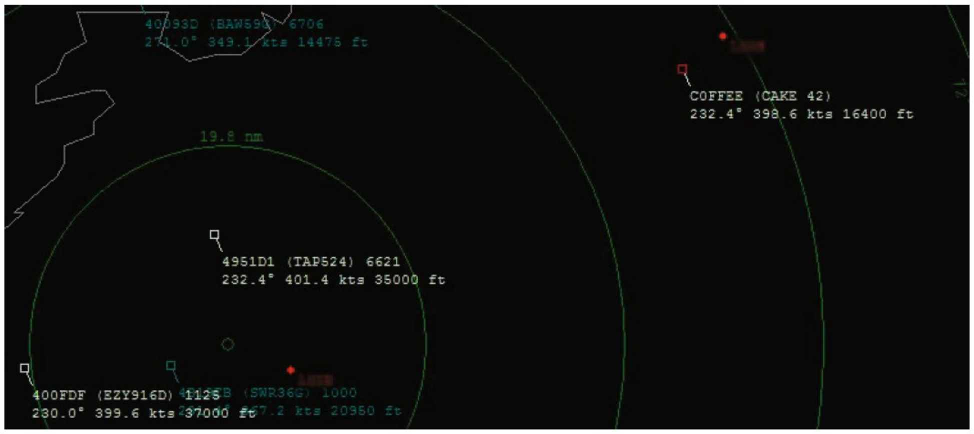


Figure 4: Single ghost aircraft injection with ICAO ID C0FFEE (top right) [15]

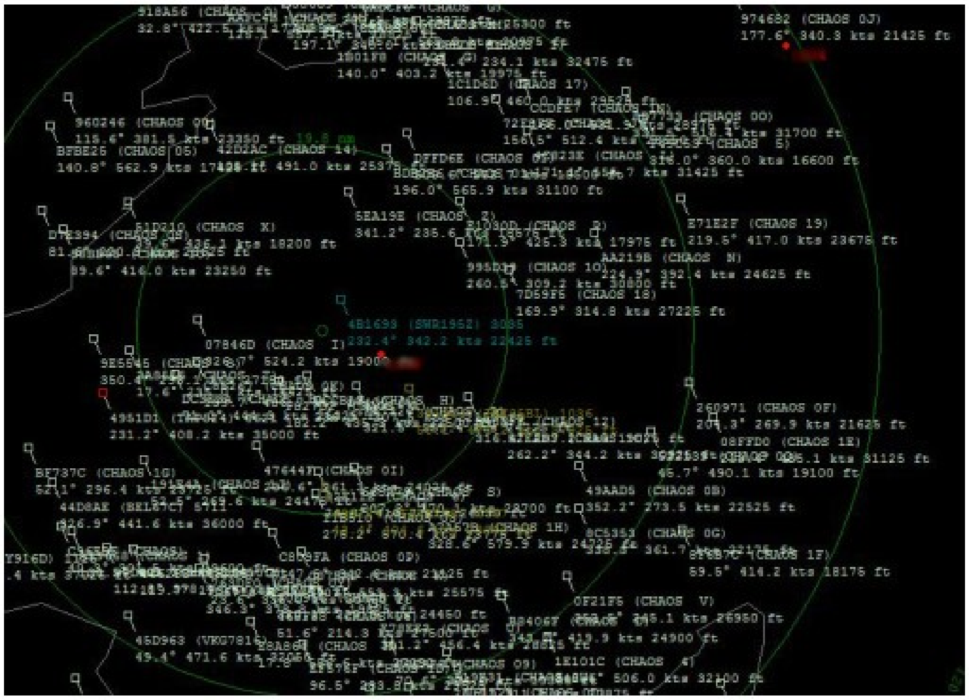


Figure 5: Ghost aircraft flooding injection using 100 randomly distributed aircraft [15]

2.3.3.2 Message Deletion

Message deletion does the opposite of a message injection attack; it aims to remove an ADS-B Out message of a legitimate aircraft so it is no longer detectable by ATC [9]. Schäfer et al. outlined two methods for executing ADS-B message deletion: through destructive interference, and through constructive interference [15]. In destructive interference, the attacker tries to nullify the legitimate ADS-B message by broadcasting its inverse, with the aim for both signals to cancel each other out at the receiver. This nullification demands precise timing and phase alignment, which is difficult to achieve with moving targets. Constructive interference introduces enough bit errors to the legitimate ADS-B Out message to exceed the error correction limits of the ADS-B system, leading to message rejection. Constructive interference is more feasible because it has less stringent synchronization requirements when compared to destructive interference.

Since protecting against message deletion attacks lies beyond the scope of this work, and the proposed solution is not designed to defend against such attacks, no further details about message deletion will be discussed in this work.

2.3.3.3 Message Modification

Message modification is similar to message injection, but instead of transmitting a ghost aircraft ADS-B message, it modifies messages from aircraft before they are received by ADS-B In [3]. Schäfer et al. described that ADS-B Out message modification can be performed using two methods: overshadowing and bit flipping [15]. Overshadowing involves overpowering the legitimate ADS-B Out message with a malicious one, making the legitimate message appear as noise. The receiver would accept the malicious message, which would have been crafted by the attacker with inaccurate aircraft information. Bit flipping changes specific bits of the ADS-B Out message in order to alter aircraft information [15]. While both methods require precise timing, overshadowing is generally more feasible than bit flipping. This is because overshadowing only requires the attacker to transmit at a higher signal strength during the message's reception window to ensure that the receiver locks onto the malicious message instead of the legitimate one. On the other hand, bit flipping requires precise synchronization with the legitimate transmission at the carrier phase level and must introduce just enough power at the precise moment to alter specific bits without corrupting the ADS-B message entirely. This is difficult to accomplish, especially with moving targets such as aircraft. For overshadowing to be successful, the malicious signal needs to be transmitted during the period when the receiver is actively processing an incoming signal and must be strong enough to dominate the legitimate transmission. Schäfer et al. calculated that an attacker could successfully modify ADS-B messages by overshadowing using low-cost off-the-shelf equipment when they are within 10 km of the receiver ground station [15].

2.4 ADS-B Attack Mitigation Techniques

Researchers have dedicated substantial efforts to developing mitigation strategies against the ADS-B vulnerabilities discussed earlier. These strategies aim to safeguard the integrity, confidentiality, and availability of ADS-B communications, ensuring correct and secure ATC data. This section will explore some of the mitigation techniques developed and tested by researchers.

2.4.1 Timestamp Authentication

One defensive strategy that was proposed by Kim et al. is the use of precise timestamp and position information as a way to authenticate ADS-B messages [16]. This can be effective against message spoofing as it allows for filtering out of malicious messages without significant delay. To implement this strategy, Kim et al. proposed an authentication process that compares the timestamp and location information contained within the ADS-B message against expected values, as shown in equation (1).

$$T_{propagation}\,(L_{s1}, L_{r1}) = dist(L_{s1}, L_{r1}) / c \qquad (1)$$

where $T_{propagation}\,(L_{s1}, L_{r1})$ is the time it takes the ADS-B message to propagate from ADS-B Out (transmitter at location L_{s1}) to ADS-B In (receiver at location L_{r1}), $dist(L_{s1}, L_{r1})$ is the distance from ADS-B Out (transmitter at location L_{s1}) and ADS-B In (receiver at location L_{r1}), and c is the propagation speed of the message. If $T_{propagation}\,(L_{s1}, L_{r1})$ exceeds a threshold value, then it would indicate a message injection attack and the message in question would be rejected [16]. Figure 6 illustrates the propagation time of an ADS-B message as indicated by equation (1). It depicts an example where the receiver is another aircraft utilizing ADS-B In, but the same concept can be applied if the receiver was a ground station at ATC, as demonstrated earlier in Figure 1.

Standard ADS-B messages do not include timestamp data in their frame structure. To overcome this, the researchers in [16] propose modifying the ADS-B protocol to embed an 8-bit timestamp within the message. This is achieved by reducing the PI block from 24 to 16 bits and using the remaining 8 bits to store the timestamp. The researchers justify this modification by demonstrating that a 16-bit PI block still provides sufficient error-detection capability for ADS-B messages.

To evaluate the defence, the researchers simulated 100 legitimate aircraft and 100 ghost aircraft sending spoofed ADS-B messages. Using the criteria outlined in equation (1), 99% of the ghost aircraft were detected and their ADS-B messages were discarded, while all of the legitimate aircraft ADS-B Out messages were successfully received, achieving a 0% false positive rate [16].

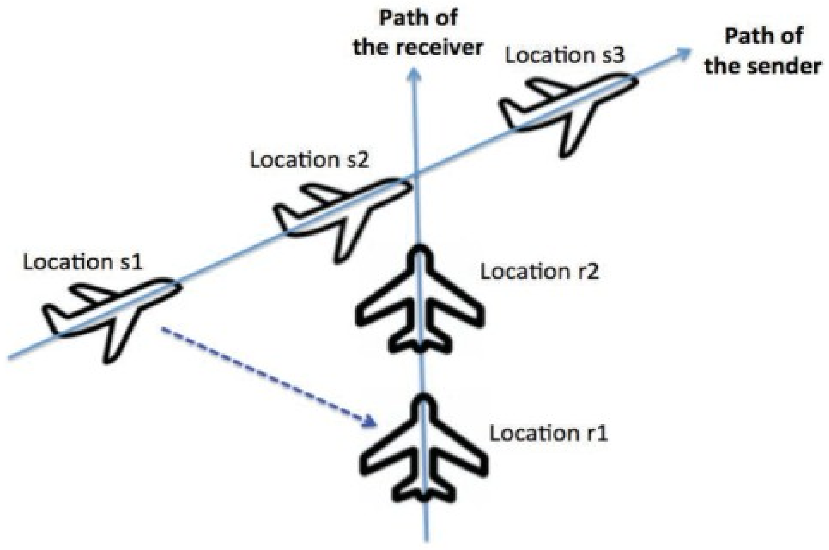


Figure 6: Propagation time of ADS-B message [16]

2.4.2 Multilateration

Another defensive strategy that was proposed utilizes multilateration. This technique relies on multiple independent receivers and sensors (at least three for two-dimensional positioning, and at least four for three-dimensional positioning) to calculate the aircraft's position [17]. It does so by analyzing the Time Difference of Arrival (TDOA) of signals from the aircraft to these receivers and sensors. The receivers/sensors do not need to be the same; one can be primary radar, another can be secondary radar, or a Traffic Collision Avoidance System [17, 18], as shown in Figure 7. This technique analyzes and compares the signals from each receiver to verify the location of an aircraft and determine if a message injection, modification, or deletion attack is suspected. Multilateration complements ADS-B systems to enhance the integrity of aircraft surveillance. This can be implemented by utilizing the existing ground station used to track aircraft position, conducting time difference and calculation, and conducting the position calculation using the TDOA measurements [17].

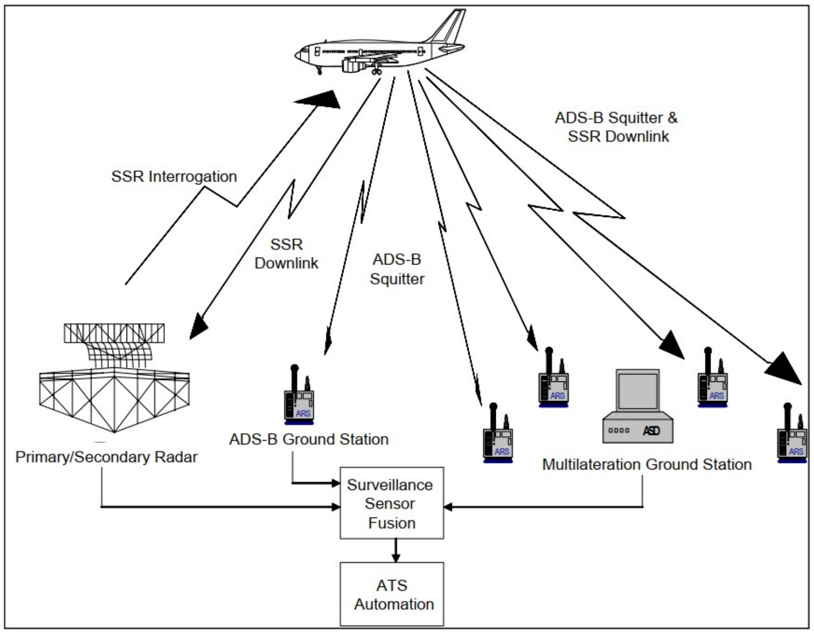


Figure 7: Multilateration implementation [17]

2.4.3 Multichannel Receiver

Using a multichannel receiver is another strategy that was proposed by researchers to defend against jamming attacks in ADS-B systems. Leonardi et al. proposed the use of algebraic manipulation using singular value decompression (SVD) to effectively differentiate and filter out jamming signals from legitimate ADS-B transmissions by utilizing Angle of Arrival (AOA) [9]. The researchers consider several independent sources, such as different ADS-B messages and potential jamming signals, being received by an antenna array connected to the receiver.

The researchers evaluated the multichannel's ability to mitigate jamming on ADS-B signals in [9] by conducting trials with four different types of jammers, ranging from a signal-to-interference-ratio (SIR) of -23 dB to 6 dB, over several hundred real ADS-B signals. The jamming signals were then injected along with clean signals, the Extended Projection Algorithms (EPA) decoding algorithm was applied, and the results were compared to the clean ADS-B signals [9]. The results are shown in Figure 8.

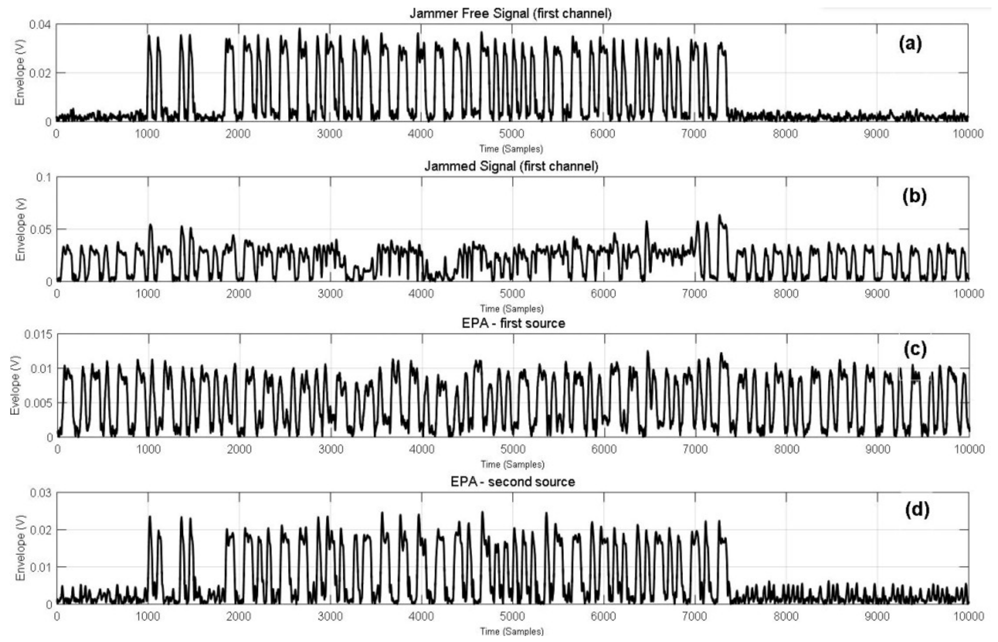


Figure 8: Evaluation results of utilizing multichannel receivers against a square wave jammer with SIR = 0 dB: (a) ADS-B signal with no jamming, (b) jammed signal envelope, (c) first separated source envelope using EPA, (d) second separated source envelope using EPA [9]

The results show that at SIR of 0 dB, the EPA algorithm was able to differentiate the ADS-B signal (Figure 8 (d)) from the jamming signal (Figure 8 (c)). Also, the researchers evaluated the results at the most challenging scenario, where the SIR was -26 dB, and found that without utilizing EPA, the receiver lost the ability to decode ADS-B signals with any jammer type [9]. When the EPA was applied, the loss rate was reduced to 22.7%, meaning that 77.3% of messages were successfully recovered.

2.4.4 Message Encryption

Several encryption techniques have been proposed by researchers as a defensive strategy against eavesdropping and some forms of message manipulation in ADS-B systems. This section will discuss works that proposed or applied encryption algorithms specifically for ADS-B systems.

2.4.4.1 Advanced Encryption Standard

Zhang et al. utilized Advanced Encryption Standard-128 (AES-128), a widely-recognized symmetric encryption algorithm known for its balance of robustness and low computational effort, to encrypt ADS-B messages as shown in Figure 9.

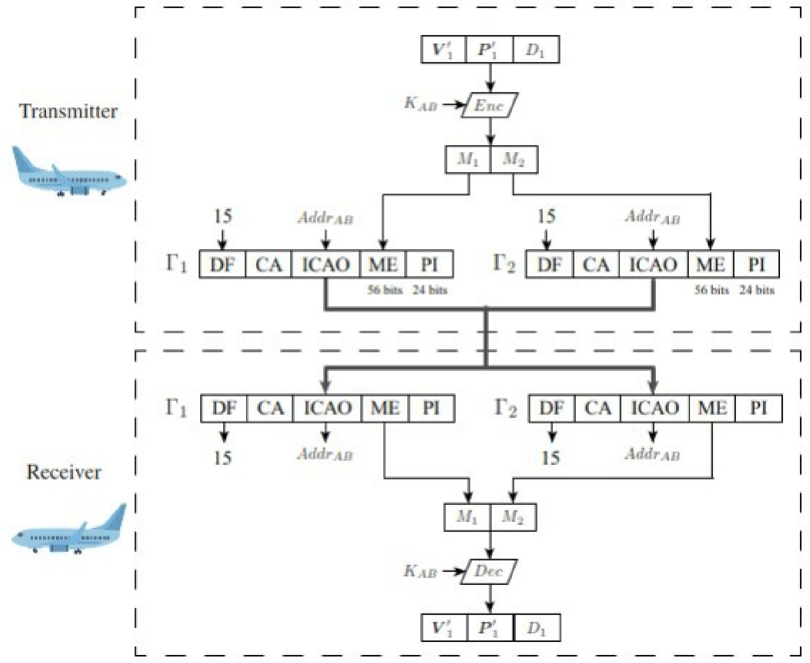


Figure 9: AES encryption of ADS-B messages [8]

In Figure 9, ADS-B Out uses the aircraft's velocity, position, and message timestamp, denoted by V'_1, P'_1, and D_1, respectively, and passes them through the AES algorithm, denoted by Enc. The AES algorithm uses an encryption key, K_{AB}, resulting in a 128-bit ciphertext, M, containing the payload. The goal is to fit the ciphertext into the ME block. However, since the ME block is only 56 bits, M is split into two messages, M_1 and M_2, each holding 64 bits of encrypted text, and sent using the two ME blocks from two separate ADS-B messages. Zhang et al. proposed to reduce the PI block down to 16 bits to allow the ME block to increase to 64 bits [8]. ADS-B Out will then transmit the two encrypted messages, where an ADS-B In receiver can decrypt the messages using the same key, K_{AB}, after reassembly in order to read the payload. While the paper does not discuss the impact of reducing the size of PI block to 16 bits, this change would likely decrease ADS-B's error detection capability, potentially increasing the number of dropped messages, albeit enhancing message security.

2.4.4.2 Stage Identity-Based Encryption

Baek et al. proposed the use of Staged Identity-Based Encryption (SIBE) as a method to secure ADS-B communications that utilizes both symmetric and asymmetric encryption techniques [19]. This scheme relies on the Public Key Infrastructure where the public key is derived from publicly known information, such as an aircraft's ICAO address, registration number, or flight details. A private key generator is responsible for generating the corresponding private keys. The use of SIBE eliminates the need for a traditional certificate authority and reduces the overhead associated with public key lookup.

In this approach, SIBE is used to establish a session key between aircraft, which is then employed for symmetric message authentication [19]. The symmetric key is used in conjunction with a Message Authentication Code (MAC) to ensure the integrity and authenticity of each ADS-B message broadcast. Baek et al. highlight that this scheme adds minimal computational overhead to ADS-B messages while improving protection against message injection and spoofing attacks. However, challenges related to the distribution and management of private keys, especially on a global scale, remain an area for further research.

2.5 Gaps in Existing Defensive Strategies

While the surveyed defensive strategies for ADS-B systems – timestamp authentication, multilateration, multichannel receivers, AES encryption, and SIBE – show promise in mitigating specific vulnerabilities, certain gaps persist. For instance, as shown in Table 4, timestamp authentication, despite its effectiveness in identifying spoofed messages, does not address the potential for eavesdropping, replay, or message modification attacks. This limitation underscores the need for a layered security approach that incorporates additional measures to mitigate a wide range of threats. Similarly, multilateration and the use of multichannel receivers, while effective in enhancing location verification and jamming mitigation, require substantial infrastructure investments and their implementation may face challenges in scalability and integration with existing ATC systems. ADS-B message encryption using AES, though robust in safeguarding message confidentiality, necessitates additional bandwidth for the transmission of encrypted messages, potentially complicating implementation within the current ADS-B framework. Both encryption methods, AES and SIBE, raise concerns as to the computational burden associated with the selected cryptographic operations, which can affect system performance, especially in terms of latency.

Table 4 summarizes the effectiveness of the defensive techniques discussed in this literature review against each vulnerability along with the implementation difficulty. The implementation difficulty was assessed by the author based on the complexity of applying the defensive technique, the required changes to existing ADS-B infrastructure, and the associated cost of implementation.

Table 4: Summary of defensive techniques addressing ADS-B vulnerabilities and implementation difficulty

	Timestamp Authentication	Multilateration	Multichannel Receiver	AES	SIBE
Eavesdropping				✓	✓
Jamming			✓		
Message Injection	✓	✓		✓	✓
Message Deletion		✓			
Message Modification		✓		✓	✓
Implementation Difficulty	Low	Medium	Medium	Medium-High	High

2.6 Format Preserving Encryption

Format-Preserving Encryption (FPE) is a type of encryption where the ciphertext maintains the same size and format as the plaintext [20]. This is particularly useful for applications such as encrypting credit card numbers or other fixed-format data in legacy systems and protocols where the data structure must remain unchanged. Traditional encryption methods like AES and RSA do not preserve the format, resulting in ciphertext that may be incompatible with existing systems that expect data in a specific format. Given the limitations of many existing ADS-B attack mitigation techniques, FPE presents a promising alternative by enhancing security without altering the ADS-B framework.

Agbeyibor et al. state that FPE can be highly beneficial for securing legacy critical infrastructure systems that were not originally designed with modern security measures in mind [21]. The study highlights three FPE algorithms recommended by the National Institute of Standards and Technology (NIST): FF1, FF2, and FF3. These algorithms are derived from AES and are designed to encrypt data without altering its format, making them suitable for use in systems with strict format requirements. The evaluation conducted in [21] shows that these FPE algorithms provide high levels of security while maintaining operational efficiency. FF1 supports the widest range of data lengths, FF2 provides additional protection against side-channel attacks, and FF3 is the simplest and fastest encryption method, making it suitable for resource-limited environments. In this research, FF1 and FF2 were not considered as potential FPE algorithms because FF3's lower computational complexity and reduced hardware requirements make it more suitable for resource-constrained systems like an embedded computer.

FPE provides an effective way to enhance ADS-B integrity without requiring modifications to its structure by preserving the message format and length. The use of FPE also ensures that encryption can be implemented using resource-limited environments. This makes its application in real-world aviation more feasible.

2.6.1 FF3

FF3 is a symmetric FPE algorithm derived from AES and standardized by NIST in [22]. It employs a Feistel network to achieve FPE which allows FF3 to maintain the same size and structure of the plaintext in the resulting ciphertext. Figure 10 outlines the general Feistel structure where the plaintext is divided into two halves, denoted as A and B, before encryption begins [22]. The number of characters in these halves is represented as u and v, with their sum denoted as n. Each round applies a round function, $F_K(n,T,i)$, which derives its transformation from the encryption key, a 64-bit tweak T, and the round number i. The output of each round influences the subsequent round. At the end of each round, the roles of the left and right halves are swapped before the next iteration begins. This process gets repeated which ensures that every part of the plaintext contributes to the final ciphertext. FF3 uses eight rounds which is reduced from the 10 that FF1 uses. This achieves a balance between security and computational efficiency [22].

A critical component in FF3 is the radix, which refers to the base of the numeral system used to represent the plaintext and ciphertext. The radix will determine the length requirements of the plaintext according to the below requirements [22]:

$$\text{radix} \in [2 \ldots 2^{16}] \tag{2}$$

$$\text{radix}^{\text{minlen}} \geq 100 \tag{3}$$

$$2 \leq \text{minlen} \leq \text{maxlen} \leq 2 \lfloor \log_{\text{radix}} (2^{96}) \rfloor \tag{4}$$

These three equations determine the minimum length (*minlen*) and the maximum length (*maxlen*) of the plaintext and ciphertext depending on the *radix* value. As shown in equation 4, the *minlen* is always 2 as FF3 requires at least two halves to function. For a *radix* of 2 (binary), the *maxlen* must satisfy equation 4 which yields a *maxlen* of 96 characters, and for a *radix* of 10 (decimal), the *maxlen* is 28 characters. As for the key length, FF3 supports key lengths of 128, 192, or 256 bits [22].

Another key feature in FF3 is the use of a tweak, which is a user-defined parameter. The tweak adds variability into the encryption process, which ensures that even if the same plaintext is encrypted multiple times with the same key, the resulting ciphertext will be different [22]. The tweak, therefore, prevents patterns in the ciphertext that could otherwise be analyzed to find weaknesses in the encryption algorithm. It is important to note that the tweak should not necessarily be kept secret, like the encryption key. Instead, it could be combined with the plaintext and fed into the encryption algorithm in a manner similar to the use of password salt. FF3 requires that the tweak length be 64 bits.

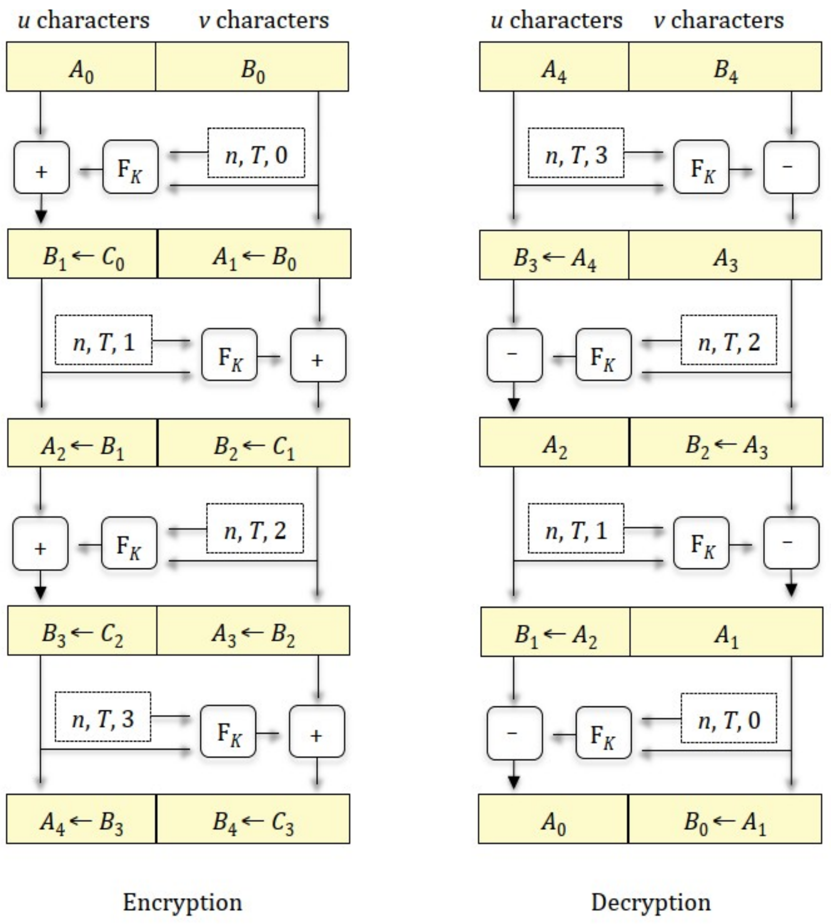


Figure 10: Feistel Structure of FF3 [22]

2.6.2 FFX

FFX is also a symmetric FPE algorithm, but unlike FF3, it is designed as a flexible framework where some parameters (number of rounds, tweak length, etc.) can be customizable by the user to meet specific security and performance requirements [23]. For example, selecting 8 rounds and a 64-bit tweak aligns with the specifications for FF3, while using 10 rounds will result in an algorithm similar to FF1. This makes FFX highly flexible which allows it to be tailored to various applications.

2.6.3 AES-CTR

Advanced Encryption Standard in Counter Mode (AES-CTR) is a symmetric encryption algorithm that uses a stream cipher [24]. AES-CTR is not an FPE algorithm, but it does preserve the length of the plaintext if it was less than 128 bits long, unlike other modes of AES. This is because AES-CTR does not require padding of plaintexts to match the fixed block size of AES.

The encryption process in AES-CTR starts with the initialization of a 128-bit counter block, composed of a nonce and an Initialization Vector (IV). The nonce, similar to a tweak in FPE, ensures the encryption process produces unique outputs even when the same plaintext and key are used multiple times. The IV should be 64 bits long. The counter block is incremented for each subsequent block of plaintext, and each block is processed through AES encryption to generate the keystream. The keystream is then XORed with the plaintext to produce the ciphertext. AES-CTR supports key sizes of 128, 192, and 256 bits with the number of rounds being 10 when a 128-bit key is used.

3 Methodology

This chapter outlines the experimental setup, implementation, and evaluation of the three FPE algorithms for enhancing the integrity of ADS-B messages. The methodology is structured to reflect the design and execution of the experiment, beginning with an overview of the system architecture and components, followed by the encryption and decryption processes. It then details the performance evaluation criteria used to assess the computational feasibility of the selected algorithms and embedded computer. Additionally, KMD challenges are discussed along with conceptual solutions for securely transmitting keys in a real-world aviation environment.

3.1 System Design

This section outlines the proposed architecture, intended as a future onboard solution, and the experimental setup that was implemented to evaluate the effectiveness of the selected FPE algorithms in enhancing the integrity of ADS-B messages. The design consists of two main components: an in-line device as the embedded computer onboard the aircraft used for encrypting ADS-B messages, and a Windows-based virtual machine (VM) simulating ATC. These two components were placed in the same network to enable real-time encryption, transmission, decryption and validation of ADS-B messages.

3.1.1 Proposed Architecture

Unlike the setup shown in Figure 1, where the data flows directly from the ADS-B Equipment to the antennas to be broadcast, the proposed architecture places an in-line device as the embedded computer between the ADS-B Equipment and the ADS-B Out antennas as shown in Figure 11. This allows the in-line device to intercept all outgoing ADS-B messages. Our solution treats the ADS-B Equipment as a black box, meaning it does not require any modifications or any detailed understanding of its internal workings. This approach ensures compatibility with existing ADS-B systems while simplifying integration. This is possible because all ADS-B systems adhere to the same protocol. Therefore, as long as the protocol is understood, the solution can be applied.

Figure 12 provides a closer look at the overall architecture of the in-line device. In this design, the raw ADS-B message first passes through an analogue-to-digital converter (ADC) which digitizes the analogue output of the ADS-B Equipment to enable further processing and encryption. Next, the ME Extraction and Encryption Module (highlighted in yellow in Figure 12) identifies and extracts the ME block and applies FPE. Then, the message is reassembled with the encrypted ME block and the checksum is recomputed and stored in the PI field. Finally, it is passed to the digital-to-analogue converter (DAC) before being broadcast using the ADS-B Out Antenna.

In this thesis, the primary focus is on the ME Extraction and Encryption Module. While the overall in-line device includes other components (ADC, Reassembler, Checksum calculator, and DAC), the key research question addressed here focuses on how to extract and encrypt the ME block using FPE in order to enhance the integrity of ADS-B while respecting the structure and transmission constraints of the ADS-B protocol.

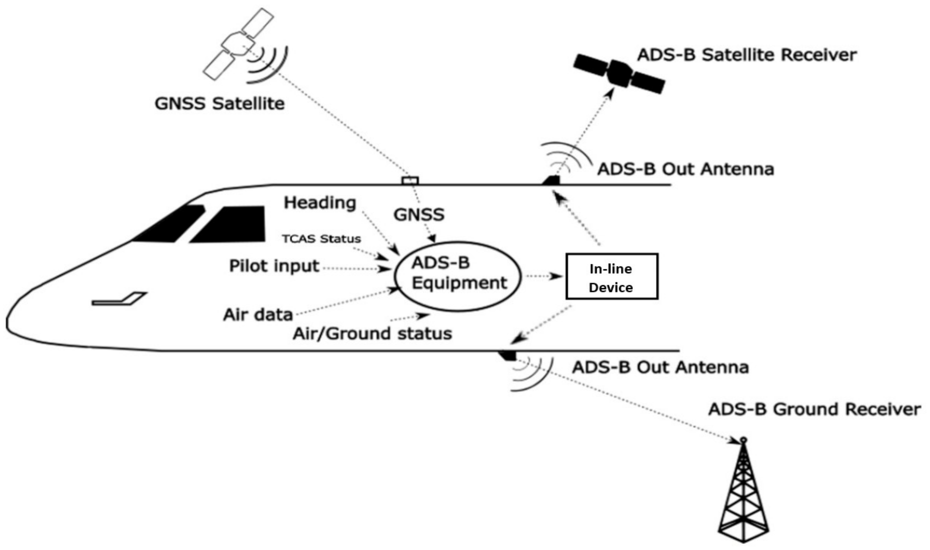


Figure 11: Proposed solution of placing in-line device between ADS-B Equipment and ADS-B Out Antenna for encryption of outgoing ADS-B messages. Adapted from [6]

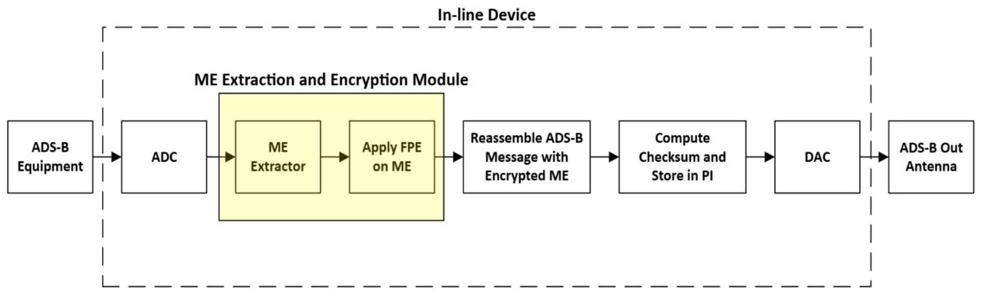

In-line Device
ME Extraction and Encryption Module
ADS-B Equipment
ADC
ME Extractor
Apply FPE on ME
Reassemble ADS-B Message with Encrypted ME
Compute Checksum and Store in PI
DAC
ADS-B Out Antenna

Figure 12: In-line device architecture

3.1.2 Experimental Setup and Hardware

The experimental setup and hardware of the ME Extraction and Encryption Module include the following components:

1. Software Defined Radio (SDR): An SDR dongle was used to capture actual ADS-B messages which were logged to a file in hexadecimal format for use during the encryption process.
2. Raspberry Pi (RP): A RP 5 with 4 GB of RAM and an Arm Cortex-A76 quad-core processor was used as the embedded computer. The selection of the embedded computer was based on the need for a compact, low-power, and cost-effective embedded system capable of performing encryption in real time without introducing significant latency. Several embedded computers were considered but the RP was selected for its computational efficiency, compatibility with Python-based encryption libraries, small form-factor, and affordability, making it suitable for an embedded computer that could be deployed in real-world aviation to encrypt ADS-B messages. The RP was configured with RP OS, a Debian-based operating system, to provide an adequate environment for running custom scripts.
3. ATC VM: A Windows-based VM, running Windows 11, simulated the ATC environment. The VM was equipped with Python libraries such as `pyffx`, `ff3,` and `pyModeS` for decrypting and decoding ADS-B messages.

The RP and ATC VM were configured within the same local network. This allowed the RP to communicate with the ATC VM wirelessly, using the built-in Wi-Fi module on the RP.

3.1.3 Experiment Workflow

The experimental setup followed a structured workflow to evaluate the encryption and decryption process, as shown in Figure 13:

1. Message Capture and Processing: Actual ADS-B messages from local air traffic were captured over several hours using the SDR dongle. The messages were filtered to ensure a mix of different types of ADS-B messages and stored in a log file. This allowed the RP to read the messages from this log, simulating the role of an onboard encryption system without relying on live message capture during the experiment. The log file contained 1,000 different ADS-B messages, which we considered sufficient for our experimental purposes.

2. Encryption on the RP: The RP read the pre-recorded ADS-B messages from the log file, identified the ME block, and applied encryption using one of three FPE algorithms: FF3, FFX, and AES-CTR as discussed in section 3.2. The algorithms ensured the ME block maintained the same length and format required by the ADS-B protocol. The encrypted ME blocks were then transmitted to the ATC VM via a Transmission Control Protocol (TCP) connection.
3. Decryption and Validation on the ATC VM: The ATC VM received the encrypted ME blocks along with the original unencrypted ADS-B messages from the RP and decrypted the ME block using the corresponding FPE algorithm and encryption key. The decrypted ME block was compared with the original to verify the integrity of the message. The decrypted messages were also decoded and analyzed for correctness.

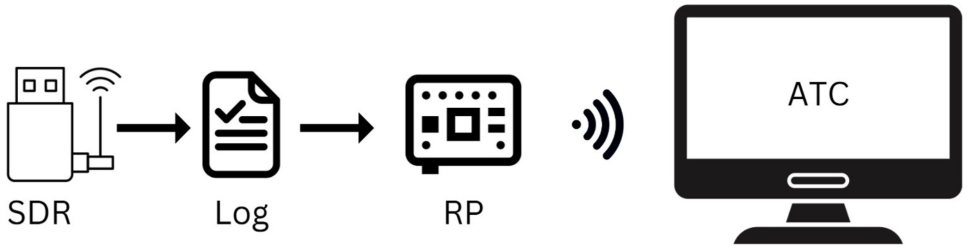


Figure 13: Experiment workflow

During the experiment, key performance metrics were recorded to evaluate the computational feasibility of each FPE algorithm. These will be discussed in section 3.3.

3.1.4 Limitations and Assumptions

The experimental setup aimed to simulate real-world conditions, as much as practicable, with some limitations. The use of pre-recorded ADS-B messages instead of live capture of ADS-B messages is one such limitation. This limitation means that the experimental setup loses some variability and unpredictability of real-time message transmission, but it did ensure a consistent dataset for testing.

Another limitation was the use of a TCP connection for the transmission of encrypted ADS-B messages from the RP to the ATC VM. This does not replicate the broadcast nature of the 1090 MHz radio frequency used in real-world ADS-B systems, but it did provide a reliable communication channel for a controlled evaluation. In a live implementation, where the encrypted messages are transmitted via an antenna over the air, factors such as signal interference and range limitations would exist, which were not accounted for in this experiment. In a traditional ADS-B system broadcasting messages over 1090 MHz, corrupted messages are handled by the PI block, as explained in section 2.1, where any errors in the received ADS-B message result in that message being discarded. This is usually not a significant issue since aircraft continuously send ADS-B messages multiple times per second, ensuring that even if some messages are lost or corrupted, subsequent transmissions provide the necessary information to ATC and other aircraft.

Finally, the experimental setup did not include real ADS-B equipment, a real or simulated aircraft environment, or real antennas for transmission and reception (apart from the antenna in the SDR used to capture ADS-B messages for the data set). The RP simulated an embedded system onboard the aircraft for encryption, but without integration into actual aircraft systems. Similarly, the ATC VM used custom scripts and open-source tools for decryption and decoding ADS-B messages, rather than official ATC software. Therefore, the results may not fully reflect the performance or feasibility of deployment in real-world aviation scenarios.

The experiment also assumed secure KMD, which is likely more challenging to implement in a real-world scenario. Section 3.4 will discuss KMD in more detail and propose some solutions to overcome this challenge.

3.2 Implementation of FPE Algorithms

The implementation of FPE algorithms in this experiment was carried out on an RP simulating the onboard embedded system for encryption, and a Windows-based VM simulating an ATC environment. This section discusses the implementation details of each of the three FPE algorithms, including the encryption and decryption processes and how messages were validated for correctness.

3.2.1 FF3 Implementation

The FF3 algorithm was implemented to encrypt the ME block of ADS-B messages while preserving the length and format of the original message. The implementation involved two main components: the RP, which handled the encryption, and the ATC VM, which handled the decryption and validation processes.

3.2.1.1 FF3 Raspberry Pi Implementation

The RP processes pre-recorded ADS-B messages from the log file, simulating real-world message capture. The encryption process included the following steps:

1. Message Capture: The RP read ADS-B messages from the log file, which contained 1,000 valid messages, one message at a time. Each message consisted of 28 hex characters, representing 96 bits including the ME block (14 hex characters representing 56 bits), which was extracted for encryption.
2. Encryption: The FF3 algorithm was implemented using the `ff3` Python library from [25]. The `ff3` library only accepts a decimal input so the ME block was converted to a decimal string for encryption. The FF3 cipher was initialized using a 256-bit key and a 64-bit tweak.
3. Transmission to ATC VM: The encrypted ME block was transmitted to ATC VM, as a decimal string, along with the original unencrypted ADS-B message (for validation purposes). The transmission was done using a TCP connection.

The encryption and transmission steps continued until all 1,000 ADS-B messages from the log were processed.

3.2.1.2 FF3 ATC VM Implementation

The ATC VM received the encrypted messages from the RP and was responsible for the decryption and validation of the ADS-B messages. The decryption and validation processes included the below steps for all 1,000 messages received from the RP.

1. Message Reception: After a connection was established with the RP, the ATC VM received the original ADS-B message and the encrypted ME block, one message at a time.
2. Decryption: The decryption process used the same FF3 cipher configuration as on the RP during encryption (key and tweak). After decrypting the ME block, it was converted back to hexadecimal to reconstruct the full ADS-B message.
3. Message Reconstruction and Validation: The decrypted ME block (now in hexadecimal) was reintegrated into the rest of the ADS-B message using the remaining characters from the received original ADS-B message. The reconstructed ADS-B message was then compared with the original plaintext message to validate the encryption/decryption process. Any discrepancies were flagged as validation failures.
4. Message Decoding: In addition to decryption and validation, every ADS-B message was also decoded using the `pyModeS` library from [26] to verify the message types and fields (aircraft identification, position, and velocity).

3.2.2 FFX Implementation

The FFX algorithm was implemented to encrypt the ME block of ADS-B messages while preserving the length and format of the original message. The encryption, decryption, and validation process followed a similar process to that of FF3, with some notable differences which will be explained below.

3.2.2.1 FFX Raspberry Pi Implementation

Similar to the FF3 implementation, the RP processed the pre-recorded ADS-B messages from the log file and followed the encryption process shown below.

1. Message Capture: The same 1,000 ADS-B messages were processed, similar to the process used in FF3 from section 3.2.1.1.
2. Encryption: The FFX algorithm was implemented using the `pyffx` Python library from [27]. Unlike the `ff3` library, `pyffx` accepts the input as hexadecimal strings. Therefore, the ME block captured was directly encrypted using a 256-key configuration and without using a tweak. As explained in section 2.6.2, FFX allows for custom parameters such as number of rounds. 10 rounds were used in this implementation to have a balance between increased security when compared to the eight rounds used in FF3 and high efficiency during the encryption process.
3. Transmission to ATC VM: Similar to that of FF3 from section 3.2.1.1.

3.2.2.2 FFX ATC VM Implementation

Similar to the FF3 implementation, the ATC VM received the encrypted messages from the RP and was responsible for the decryption and validation of the ADS-B messages. This process continued until all 1000 messages were processed as per the steps below:

1. Message Reception: The same 1,000 ADS-B messages were processed, similar to the process used in FF3 from section 3.2.1.2.
2. Decryption: The decryption process used the same FFX cipher configuration as on the RP during encryption (256-bit key).
3. Message Reconstruction and Validation: Similar to that of FF3 from section 3.2.1.2.
4. Message Decoding: Similar to that of FF3 from section 3.2.1.2.

3.2.3 AES-CTR Implementation

The AES-CTR algorithm was implemented to encrypt the ME block of ADS-B messages. As discussed in section 2.6.3, AES-CTR is not a true FPE algorithm, however, it does preserve the length of the plaintext in this implementation since the ME block is smaller than 128 bits. The encryption, decryption, and validation processes followed a similar structure to that of FF3 and FFX, with some notable differences as explained below:

3.2.3.1 AES-CTR Raspberry Pi Implementation

The RP handled the encryption process for ADS-B messages as follows:

1. Message Capture: The same 1,000 ADS-B messages were processed, similar to the process used in FF3 from section 3.2.1.1.
2. Encryption: The AES-CTR encryption was implemented using the `cryptography.hazmat` Python library from [28]. The ME block was converted to bytes for compatibility with the library. A 256-bit key and a 128-bit nonce were used.
3. Transmission: Similar to that of FF3 from section 3.2.1.1.

3.2.3.2 AES-CTR ATC VM Implementation

The ATC VM received the encrypted messages and handled the decryption and validation processes as follows:

1. Message Reception: The same 1,000 ADS-B messages were processed, similar to the process used in FF3 from section 3.2.1.2.
2. Decryption: The decryption process used the same AES-CTR cipher configuration as on the RP during encryption (256-bit key). After decryption, the key was converted back to hex to preserve its format.
3. Message Reconstruction and Validation: Similar to that of FF3 from section 3.2.1.2.
4. Message Decoding: Similar to that of FF3 from section 3.2.1.2.

3.3 Performance Evaluation Criteria

This section outlines the criteria used to evaluate the performance of the three selected FPE algorithms: FF3, FFX, and AES-CTR. Each metric forming this criteria was chosen to assess the feasibility of implementing these algorithms on the RP embedded system, simulating an on-board encryption system, and on the Windows-based ATC VM, simulating an ATC environment. The evaluation metrics included encryption and decryption times as well as the standard deviation of encryption times, CPU and memory usage, and the thermal behaviour of the RP. This performance evaluation served two purposes: to assess the selected embedded system in encrypting ADS-B messages, and to determine the most suitable FPE algorithm for encrypting ADS-B messages. The following sections describe the setup and methodology used for measuring each metric. Each performance metric discussed in this section was calculated using a dedicated script or set of scripts. For example, encryption and decryption times along with the standard deviation of encryption times were all measured separately using a single set of scripts, and CPU and memory usage metrics were measured through another set of scripts independently. This approach ensured accurate and independent measurements for each metric, without risking interference in measurements due to the recording of multiple metrics all at once.

3.3.1 Encryption and Decryption Times

Encryption and decryption times are critical metrics in evaluating the latency introduced by the encryption process. As shown in Table 2, ADS-B messages are transmitted as frequently as every 0.5 seconds. Therefore, it is essential to ensure that encryption and decryption of the ME block do not introduce delays that could compromise the system's performance. The RP handled the encryption time measurement, while the ATC VM handled the decryption time measurement.

3.3.1.1 Encryption Time on the Raspberry Pi

The encryption time was measured on the RP during the encryption of the ME block of each ADS-B message. The Python library `time.perf_counter` was used to record the start and end times of the encryption, just before and just after the encryption function of each FPE algorithm. This was done for every one of the 1,000 ADS-B messages. However, to simulate the transmission rate of a real aircraft, a delay of 0.5 seconds was introduced between each message. After each message was encrypted and the encryption time recorded, the encrypted ME block, the original ADS-B message, and the encryption time were transmitted to the ATC VM via TCP for further processing.

3.3.1.2 Decryption Time on the ATC VM

On the ATC VM, the decryption time was measured using the same method as in encryption; when an encrypted ADS-B message was received, the ATC VM recorded the start time just before decryption, decrypted the message, then recorded the end time right after the decryption process ended.

In addition to measuring decryption time, ATC VM also received the encryption time for each message from the RP. The encryption times, decryption times, and total time were all recorded in a CSV file for analysis. This was done for all 1,000 messages from the log.

3.3.1.3 Standard Deviation of Encryption Times

In addition to the encryption and decryption times, the standard deviation of the encryption time was also calculated to measure the variability in the time taken to encrypt ADS-B messages. A low standard deviation is desirable for maintaining consistent message delivery, as it ensures that the encryption process done by the embedded system does not introduce significant variability, which would impact system performance in a real-world deployment.

The standard deviation of encryption was calculated on the ATC VM by using the `statistics.stdev` function in Python with all the encryption times received from the RP. To ensure the reliability of this metric, the experiment was executed 25 times, which resulted in 25 values of standard deviation. The final standard deviation value of encryption time was calculated as the average of these 25 values.

3.3.2 CPU and Memory Usage

The CPU and memory usage were recorded to evaluate the computational overhead introduced by the encryption process on RP. These metrics are important for evaluating the feasibility of using the selected embedded system for encrypting ADS-B messages using an FPE algorithm. The CPU and memory usage measurements were only done on the RP during the encryption phase; the ATC VM was not involved in these measurements. This is justifiable because the system design assumed that the ATC equipment would have sufficient computational resources to handle the decryption of FPE algorithms.

The CPU usage was measured using the `resource` Python library to capture both user and system times of the encryption process on the RP. The user time represents the total time spent executing the encryption logic, while the system time represents the total time spent executing in kernel mode to support the encryption process [29].

The script recorded both the user and system times at the beginning and at the end of the encryption process. The difference between these two values provided the total CPU time used for encrypting all 1,000 ADS-B messages. Also, the runtime was recorded to allow the calculation of CPU utilization as a percentage of single-core capacity by dividing the total CPU time by the total runtime. This procedure was executed 25 times, and the final CPU usage value was calculated as the average of these 25 measurements.

Memory usage was recorded using the `psutil` Python library using the `memory_info().rss` function, which returns the memory the encryption process has used [30]. During the encryption of each ADS-B message, the script checked the current memory usage value and updated the peak memory usage value if the current value exceeded the previous peak. This ensures that the peak memory used during the entire encryption process is recorded. The peak memory was captured in megabytes (MB) and also displayed as a percentage of the total system memory. Similar to the CPU measurement, this procedure was executed 25 times, and the final memory usage value was calculated as the average of these 25 measurements.

3.3.3 Thermal Behaviour of the Raspberry Pi

The thermal behaviour of the RP during encryption was evaluated to ensure the selected embedded system could sustain the continuous encryption operations without overheating. Excessive heat could lead to failures and/or thermal throttling which would affect performance. This metric is important to measure, especially for a device intended to be installed on aircraft.

To ensure accurate temperature measurements, the RP was first powered on and left to idle without any load for 60 minutes to allow the temperature to stabilize. Additionally, before starting temperature monitoring, the RP was confirmed to be at room temperature. For each FPE algorithm, a Python script was executed on the RP to encrypt ADS-B messages continuously without delay. The script simulated an extreme case of load by encrypting 1,000,000 ADS-B messages as quickly as the RP could do so while measuring the RP CPU temperature every second. The 1,000,000 messages were sourced by repeating the 1,000 messages from the ADS-B log file. Each temperature reading was logged to a CSV file along with corresponding timestamps and the message count for later analysis. This experiment was conducted at room temperature and all FPE algorithms were tested in the same manner (ensure RP is at room temperature, power on and idle for 60 minutes, then encrypt 1,000,000 messages while recording the temperature every second). The temperature of the RP CPU was recorded using the `get_cpu_temperature()` function. All measurements were done exclusively on the RP; the ATC VM was not involved in the temperature monitoring.

3.4 Key Management and Distribution

KMD is a challenge with many symmetric encryption algorithms, including in the proposed solution of using an FPE algorithm. The experimentation or testing on a KMD solution is outside the scope of this work, but this section does present a conceptual framework for addressing KMD using a hybrid encryption model and also explores the potential of leveraging existing aviation communication systems for the distribution of encryption keys securely.

3.4.1 Hybrid Encryption Model

The hybrid encryption model proposed to solve the KMD challenge is similar to the encryption model used in [19], known as the SIBE, with one key difference. After relying on asymmetric encryption to exchange the symmetric key, SIBE proposes the use of AES-128 to encrypt the ADS-B messages. The issues with using AES-128 have already been outlined in section 2.5, which was the need to use two ADS-B messages to transmit information required in one. The proposed embedded system could alleviate this issue by the use of an FPE algorithm.

Similar to the SIBE, in this hybrid framework, each ATC would have a public-private key pair, and the public keys would all be preloaded on all aircraft via the Flight Management System (FMS) as explained in [19]. When an aircraft enters an airspace managed by a certain ATC, the aircraft, more specifically the RP, generates a symmetric key and encrypts it using the ATC's public key. The encrypted session key is then transmitted to that ATC, which decrypts it using its private key. One viable channel for key transmission is the Aircraft Communications Addressing and Reporting System (ACARS), which is described in section 3.4.2. The session key, which is now shared with ATC, would then be used by the RP to encrypt ADS-B messages using an FPE algorithm as demonstrated earlier. The ATC, now in possession of the decrypted key, would be able to decrypt all ADS-B messages. This process can be repeated as the aircraft enters other airspaces managed by other ATCs until the aircraft lands.

The use of this hybrid asymmetric-symmetric encryption model addresses the challenge of KMD. By having the embedded system randomly generate a new session key each time it enters a new airspace, and by safeguarding all private keys, the framework mitigates the risks of key compromise.

3.4.2 Leveraging Aircraft Communication Systems

Another possible solution to the KMD challenge is the use of existing/new aircraft communication systems to securely distribute keys from aircraft to ATC.

Systems such as the Aircraft Communications Addressing and Reporting System (ACARS) and satellite internet may be viable solutions for the distribution of encryption keys.

ACARS is a data link protocol that provides communication between aircraft and ground stations [31]. The communications include sharing weather updates, flight plans, and maintenance reports. ACARS messages are sent over Very High Frequency (VHF) or via Satellite Communication (SATCOM), which are susceptible to interception and security breaches. However, ACARS over IP (AoIP) has been introduced as a more secure alternative [32]. AoIP transmits ACARS messages over packets, which are transmitted over broadband cellular or satellite networks with enhanced encryption and Virtual Private Network (VPN) capabilities. This makes AoIP a viable solution for the secure transmission of the encryption keys used by the on-board embedded system for the encryption of ADS-B messages.

Satellite internet is another viable solution for the secure transmission of encryption keys. A key player in satellite internet networks is Starlink, which operates thousands of satellites in low-earth orbit (550 km) [33]. This allows Starlink to achieve lower latency (25 ms) compared to other satellite internet operators, who generally operate their satellites in geostationary orbit (35,786 km). Starlink already provides/will provide internet services to several airlines, including United Airlines, Air France, Air New Zealand, and Qatar Airways [34]. An aircraft using the proposed embedded system to encrypt ADS-B messages could utilize Starlink's high-speed, low-latency internet connection to transmit the encryption key to ATC.

3.5 Summary

This chapter outlined the methodology used to evaluate the feasibility of implementing FPE on ADS-B messages. The system was designed to simulate real-world conditions as closely as possible, using an RP as the embedded computer and a virtual ATC environment for decryption and validation. The experimental workflow ensured a controlled evaluation of the three FPE algorithms while key performance metrics provided details into their computational impact on the RP. Despite the limitations outlined in section 3.1.4, the methodology demonstrates a practical approach to enhancing ADS-B integrity without altering its structure.

4 Results and Discussion

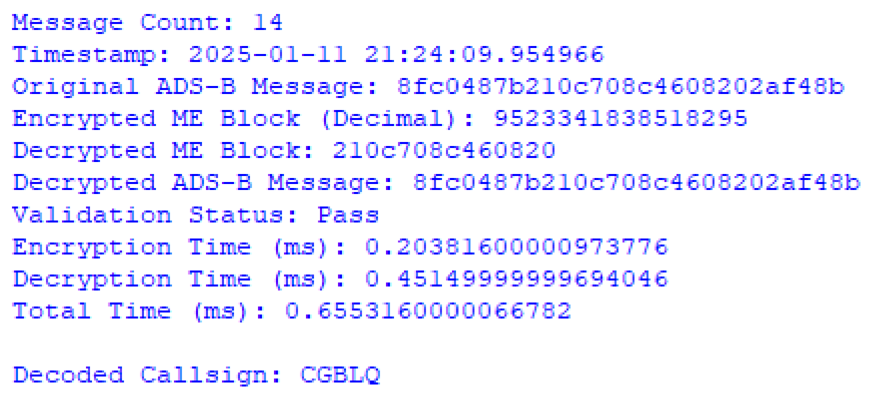
This chapter presents the performance analysis and security evaluation of the proposed solution for enhancing ADS-B integrity using FPE. It begins with an assessment of how well the encryption algorithms preserve the format of ADS-B messages and their ability to support accurate message decoding. The chapter then evaluates the effectiveness of the solution in protecting against eavesdropping, message injection, and message modification attacks. Performance metrics such as encryption and decryption times, CPU and memory usage, and thermal behaviour are analyzed to determine the computational feasibility of implementing FPE in an RP. Finally, a security analysis of the three FPE algorithms is provided, followed by a discussion on challenges related to real-world implementation in the aviation industry.

4.1 Performance Analysis of FPE Algorithms

4.1.1 Format Preservation and Decoding

All three FPE algorithms – FF3, FFX, and AES-CTR – were successful in preserving the length/format of the ME blocks throughout the encryption and decryption processes. This was confirmed by comparing the decrypted ADS-B message by ATC VM with the original unencrypted ADS-B message to see if they were identical. All 1,000 messages passed this validation check for all three FPE algorithms.

Additionally, the preserved format of the ADS-B message enabled the proper decoding of all ADS-B messages using the `pyModeS` Python library in [26], which only decodes correctly formatted ADS-B messages. Decoding ADS-B messages extracts aircraft parameters such as aircraft callsign, speed, and position. As shown in Figure 14, the encrypted ME block (in decimal to satisfy the FF3 library requirements) gets decrypted and converted back to a hexadecimal string. Once reconstructed with the rest of the ADS-B message, it is compared with the original ADS-B message, and the validation check passes if the two are identical. The sample ADS-B message shown in Figure 14 shows a decoded message containing the callsign CGBLQ. All 1,000 messages were successfully and correctly decoded after being decrypted using all three FPE algorithms.



```
Message Count: 14
Timestamp: 2025-01-11 21:24:09.954966
Original ADS-B Message: 8fc0487b210c708c4608202af48b
Encrypted ME Block (Decimal): 9523341838518295
Decrypted ME Block: 210c708c460820
Decrypted ADS-B Message: 8fc0487b210c708c4608202af48b
Validation Status: Pass
Encryption Time (ms): 0.20381600000973776
Decryption Time (ms): 0.45149999999694046
Total Time (ms): 0.6553160000066782

Decoded Callsign: CGBLQ
```

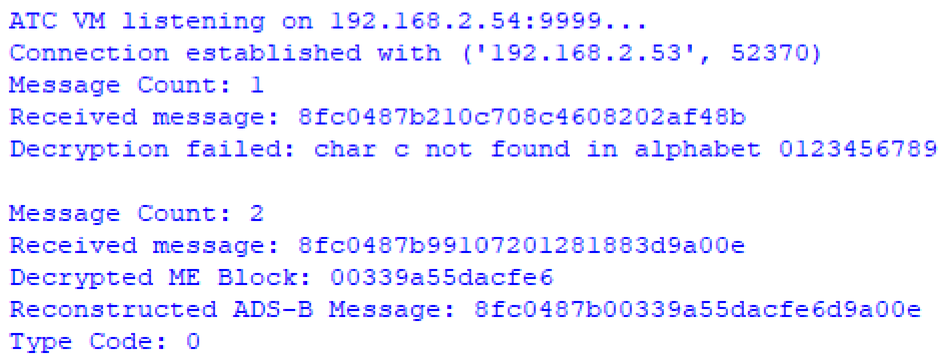
Figure 14: Sample output of FF3 decryption script in ATC VM

4.1.2 Validation of ADS-B Integrity Protection

To validate whether the proposed solution enhanced the integrity of ADS-B, the integrity of the proposed system was evaluated through targeted attack simulations. These simulations were designed to validate the system's ability to reject fabricated or tampered messages i.e. protection against message injection and modification attacks.

As explained in section 2.3.3.1, a message injection attack involves transmitting fabricated ADS-B messages to simulate non-existent aircraft, thereby compromising air traffic data [15]. To evaluate the ability of the proposed solution to protect against this type of attack, two ADS-B messages were transmitted to the ATC VM without encryption. The assumption here is that an attacker would not be aware of the encryption key. The two ADS-B messages used in this simulation were collected from the ADS-B log file of 1,000 messages. They were legitimate and correctly formatted ADS-B messages that would have been decoded correctly when using the `pyModeS` library. However, since the ATC VM was setup to first decrypt the ADS-B message before decoding it, the injection attack failed.

The results, shown in Figure 15, demonstrate that the decryption process failed for both injected messages. The first message resulted in an error because the received ME block contained a non-decimal character 'c', and the second message failed because its type code '0' is not a valid type code. Both errors were due to decrypting a non-encrypted ADS-B message.

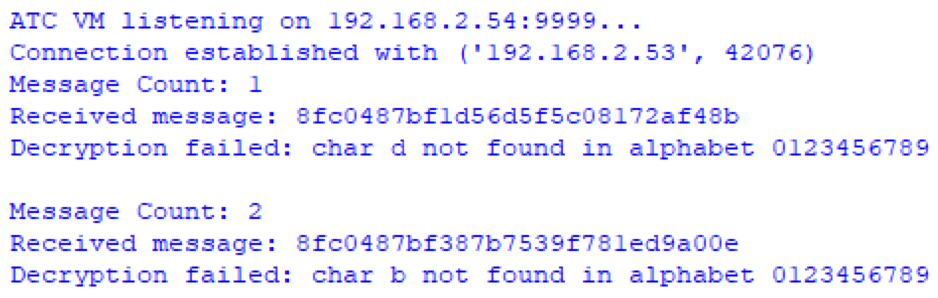


```
ATC VM listening on 192.168.2.54:9999...
Connection established with ('192.168.2.53', 52370)
Message Count: 1
Received message: 8fc0487b210c708c4608202af48b
Decryption failed: char c not found in alphabet 0123456789

Message Count: 2
Received message: 8fc0487b99107201281883d9a00e
Decrypted ME Block: 00339a55dacfe6
Reconstructed ADS-B Message: 8fc0487b00339a55dacfe6d9a00e
Type Code: 0
```

Figure 15: Output of message injection attack with proposed solution

Similarly, the proposed solution also successfully protected against message modification attacks, which modify messages from aircraft before they are received by ADS-B In [3]. This test was conducted by having the RP script encrypt an ADS-B message as designed, but then modify some characters in the encrypted ME block before transmitting the messages to the ATC VM, simulating a message modification attack. Once the message is received by the ATC VM, it attempts to decrypt and decode it. However, as shown in Figure 16, it fails to do so since the modification either introduced invalid characters or resulted in an improperly formatted ADS-B message for decoding. Additionally, even if the decryption and decoding process had succeeded, the modification would have resulted in a failed validation check which would have discarded the message. A sample output of the message modification simulation is shown in Figure 16.



```
ATC VM listening on 192.168.2.54:9999...
Connection established with ('192.168.2.53', 42076)
Message Count: 1
Received message: 8fc0487bf1d56d5f5c08172af48b
Decryption failed: char d not found in alphabet 0123456789

Message Count: 2
Received message: 8fc0487bf387b7539f781ed9a00e
Decryption failed: char b not found in alphabet 0123456789
```

Figure 16: Output of message modification attack with proposed solution

The results of the message injection and modification attacks show that the proposed solution is successful in enhancing the ADS-B integrity. Fabricated or modified messages are easily rejected by the proposed solution which could also alert the ATC to such errors, possibly indicating an attack. The applied FPE algorithms for ADS-B messages also have the added benefit of protecting against eavesdropping attacks, though this is not the main purpose of the proposed solution.

4.1.3 Encryption and Decryption Times

As discussed in section 3.3.1, encryption and decryption times are critical metrics to measure to assess the feasibility of the FPE algorithms for ADS-B systems. ADS-B messages are transmitted as often as every 0.5 seconds so any delay introduced by the encryption or decryption processes must not impact this transmission frequency.

The encryption and decryption times for FF3, FFX, and AES-CTR were evaluated over 1,000 ADS-B messages with an introduced inter-message delay of 0.5 seconds to simulate a real-world ADS-B system. After encrypting each ME block, the reconstructed ADS-B message, now with an encrypted ME block, was transmitted to the ATC VM for decryption. Results of the encryption times on the RP are shown in Figure 17 and the results of the decryption times on the ATC VM are shown in Figure 18. Additionally, Figure 19 shows the combined encryption and decryption times.

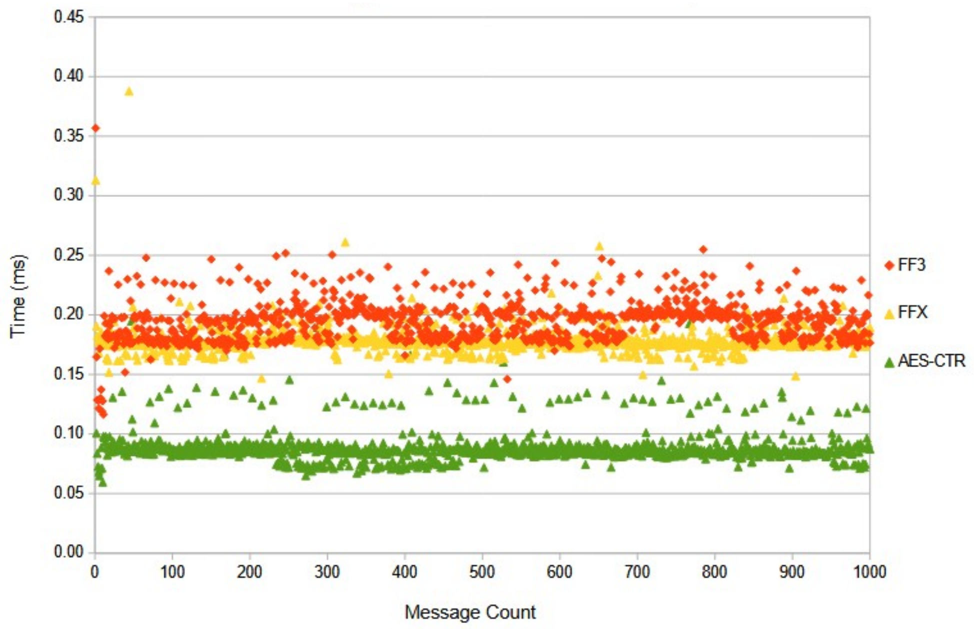


Figure 17: Encryption time with 0.5 s delay on the RP using all three FPE algorithms

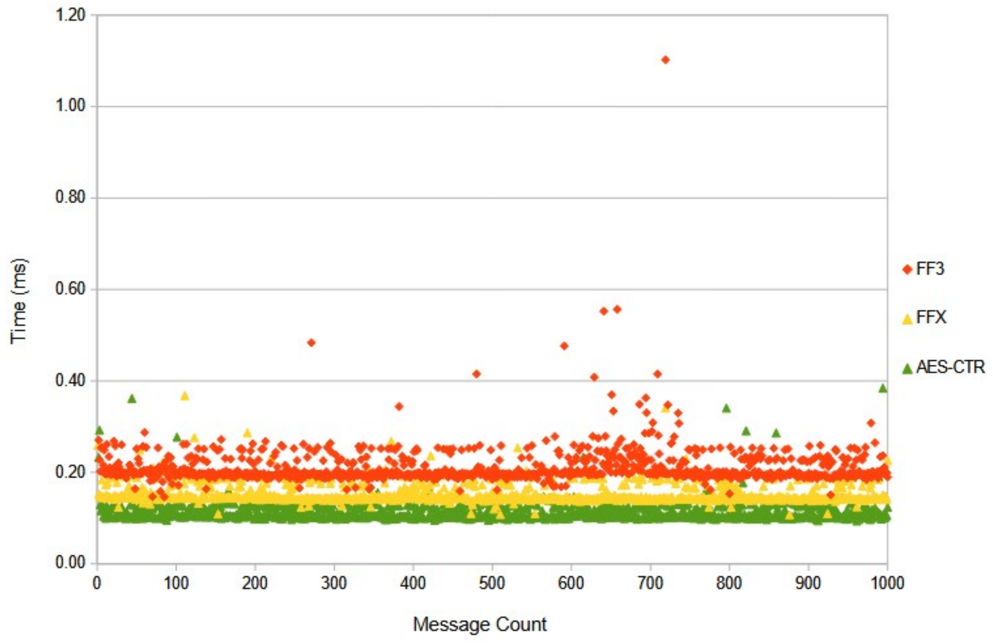


Figure 18: Decryption time with 0.5 s delay on the ATC VM using all three FPE algorithms

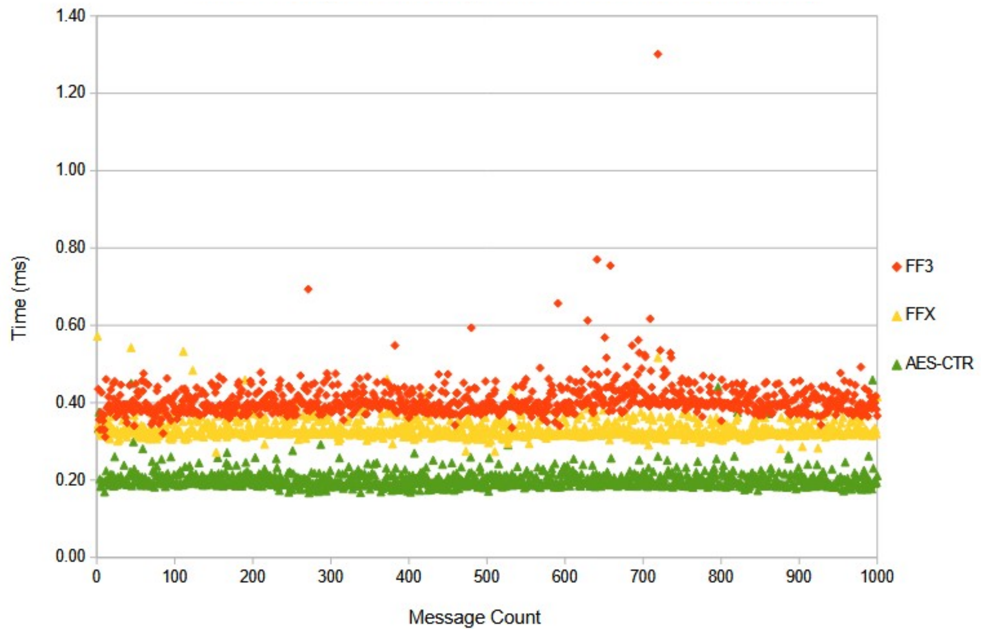


Figure 19: Combined encryption and decryption time with 0.5 s delay using all three FPE algorithms

Figure 17 shows that using AES-CTR resulted in the lowest encryption time with an average encryption time of 0.09 ms, while FFX resulted in an average of 0.18 ms and FF3 resulted in the highest encryption time with an average of 0.19 ms. Similar results can be seen for decryption times, where AES-CTR resulted in the lowest times with an average of 0.11 ms, followed by FFX with an average of 0.15 ms and finally FF3 with the highest decryption time with an average of 0.21 ms. Additionally, both FF3 and AES-CTR resulted in some variability where some encryption times were as high as 0.36 ms and 0.19 ms, respectively. The results are summarized in Table 5.

Table 5: Summary of encryption, decryption, and combined times for FF3, FFX, and AES-CTR

FPE Algorithm	**Average Encryption Time (ms)**	**Average Decryption Time (ms)**	**Average Combined Time (ms)**	**Max Combined Time (ms)**
FF3	0.19	0.21	0.41	1.30
FFX	0.17	0.15	0.33	0.57
AES-CTR	0.09	0.11	0.20	0.46

The standard deviation of encryption times was also calculated for all FPE algorithms and the results are shown in Figure 20. FFX resulted in the least standard deviation of 0.012 ms, while FF3 resulted in the highest standard deviation of 0.017 ms. However, the differences between all three are negligible and all resulted in very low encryption time variability given the highest transmission frequency of ADS-B of 2 Hz (0.5 seconds).

As shown in Table 2, there could be times where multiple ADS-B messages are transmitted at the same time. In the worst-case scenario, an aircraft may simultaneously transmit up to six different ADS-B messages: Aircraft Identification, Airborne Position, Airborne Velocity, Aircraft Status, Target States and Status, and Operational Status messages. Using the maximum combined time from Table 5 (achieved by FF3 at 1.30 ms), the worst-case scenario where six messages are transmitted within the same 500 ms period would be 7.8 ms. This would account for a 1.56% increase to the processing time. This scenario represents the absolute worst case as it not only assumes the highest recorded encryption and decryption time but also considers the worst-performing FPE algorithm (FF3) and aligns with the ADS-B Out system transmitting the maximum number of messages simultaneously. However, this scenario is uncommon and a more realistic scenario would result in an even lower encryption and decryption overhead, making the time introduced by the encryption and decryption processes negligible.

The results confirm that AES-CTR is the most efficient algorithm in terms of encryption and decryption times, making it the most suitable FPE algorithm for encrypting ADS-B messages. That said, both FF3 and FFX also did well in terms of encryption/decryption times. The results of the standard deviation of encryption times were also very similar for all three FPE algorithms. Therefore, any of the three tested FPE algorithms would be an acceptable solution for encrypting ADS-B messages.

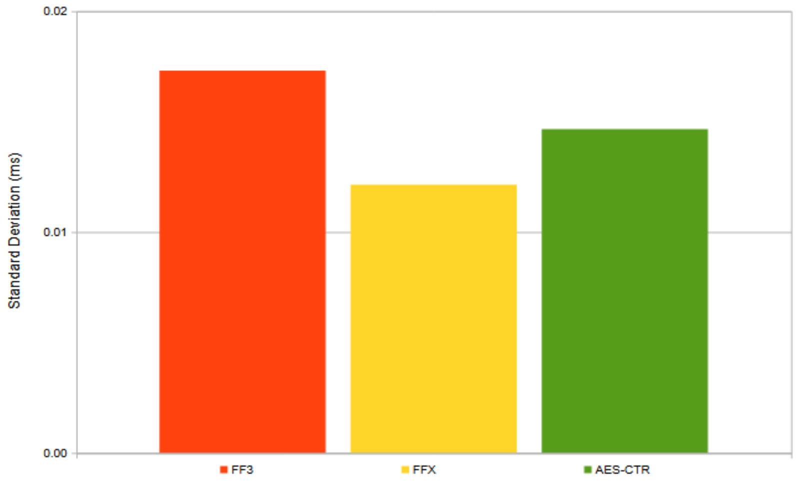


Figure 20: Standard deviation of encryption times for all three FPE algorithms

4.1.4 CPU and Memory Usage

Evaluating the CPU and memory usage is important for determining the computational demands of the FPE algorithms when implemented on the selected embedded system. The results of the CPU usage are shown in Figure 21. They show that all three FPE algorithms performed similarly with AES-CTR using the least CPU resources at 24.58% while FF3 and FFX used 24.96% and 25.0%, respectively. The results indicate that the encryption process runs on a single core while the RP used is a quad-core computer. This confirms that the encryption workload is primarily CPU-bound rather than memory-bound, but does not saturate the entire processor. The marginally lower CPU utilization of AES-CTR aligns with the observed encryption and decryption time results discussed in section 4.1.3. We therefore conclude that all three FPE algorithms are suitable for ADS-B message encryption using the selected embedded system. Figure 22 shows the memory usage results for all three FPE algorithms. Again, all algorithms used little memory making any of them suitable for encrypting ADS-B messages using the selected embedded system.

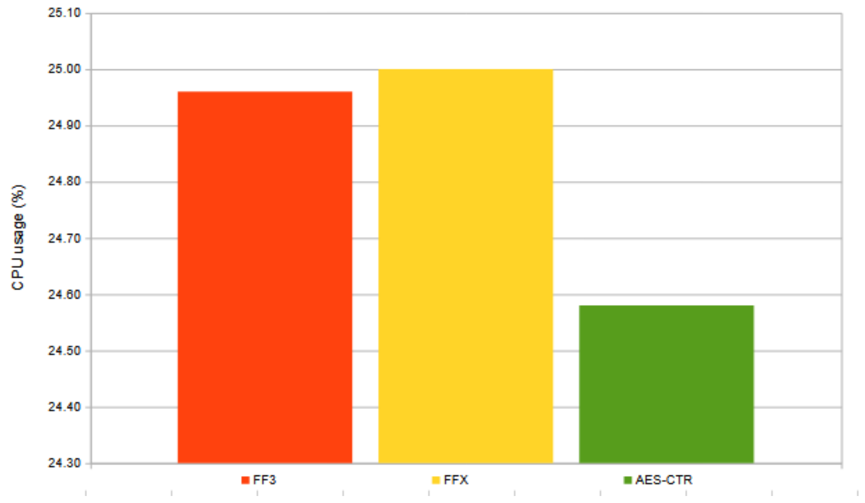


Figure 21: RP CPU usage of all three FPE algorithms

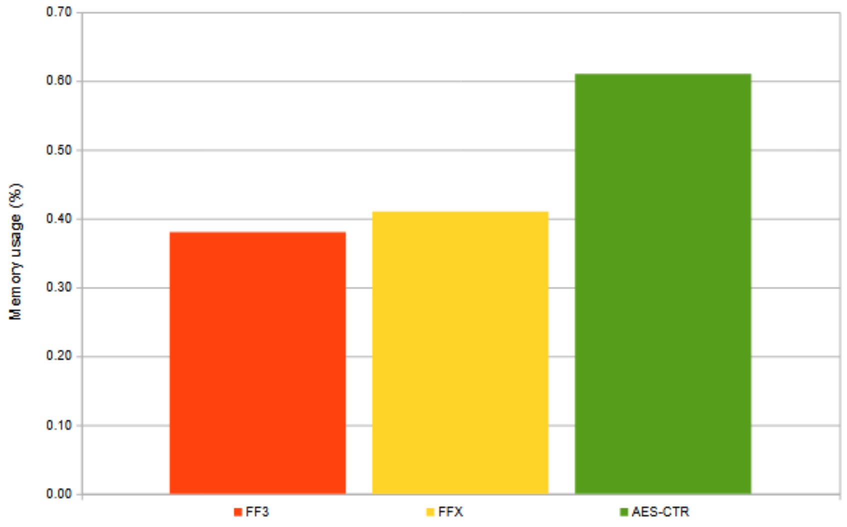


Figure 22: RP memory usage of all three FPE algorithms

4.1.5 Thermal Behaviour of RP

As explained in section 3.3.3, the temperature of the RP CPU was measured during a stress test where each FPE algorithm was used to encrypt 1,000,000 ADS-B messages after having the RP idle for 60 minutes to get a stable baseline temperature. Encrypting 1,000,000 messages only took approximately nine minutes for each FPE algorithm. As shown in Figure 23, the three FPE algorithms performed similarly, with AES-CTR resulting in the lowest maximum temperature of 81.8°C, followed by FFX and FF3 with a maximum CPU temperature of 85.1°C and 85.6°C, respectively.

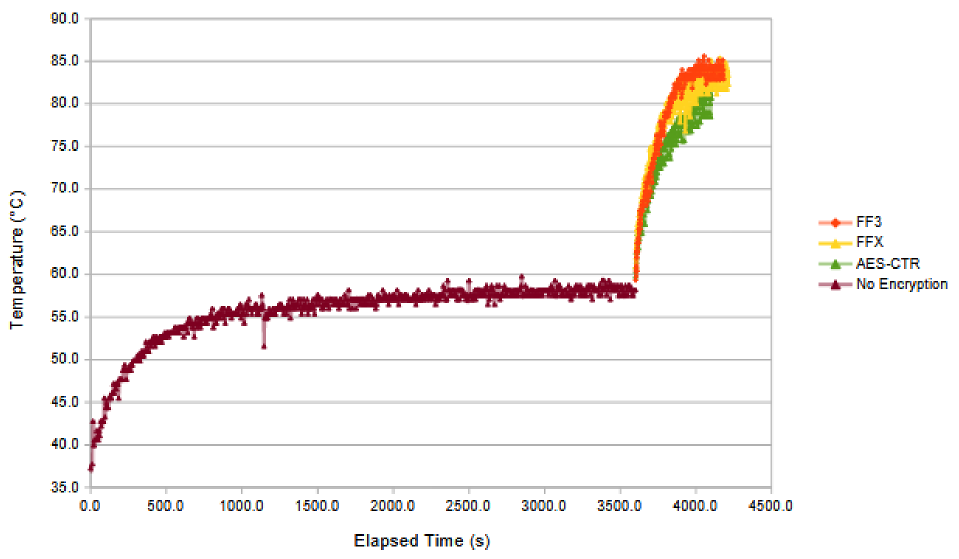


Figure 23: Thermal performance of RP during stress test for all three FPE algorithms

According to [35], the Arm Coretx A76 CPU on the RP begins thermal throttling at 80°C and reaches severe throttling at 85°C. This can affect the RP's performance by reducing the CPU speeds from 2.4 GHz to about 1.5 GHz, potentially impacting the RP's performance in encrypting and transmitting ADS-B messages. However, it is important to note that this test was designed to push the RP far beyond what it would realistically experience in an operational setting. In an actual deployment, the RP would encrypt and transmit ADS-B messages at a much lower rate, resulting in a significantly lower computation load and, therefore, lower peak temperatures. Consequently, the thermal test conducted above should not be interpreted as a representation of operational conditions. Instead, this test serves as an initial exploration of the RP’s temperature response under extreme loads and highlights the need for further study under more realistic conditions.

4.2 Security Analysis of FPE Algorithms

This section provides a qualitative security analysis of the three FPE algorithms used in this thesis: FF3, FFX, and AES-CTR. The analysis is based on existing literature rather than the conduct of specific experiments. The section discusses security characteristics, advantages, and potential vulnerabilities of these algorithms.

4.2.1 FF3

As discussed in sections 2.6.1 and 3.2.1, a 256-bit key was used along with a 64-bit tweak and eight Feistel rounds. According to Agbeyibor et al., FF3 builds on AES as its core block cipher, like other FPE algorithms [21]. Therefore, its cryptographic strength is closely tied to AES. The publication at [21] showed that all three NIST-recommended FPE algorithms (FF1, FF2, and FF3) demonstrate high ciphertext entropy and no major exploitable weaknesses.

One difference of FF3 when compared to other FPE algorithms is the use of an eight-round Feistel structure, which is fewer than the number of rounds used in FF1 and FF2. Consequently, Agbeyibor et al. saw that FF3 required the least hardware resources and offered faster encryption compared to FF1 and FF2. Despite its fewer rounds, FF3 also achieved comparable entropy to the other FPE algorithms.

4.2.2 FFX

As discussed in sections 2.6.2 and 3.2.2, a 256-bit key was used with 10 Feistel rounds and without any tweak. Given these parameters, the FFX implementation mimics the behaviour of FF1 which also uses 10 Feistel rounds and no tweak. Therefore, the findings in [21] on FF1 are relevant to the FFX implementation in this thesis. In theory, using 10 rounds in FF1 should offer more resilience against attacks than eight rounds, though the results in [21] showed that the entropy when using FF1 and FF3 were almost identical. However, the results also showed that FF1 is the slowest of the three NIST-recommended algorithms, while still respecting the timing requirements of the ADS-B protocol.

4.2.3 AES-CTR

As explained in section 2.6.3, AES-CTR is not a true FPE algorithm but it does preserve the format for smaller data blocks, such as the ME block which is 56 bits. Ketata et al. highlighted how AES-CTR can be efficiently parallelized in software underscoring its high throughput and relatively low computational overhead when used on multi-core systems [36]. However, the literature also discusses several considerations when using AES-CTR.

One limitation of AES-CTR is its inability to provide authenticity or integrity checks on its own [37]. This is because the CTR mode transforms AES into a stream cipher, which means that any malicious modifications of the ciphertext can be undetected unless the implementation is combined with a separate MAC or digital signature. However, as was discussed in section 4.1.2, any modification to the encrypted ADS-B message should result in an error when ATC tries to decrypt and decode the message. Hence, ADS-B integrity is still protected when using AES-CTR.

Another limitation of AES-CTR is reusing the same nonce and counter combination under the same encryption key [37]. Reusing the same combination could allow an attacker to detect repeating keystream segments and recover plaintext data. This highlights the importance of robust KMD as well as nonce/tweak and key generation to mitigate this vulnerability.

4.3 Implementation Challenges

While the proposed embedded system solution enhances the integrity of ADS-B by encrypting ADS-B messages using an FPE algorithm, there are still some challenges in implementing this solution in a real-world aviation environment. These challenges include regulatory, financial, and operational considerations that will be discussed in this section.

4.3.1 Regulatory and Certification Challenges

The aviation industry is one of the most regulated industries worldwide, with national and international aviation authorities such as Transport Canada, the FAA, and ICAO enforcing certification requirements [38]. Therefore, proposing the installation of new hardware and software requires comprehensive design reviews, documentation, and formal testing. These factors, in addition to the delays associated with this process, might discourage the airline industry from upgrading the ADS-B system, especially when protecting its integrity is not mandated.

4.3.2 Financial and Operational Constraints

The airline industry might also be discouraged by this proposed solution due to the added financial and operational burdens. Even though the hardware proposed is of relatively low cost, it does not account for the cost of the required research, development, acquisitions, installation, and aircraft downtime. Additionally, the installed hardware and software will require maintenance staff training and changes to maintenance procedures and documentation. Upgrades to ATC and training for ATC personnel would also be required.

From an operational perspective, the industry must also consider a mechanism to deal with embedded system failure to encrypt messages, such as reverting back to transmitting unencrypted ADS-B messages. In such a case, the aircraft system would need to fail safe and notify ATC to also discontinue the decryption process (with the use of a flag in the ADS-B message for example). A similar safeguard mechanism must exist if the decryption hardware or software fails, ensuring that air traffic management is not compromised. These mechanisms must occur automatically with minimal human intervention to have the least impact on air traffic management.

Despite these regulatory, financial, and operational challenges, the proposed solution demonstrates that FPE can be implemented efficiently on a low-cost embedded computer such as the RP without altering the structure of ADS-B messages. This makes the proposed solution viable for enhancing ADS-B integrity.

5 Conclusion

5.1 Recommendations for Future Work

This thesis focused specifically on the ME Extraction and Encryption Module of the in-line device. The complete in-line device described in section 3.1.1 includes additional components such as the ADC, Reassembler, Checksum Calculator, and DAC which were not part of the experimental setup. As a result, the potential processing time added by these components was not accounted for in the evaluation. Future research should focus on integrating and testing these components to assess the feasibility of a fully operational system.

The results shown in this thesis were based on experiments conducted in a laboratory environment using an RP and an ATC VM simulating an ATC receiver. Future studies could conduct real-world trials involving live ADS-B transmissions on an actual aircraft, or replicate the setup in a laboratory using aircraft and ATC equipment. Such trials would mimic a real-world application and introduce environmental factors such as radio frequency interference and distance limitations which would provide more accurate results on how the encryption system behaves under real operational conditions.

While the thermal performance of the RP was evaluated under extreme conditions, a more detailed analysis is needed to fully characterize its thermal performance under realistic operating conditions. Future work should investigate the RP's temperature response under expected ADS-B transmission rates in a controlled environment. This is especially important when considering that the RP would be fitted onboard an aircraft, likely in the avionics bay, where the ambient temperature would be higher than room temperature due to all the other avionics emitting heat. Understanding the RP's thermal characteristics in such an environment is essential for assessing its long-term stability for real-world ADS-B encryption.

The thesis focused on three FPE algorithms: FF3, FFX, and AES-CTR. Future studies could examine other FPE variants, such as FF1 or FF2, or develop a novel asymmetric FPE algorithm. An asymmetric FPE algorithm would solve the KMD challenge by relying on public/private keys instead of a shared encryption key while preserving the format and length of ADS-B messages as to not require major alterations to the current ADS-B framework.

5.2 Summary of Findings

This research demonstrates that enhancing ADS-B integrity through FPE can be accomplished with minimal disruption to existing aviation infrastructure. By placing an embedded system, such as the proposed RP, between the ADS-B equipment in the aircraft and the ADS-B antenna, the ME blocks of ADS-B messages were successfully encrypted without altering their format or length. On the receiving end, ATC was also successful in decrypting and decoding the encrypted ADS-B messages. Three encryption algorithms – FF3, FFX, and AES-CTR – were implemented and evaluated for encryption/decryption time, standard deviation of encryption times, CPU and memory usage, and thermal behaviour on the selected embedded system.

All tested FPE algorithms successfully preserved the structure of ADS-B messages, ensuring that existing decoding software (such as the `pyModeS` library) could decode the decrypted data. Additionally, the proposed solution was successful in enhancing ADS-B integrity by simulating message injection and modification attacks. In these attacks, non-encrypted messages that were injected and modified messages failed to decrypt or validate correctly. The proposed solution also protects against eavesdropping due to the added encryption. Table 6 demonstrates the attacks mitigated by the proposed FPE solution while maintaining a relatively low-medium implementation difficulty compared to other cryptographic solutions.

The performance metrics measured using the selected embedded system included encryption and decryption times, CPU and memory usage, and thermal behaviour of the RP. For encryption and decryption times, all three algorithms resulted in minimal latency, with AES-CTR having the least encryption and decryption times, followed closely by FFX and FF3. Even in the worst case, the maximum combined encryption and decryption time was 1.3 ms, which is negligible compared to the highest ADS-B message transmission rate of every 0.5 seconds. The RP's CPU and memory usage was within acceptable ranges for all three algorithms. As for the thermal behaviour of the RP, it never failed to encrypt messages during the stress test of encrypting 1,000,000 messages.

Table 6: Summary of defensive techniques including proposed FPE solution for ADS-B vulnerabilities and implementation difficulty

	Timestamp Authentication	Multilateration	Multichannel Receiver	AES	SIBE	Proposed FPE
Eavesdropping				✓	✓	✓
Jamming			✓			
Message Injection	✓	✓		✓	✓	✓
Message Deletion		✓				
Message Modification		✓		✓	✓	✓
Implementation Difficulty	Low	Medium	Medium	Medium-High	High	Low-Medium

From a security evaluation perspective, FF3, FFX, and AES-CTR all demonstrated strong cryptographic properties. The aim of this thesis was to propose a solution that enhances the integrity of ADS-B messages using FPE while respecting the structure and transmission constraints of the ADS-B protocol and ensuring it can be implemented on an embedded computer. The results of this work confirm that this aim was met. The proposed encryption system successfully protected ADS-B messages against injections and mortification attacks while maintaining compatibility with existing decoding tools. Additionally, the performance evaluation demonstrated that the selected embedded system could handle the encryption workload without introducing significant latency, making this solution practical for real-world deployment.

References

[1] Transport Canada, *AC 700-009: Automatic Dependent Surveillance - Broadcast (ADS-B) Operational and Maintenance Considerations*, Advisory Circular, Jul. 02, 2021. Accessed: Feb. 13, 2024. [Online]. Available: https://tc.canada.ca/sites/default/files/2021-07/AC_700-009_ISSUE_03.pdf

[2] H. Ali and S. P. Leblanc, 'Vulnerabilities of Automatic Dependent Surveillance-Broadcast on Aircraft: Survey', in *2024 International Conference on Computing, Internet of Things and Microwave Systems (ICCIMS)*, Gatineau, QC, Canada: IEEE, Jul. 2024, pp. 1–5. doi: 10.1109/ICCIMS61672.2024.10690823.

[3] NAV Canada, *Optimizing the Benefits of Space-based ADS-B*. Accessed: Dec. 15, 2024. [Online]. Available: https://www.navcanada.ca/en/air-traffic/space-based-ads-b.aspx#Benefits

[4] NAV Canada, *ADS-B Performance Requirements*. Accessed: Jan. 18, 2024. [Online]. Available: https://www.navcanada.ca/en/air-traffic/space-based-ads-b/ads-b-performance-requirements.aspx

[5] FAA, *Automatic Dependent Surveillance-Broadcast (ADS-B)*, Jan. 17, 2023. Accessed: Mar. 03, 2024. [Online]. Available: https://www.faa.gov/air_traffic/technology/equipadsb

[6] Transport Canada, *AC 500-029: Certification of Automatic Dependent Surveillance – Broadcast (ADS-B)*, Advisory Circular, Jan. 04, 2024. Accessed: Feb. 14, 2024. [Online]. Available: https://tc.canada.ca/en/aviation/reference-centre/advisory-circulars/advisory-circular-ac-no-500-029

[7] J. Sun, *The 1090 Megahertz Riddle - A Guide to Decoding Mode S and ADS-B Signals*, 2nd ed. TU Delft OPEN Publishing, 2021. Accessed: Feb. 14, 2024. [Online]. Available: https://mode-s.org/decode/book-the_1090mhz_riddle-junzi_sun.pdf

[8] Q. Zhang, Z. Wang, B. Wu, and G. Gui, 'A Robust and Practical Solution to ADS-B Security Against Denial-of-Service Attacks', *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13647–13659, Apr. 2024, doi: 10.1109/JIOT.2023.3337543.

[9] M. Leonardi, E. Piracci, and G. Galati, 'ADS-B jamming mitigation: a solution based on a multichannel receiver', *IEEE Aerosp. Electron. Syst. Mag.*, vol. 32, no. 11, pp. 44–51, Nov. 2017, doi: 10.1109/MAES.2017.160276.

[10] Federal Aviation Administration, *91.225 Automatic Dependent Surveillance-Broadcast (ADS-B) Out equipment and use.*, 14 CFR 91.225, May 28, 2010. Accessed: Dec. 25, 2024. [Online]. Available: https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-91/subpart-C/section-91.225

[11] European Union Aviation Safety Agency, *COMMISSION IMPLEMENTING REGULATION (EU) 2020/587*, Regulation (EU) 2020/587, Apr. 29, 2020. Accessed: Dec. 25, 2024. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32020R0587

[12] International Civil Aviation Organization, 'ADS-B Implementation and Operations Guidance Document'. International Civil Aviation Organization, Sep. 2015. Accessed: Dec. 25, 2024. [Online]. Available: https://www.icao.int/APAC/Documents/edocs/cns/AIGD%20Edition%2015.0.pdf

[13] P. C. van Oorschot, *Computer Security and the Internet Tools and Jewels from Malware to Bitcoin*, 2nd ed. Springer Nature, 2021.

[14] M. Leonardi, E. Piracci, and G. Galati, 'ADS-B vulnerability to low cost jammers: Risk assessment and possible solutions', in *2014 Tyrrhenian International Workshop on Digital Communications - Enhanced Surveillance of Aircraft and Vehicles (TIWDC/ESAV)*, Rome, Italy: IEEE, Sep. 2014, pp. 41–46. doi: 10.1109/TIWDC-ESAV.2014.6945445.

[15] M. Schäfer, V. Lenders, and I. Martinovic, 'Experimental Analysis of Attacks on Next Generation Air Traffic Communication', in *Applied Cryptography and Network Security*, vol. 7954, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., in Lecture Notes in Computer Science, vol. 7954. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 253–271. doi: 10.1007/978-3-642-38980-1_16.

[16] Y. Kim, J.-Y. Jo, and S. Lee, 'ADS-B vulnerabilities and a security solution with a timestamp', *IEEE Aerosp. Electron. Syst. Mag.*, vol. 32, no. 11, pp. 52–61, Nov. 2017, doi: 10.1109/MAES.2018.160234.

[17] N. Constantinescu, E. Constantinescu, and A.-I. Chira, 'Enhancing the performance of the Primary Surveillance Radar using Multilateration', *INCAS BULLETIN*, vol. 14, no. 4, pp. 35–49, Dec. 2022, doi: 10.13111/2066-8201.2022.14.4.4.

[18] A. Smith, R. Cassell, T. Breen, R. Hulstrom, and C. Evers, 'Methods to Provide System-Wide ADS-B Back-Up, Validation and Security', in *2006 ieee/aiaa 25TH Digital Avionics Systems Conference*, Portland, OR, USA: IEEE, Oct. 2006, pp. 1–7. doi: 10.1109/DASC.2006.313681.

[19] J. Baek, E. Hableel, Y.-J. Byon, D. S. Wong, K. Jang, and H. Yeo, ‘How to Protect ADS-B: Confidentiality Framework and Efficient Realization Based on Staged Identity-Based Encryption’, *IEEE Trans. Intell. Transport. Syst.*, vol. 18, no. 3, pp. 690–700, Mar. 2017, doi: 10.1109/TITS.2016.2586301.

[20] M. Li, Z. Liu, J. Li, and C. Jia, ‘Format-Preserving Encryption for Character Data’, *JNW*, vol. 7, no. 8, pp. 1239–1244, Aug. 2012, doi: 10.4304/jnw.7.8.1239-1244.

[21] R. Agbeyibor, J. Butts, M. Grimaila, and R. Mills, ‘Evaluation of Format- Preserving Encryption Algorithms for Critical Infrastructure Protection’, in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, vol. 8827, E. Bayro-Corrochano and E. Hancock, Eds., in Lecture Notes in Computer Science, vol. 8827. , Cham: Springer International Publishing, 2014, pp. 245–261. doi: 10.1007/978-3-662-45355-1_16.

[22] M. Dworkin, ‘Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption’, National Institute of Standards and Technology, NIST SP 800-38G, Mar. 2016. doi: 10.6028/NIST.SP.800-38G.

[23] M. Bellare, P. Rogaway, and T. Spies, ‘The FFX Mode of Operation for Format-Preserving Encryption’, National Institute of Standards and Technology, Draft 1.1, Feb. 2010. Accessed: Jan. 04, 2025. [Online]. Available: https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/proposed-modes/ffx/ffx-spec.pdf

[24] R. Housley, ‘Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)’, Vigil Security, Jan. 2004. Accessed: Jan. 05, 2025. [Online]. Available: https://www.ietf.org/rfc/rfc3686.txt#:~:text=AES%2DCTR%20has%20many%20properties,to%20generate%20the%20key%20stream.

[25] B. Schoening, B. Mares, P. Banerjee, and J. Robichaud-Gagnon, ‘FPE - Format Preserving Encryption with FF3 in Python’. GitHub, Mar. 13, 2024. Accessed: Nov. 16, 2024. [Online]. Available: https://github.com/mysto/python-fpe?tab=readme-ov-file

[26] J. Sun, X. Olive, and X. Hirsch, ‘The Python ADS-B/Mode-S Decoder’. GitHub, Aug. 25, 2024. Accessed: Nov. 20, 2024. [Online]. Available: https://github.com/junzis/pyModeS

[27] J. Dollinger, ‘pyffx’. GitHub, May 12, 2019. Accessed: Nov. 20, 2024. [Online]. Available: https://github.com/emulbreh/pyffx?tab=readme-ov-file

[28] Python Cryptographic Authority, ‘Welcome to pyca/cryptography’. Oct. 18, 2024. Accessed: Nov. 22, 2024. [Online]. Available: https://cryptography.io/en/latest/

[29] Linux manual pages, 'getrusage — get resource usage'. Linux manual pages. Accessed: Jan. 09, 2025. [Online]. Available: https://www.linuxcampus.net/documentation/man-html/htmlman2/getrusage.2.html

[30] G. Rodola, 'psutil Documentation'. Sep. 07, 2017. Accessed: Jan. 08, 2025. [Online]. Available: https://readthedocs.org/projects/giamptest/downloads/pdf/latest/

[31] International Communications Group, 'Introduction to ACARS Messaging Services'. International Communications Group, Apr. 17, 2006. Accessed: Jan. 10, 2025. [Online]. Available: https://www.icao.int/safety/acp/inactive%20working%20groups%20library/acp-wg-m-iridium-7/ird-swg07-wp08%20-%20acars%20app%20note.pdf

[32] Collins Aerospace, 'Complemenetary Solution to Traditional ACARS'. 2022. Accessed: Jan. 10, 2025. [Online]. Available: collinsaerospace.com/acars-over-ip

[33] Starlink, 'Satellite Technology'. Accessed: Jan. 10, 2025. [Online]. Available: www.starlink.com/ca/technology

[34] M. Goldstein, 'United Airlines Expedites Move To High-Speed Starlink Connectivity', Forbes. Accessed: Jan. 10, 2025. [Online]. Available: https://www.forbes.com/sites/michaelgoldstein/2025/01/10/united-airlines-speeds-up-move-to-high-speed-starlink-connectivity/

[35] M. Toback, '10 Key Performance Benchmarks of Raspberry Pi 5', MiniPC Tech. Accessed: Jan. 13, 2025. [Online]. Available: https://minipctech.com/performance-benchmarks-of-raspberry-pi-5/

[36] R. Ketata, L. Kriaa, L. A. Saidane, and G. Chalhoub, 'Detailed Analysis of the AES CTR Mode Parallel Execution Using OpenMP', in *2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*, Paris, France: IEEE, Nov. 2016, pp. 1–9. doi: 10.1109/PEMWN.2016.7842901.

[37] H. Lipmaa, P. Rogaway, and D. Wagner, 'Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption'. National Institute of Standards and Technology, Oct. 2000. Accessed: Jan. 15, 2025. [Online]. Available: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/ctr/ctr-spec.pdf

[38] W. Voss, 'Harsh Truth', Flight safety Foundation. Accessed: Jan. 14, 2025. [Online]. Available: https://flightsafety.org/asw-article/harsh-truth/