2021

Automatic Dependent Surveillance Broadcast (ADS-B) Security Mitigation through Multilateration

Skylar Stroman
n00925130@unf.edu

Automatic Dependent Surveillance Broadcast (ADS-B) Security Mitigation through Multilateration

A Thesis submitted to the Department of Electrical Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering
UNIVERSITY OF NORTH FLORIDA

December 2021

This thesis **Automatic Dependent Surveillance Broadcast (ADS-B) Security Mitigation through Multilateration**, submitted by **Skylar Stroman** in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering has been:

Approved by thesis committee: **Date:**

_____ 12/17/2021
Associate Professor of Electrical Engineering
Committee Chair

_____ 12/17/2021
O. Patrick Kreidl, Ph.D.
Associate Professor of Electrical Engineering
Committee Member

_____ 12/17/2021
John Nuszkowski, Ph.D.
Associate Professor of Mechanical Engineering
Committee Member

_____ 12/17/2021
Brian Kopp, Ph.D., P.E.
Associate Professor of Electrical Engineering
External – Committee Member

DEDICATION

I would like to dedicate my work to my wife Shannon.

Her continuous love, support and encouragement made this all possible.

ACKNOWLEDGEMENTS

I would like to thank Dr Patrick Kreidl, Dr Alan Harris, Dr Brian Kopp and Dr John Nuszkowski for their advisement and feedback throughout this process. Their advice allowed for a clear and concise scope of work, as well as a manageable timeline.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Automatic Dependent Surveillance Broadcast (ADS-B) was mandated January 1st, 2020 to all commercial aircraft that fly over 10,000 ft [1]. This radio frequency (RF) based technology is part of an international plan to phase out radar-based surveillance within the airspace. Unfortunately, due to a lack of encryption and other security measures, ADS-B transmission remains open for possible exploitation. This thesis will explore the use of Multilateration (MLAT) to validate location data provided within the ADS-B framework. MLAT uses the Time Difference of Arrival (TDOA) at multiple locations to determine the origin of a received signal. Additionally, as MLAT greatly depends on the topology of receiver location, multiple configurations will be examined and simulated within the confines of a real-world application. Finally, adversary spoofing scenarios were explored by injecting a stationary and moving adversary into the system. The adversary transmitted ADS-B location data from a different origin than the packets would indicate to create a fake aircraft in the airspace. The performance of the MLAT model was analyzed to determine its ability to flag the adversary's data as suspicious.

INTRODUCTION

Commercial aircraft in service today have a wide array of technology on board for safety, performance and customer experience. Inside the cockpit, avionics equipment is generally segmented into 3 main groups, Communication, Navigation and Surveillance (CNS) [2]. These technologies provide the main pillars of safety for all modern commercial aviation.

The communication group of equipment contain radio type products. Most modern aircraft have multiple radios that transmit over different frequencies, with different types of information [3]. Very-High Frequency (VHF) radio is considered the main radio for voice communication in a commercial aircraft while traveling over land [3], while High Frequency (HF) radios are generally utilized when traveling over the ocean where access to base stations are not available [4].

The navigation group of avionics focuses on guiding the aircraft to the correct destination. In a modern aircraft, this is achieved with the use of Global Positioning Satellites (GPS) that determine the location of an aircraft through a datalink with at least 3 satellites [4]. This information is used by a wide array of other equipment on the aircraft for various reasons.

Finally, avionics in the surveillance group are all focused around locating and tracking aircraft. This may be performed by other aircraft or by the Air Traffic Control (ATC) systems on the ground. Historically, ATC surveillance was achieved using radar technology that would bounce radio waves off aircraft and measure the time it took for the signal to return. This would allow the ground to physically track aircraft when in an airspace [5]. Radar technology is currently being phased out based on the cost to maintain the aged equipment as well as new means to track aircraft.

The major technology being adopted into the commercial aviation sector to replace radar surveillance is Automatic Dependent Surveillance Broadcast (ADS-B). This broadcast became a mandate on January 1, 2020 for all aircraft flying above 10,000 feet. ADS-B equipped aircraft continuously broadcast specific information including, but not limited to, identification, position, altitude and velocity [6]. This information can be used by ATC systems as well as other aircraft.

The continuous broadcast of ADS-B by all commercial aircraft allows for increased safety by giving pilots a better situational awareness regarding air traffic conditions, as well as, increasing the detection likelihood of any air-traffic collision. ADS-B also allows for better airspace surveillance by expanding the range and accuracy when compared to the legacy radar in use today [7]. With so many benefits, it is clear why ADS-B is one of the core fundamentals of the global aviation roadmap [8].

Unfortunately, during the creation of the ADS-B standard, little to no attention was given to data security [9]. This has introduced a large amount of risk into any environment where ADS-B is the main source of aircraft surveillance. This limitation is widely known in the aviation community and yet no major mitigation effort is currently being performed.

With a lack of security, ADS-B information is susceptible to multiple situations an adversary could find tempting. Eavesdropping on the information could be completed simply by a tuned antenna and a receiver [9]. The raw data could then be decoded based on the published ADS-B standard. This would provide the information to anyone who may be interested in tracking certain aircraft.

Spoofing is another way an attacker may use the ADS-B information. This may be done by injecting non-legitimate messages into the frequency band or altering legitimate messages and relaying them [9]. This would provide false data to the ADS-B communication bus and force alternative means of validation to ATC and other aircraft.

Jamming is a very common mechanism used in existing air combat tactics where a signal is broadcast over the desired frequency with enough power to block or corrupt other signals [10]. Jamming of ADS-B could occur since the frequency band of the transmission is public knowledge.

Finally, a Denial of Service (DoS) situation could occur if the communication system is flooded with a large number of false signals [9]. This would greatly limit the ability of the ATC or other aircraft to identify legitimate aircraft. This may also cause additional issues since many of the avionic systems in the modern aircraft have automatic corrective actions build into their control systems. If a fake "ghost" aircraft is detected by an aircraft with the possibility of a collision, automatic correction may go into effect, causing the aircraft to change position, altitude, heading and/or speed [3].

The utilization of ADS-B as a successor to other surveillance systems is a key objective for the aviation industry moving forward. Unfortunately, a thorough security analysis effort was not completed during the development of the standard and a wide variety of associated risks are now present. Fortunately, there are alternative ways that can be used to validate the legitimacy of an ADS-B broadcast.

The use of Multilateration (MLAT) will be investigated as a possible candidate to provide a high confidence level within the received ADS-B location data and attempt to flag any signals that may be suspicious.

The thesis is organized by first providing basic background information on relevant topics surrounding the research. Then, any technical information that is required for comprehension of the results and conclusions will be provided. Additionally, a simulation overview will be provided, as well as the results and drawn conclusion. Finally, suggested future work will be discussed to provide recommendations to others that are interested in the subject matter.

BACKGROUND

Automatic Dependent Surveillance Broadcast (ADS-B)

Prior to the introduction of ADS-B, other technologies were used to provide real time location data to ATC systems. A legacy technique still in use today is radar-based surveillance. The technology behind radar is fundamentally quite simple. A radio wave is transmitted in a certain direction until it is reflected to the source by any object, as is shown in Figure 1. Based on the time between the transmitted and received signals, the distance can be calculated. This, along with the strength of the return signal, can provide a general size estimate of the object [10].

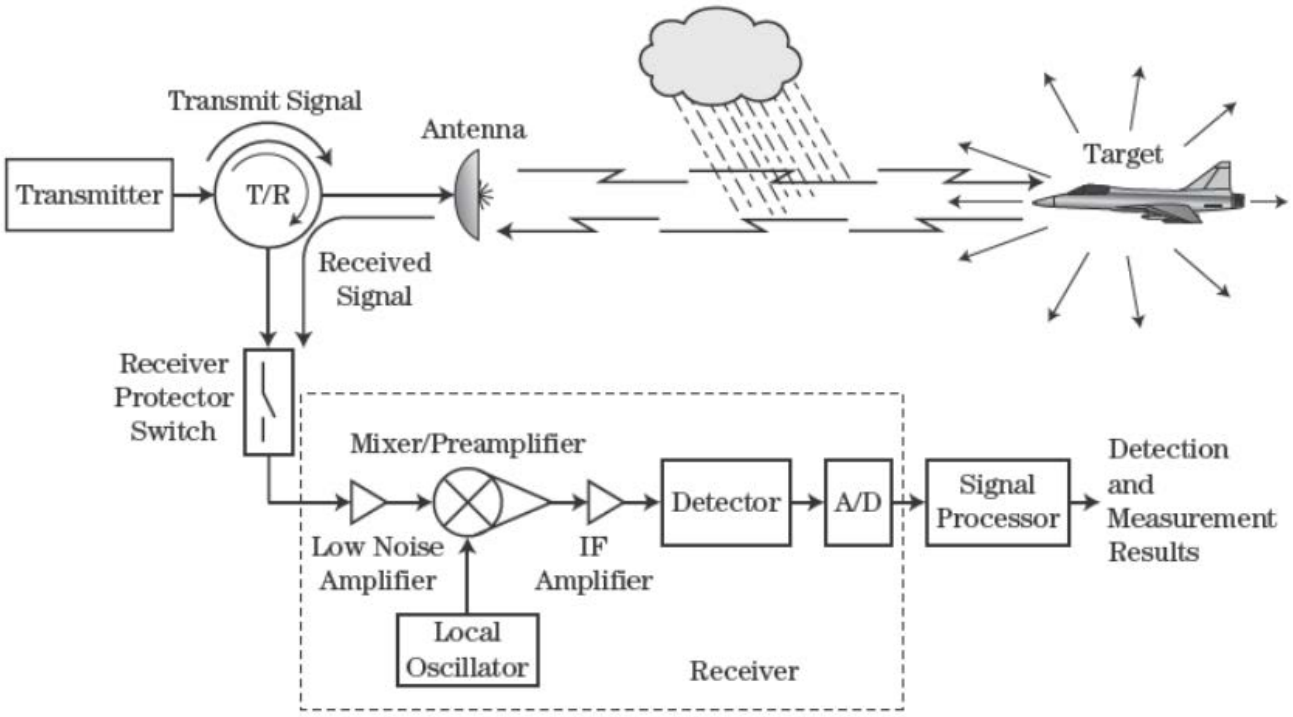

Figure 1: Main Elements of Radar Tx / Rx Process [10]

When compared to ADS-B, radar surveillance technology in use at most airports has a smaller range of detection due to the large amount of signal power loss associated with a reflected radar signal. As objects move farther and farther away from the original transmission of the radio wave, they become more difficult to individually distinguish. Additionally, the refresh rate of radar information is relatively slow (between 4-12 seconds) based on the physical rotation of the radar tower as it scans all 360 degrees of the airspace [10]. These are inherent technology limitations with radar and can only be mitigated using more elaborate systems with multiple transmission and receiving points [10]. When these radar systems were originally introduced, the amount of air traffic was significantly less than it is today, so these limitations were not considered to be a problem. As air traffic has increased, the limitations of the existing system have become more apparent and given rise to additional surveillance techniques.

In addition to the physical tracking of aircraft with radar beam reflection, information was added to the radio wave that was used to reflect off the aircraft [11]. This wave, transmitted at 1030 MHz, interfaces with the Transponder (Transmitter/Responder) equipment onboard a commercial aircraft. Based on the message query that is carried on the radar signal, the transponder will transmit an applicable response. This information is transmitted over 1090 MHz so as not to have any data collisions between transmission and receiving [12]. Mode-A and Mode-B messages are used to query the aircraft's identification. A Mode-C message provides the aircraft altitude based on the pressure altimeter equipment on board. Finally, a Mode-S message can provide a wide array of information including, but not limited to, all Mode-A/B/C data, flight status, selected altitude, roll angle, ground speed, magnetic heading and vertical climb/decent rate [12]. Mode-S information is a key corollary in the ADS-B mandate as a subset of this information is included in the data that is automatically transmitted.

The use of ADS-B in commercial aircraft is a mandate that went into effect in 2020. ADS-B has been selected by the US based NEXTGEN program and the European Union SESAR program, as one of the key technologies to implement [8]. This mandate has forced nearly all commercial aircraft to retrofit or update their transponder systems to allow for ADS-B.

ADS-B can be broken down into two different categories, ADS-B Out and ADS-B In. ADS-B Out is part of the mandate that began in 2020, while ADS-B In is currently considered optional. ADS-B Out functions by transmitting specific data packets over the specified carrier frequency. Two different frequencies are used today, 1090 MHz and 978 MHz, with 1090 MHz being a carryover from Mode-S transponder and 978 MHz acting as a new dedicated frequency for ADS-B [1]. ADS-B Out that is broadcast at 978 MHz is called Universal Access Transceiver (UAT) and can only be used on aircraft that operate between 10,000 and 18,000 feet [1]. Anything that has a cruising altitude above 18,000 feet must use the 1090 MHz frequency, which is called Extended Squitter (ES) [1].

ADS-B Out provides automatic broadcasting of an aircraft's identification, position, altitude and velocity per the RTCA DO-260B standard [13]. The identification of the aircraft is determined when it has been registered with the International Civil Aviation Organization (ICAO). This value is encoded into 24 bits of information that is used in the ADS-B data packets [6]. The position and velocity of the aircraft is determined using a Global Positioning System (GPS) receiver. This receiver uses the Global Navigation Satellite System (GNSS) to provide a position. The aircraft's altitude is determined using the on-board barometric pressure altimeter to provide the most accurate and consistent altitude [1]. These data points are packaged into 112-bit packets that are then broadcast over the 1090ES or 978 UAT datalink [1]. Data packets are consistently formatted, with only some variance based on the type of information that is being provided. The generic 112-bit packet is illustrated in Figure 2 and Table 1 [6]. The focus of this research will be centered around the airborne position message as it has the most impact on the flow of air traffic.

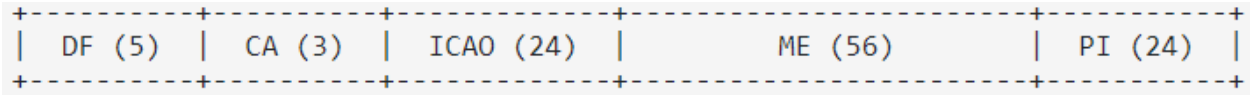

Figure 2: ADS-B Data Packet [6]

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	56	ME	Message, extended squitter
(33–37)	(5)	(TC)	(Type code)
89–112	24	PI	Parity/Interrogator ID

Table 1: Structure of ADS-B Data Packet [6]

Bits 33-88 are modified based on the type of message being sent, with the first 5 bits indicating the type of message being sent (Type Code). When sending airborne position data within the message frame, the data in Figure 3 and Table 2 illustrate the specific content in the data packet.

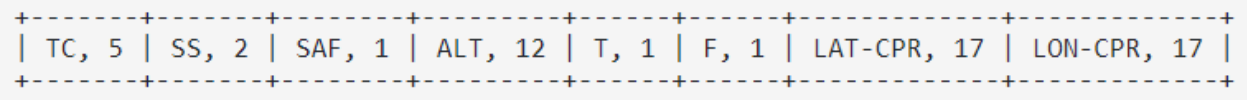

Figure 3: Airborne Position Message Frame [6]

FIELD		MSG	ME	BITS
Type Code	TC	33–37	1–5	5
9–18: with barometric altitude				
20–22: with GNSS altitude				
Surveillance status	SS	38–39	6–7	2
0: No condition				
1: Permanent alert				
2: Temporary alert				
3: SPI condition				
Single antenna flag	SAF	40	8	1
Encoded altitude	ALT	41–52	9–20	12
Time	T	53	21	1
CPR Format	F	54	22	1
0: even frame				
1: odd frame				
Encoded latitude	LAT-CPR	55–71	23–39	17
Encoded longitude	LON-CPR	72–88	40–56	17

Table 2: Airborne Position Frame Data [6]

It should be noted that a data packet corresponding to the airborne position only provides half of the information necessary, so two packet transmissions are required to determine the true aircraft location. This is completed using a Compact Position Reporting (CPR) encoding format which requires fewer bits to transmit, yet a comparatively complicated arithmetic to decode [6].

Data Validation Techniques

Many different techniques could be utilized to validate the airborne location data that is provided within the ADS-B framework including those currently in use within the aviation community and outside of it. Wireless communication technologies provide a lot of promising options that have already been applied to cellular, WiFi and radio networks. An investigation was completed to determine which of the available techniques would be best for this application based on computation complexity, cost of required equipment and estimated accuracy.

ADS-B with Timestamping

One technique being proposed is to add timestamping information onto the base ADS-B data packet format. This timestamp would be generated by the transponder and tie to the GPS clock signal. From there, the ATC would be able to perform some computation to determine the location of the signal based on the time difference of arrival (TDOA). This, coupled with other TDOA calculations from other locations, could provide an accurate estimation of the signal origin [14].

The main challenge with this technique is to alter the base formatting of the ADS-B data packet. Since the formatting has already been determined through the global aviation community, the likelihood of altering this format is extremely low.

Key Hashing Cryptography

Cryptography is a common form of additional security in use across multiple industries. Specifically, Key Hashing involves transforming a set of data into another set of values by performing bit-wise operations. Once the data is transformed, only those with knowledge of the original hash value can reverse the process.

When applied to ADS-B transmission, one technique being researched is for the transmitter to compute a hash for a bundled group of messages. From there, the bundle is split into smaller ADS-B packets and transmitted. The

Cyclic Redundancy Check (CRC) portion of the base ADS-B data packet is modified to contain a portion of the hash value. When the signal is received, the hash is value is reassembled and the data converted back to its original raw values [15].

The main drawbacks with this technique are the alteration of the base ADS-B packet format and the use of hash keys. As stated in the previous section, alteration of the ADS-B packet format is fraught with red-tape based on the amount of administrative oversight. The use of hash keys brings its own complications when looking to implementation; specifically, the management of hash key creation, change frequency, ownership and unlawful distribution each pose their own challenges and risks.

Distance Bounding

Location verification on the physical layer is being investigated with one technique centered around a challenge-response framework. In this scenario, an ATC would query the aircraft directly and measure the round trip time. Based on the known speed of the signal, a distance could be determined [16].

The technique is like the fundamentals of radar but relies on a known delay factor in the aircraft equipment that is generating the response. If this delay factor is not extremely accurate, then the arithmetic to convert the time delay into a distance would be greatly affected. Furthermore, additional equipment or functionality would have to be created to implement this in the real-world.

Multilateration

The focus of this thesis will be the use of Multilateration due to is reuse of existing infrastructure and it being a solely arithmetic task to implement. Multilateration uses the Time Difference of Arrival (TDOA) at multiple locations to estimate the origin location of a received signal. When specifically implemented within the confines of airborne position validation, only three locations are required, based on the 2-dimensional nature of the data packet [17].

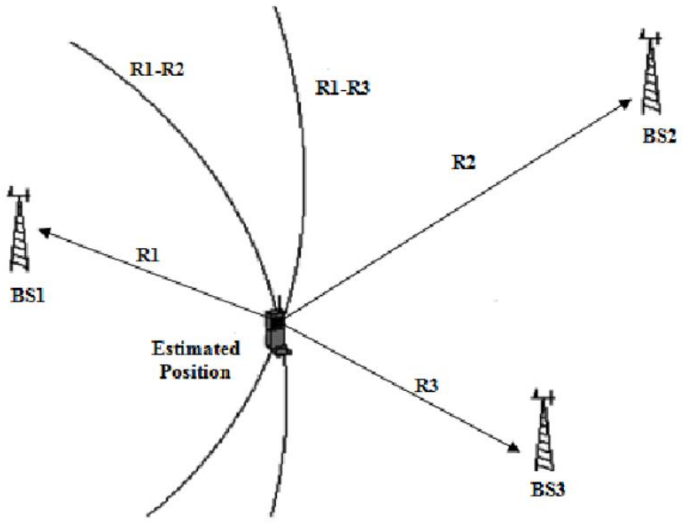

Figure 4: Simplified TDOA Setup [18]

A MLAT environment is dependent on the measurement of specific timestamps at different locations. Because of this, it is vital that the clock signals used are from the same source and with a high degree of accuracy. The majority of ATC systems utilize the GPS clock signal that is governmentally controlled to within nanoseconds of accuracy [19]. Once timestamps are gathered, arithmetic can convert the time to a range, based on the known signal speed being the speed of light.

Figure 4 illustrates a 3-receiver setup with the range of each tower from the signal origin shown, as well as the hyperbolic plot of the range differences.

Many techniques have been developed to determine an estimated location based on MLAT functionality with this thesis focusing on a 2-dimensional estimation provided by Chan [17]. Described in Equation 1, the Chan algorithm uses the difference in x coordinates, y coordinates and overall range of the three towers to determine the signal origin. Figure 5, shows the relationship between the tower x-y coordinates and distance (range) from the signal origin.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_{2,1} & y_{2,1} \\ x_{3,1} & y_{3,1} \end{bmatrix}^{-1} * \left\{ \begin{bmatrix} r_{2,1} \\ r_{3,1} \end{bmatrix} r_1 + \frac{1}{2} \begin{bmatrix} {r_{2,1}}^2 - K_2 + K_1 \\ {r_{3,1}}^2 - K_3 + K_1 \end{bmatrix} \right\} \quad (1)$$

With:

$\begin{bmatrix} x \\ y \end{bmatrix}$ = the estimated 2-D location of the signal

$x_{i,1}$ = the distance between tower i and tower 1 in the x coordinate

$y_{i,1}$ = the distance between tower i and tower 1 in the y coordinate

$r_{i,1}$ = the range between tower i and tower 1

${r_i}^2 = (x_i - x)^2 + (y_i - y)^2$

$K_i = {x_i}^2 + {y_i}^2$

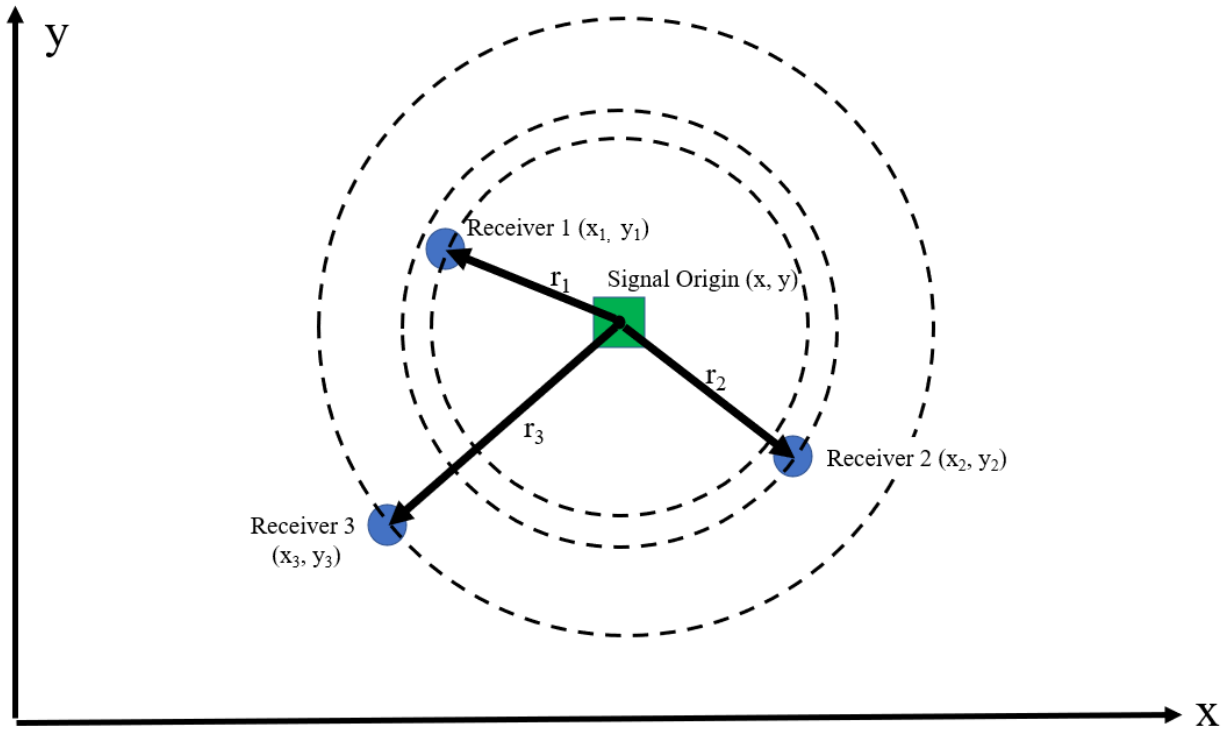

Figure 5: Multilateration Overview

CONTENT

Overview

To determine if MLAT could be used for validation of ADS-B signals, a simulation environment was setup in a MATLAB® software package. Real world conditions were modeled as much as possible to achieve results that were grounded in reality, including GPS location and clock synchronization noise. The simulation environment was created to model a baseline situation that tracked an incoming aircraft on a published airport approach path, as well as an adversary spoofing situation. Orlando International Airport (MCO) was chosen, as it was close to the authors residence. Additionally, a published, yet inactive, approach path was chosen (PIGLT FOUR) to tie in a real-world landing scenario. Finally, only two dimensions of the location data was simulated. Discussed more on page 22, the addition of the third dimension (altitude) would greatly increase the complexity of the MLAT algorithm with subjectively little added value.

The simulation modeled a theoretical aircraft as it followed the approach path and broadcasted its location over ADS-B. This data was simulated to determine the baseline accuracy of the MLAT modeling. Once a baseline was determined, an adversary was introduced into the simulation.

The adversary scenario was configured to model a spoofing situation. Specifically, the adversary would be broadcasting ADS-B data packets to create a "ghost" aircraft, along the PIGLT FOUR approach vector. The system model would take in the broadcasted location and compare it to the location provided by the MLAT calculation. If the variance between the ADS-B location and the calculated location was greater than a baseline threshold, the data point would be flagged as suspicious.

This configuration was simulated with 4 different ADS-B receiver tower topologies. The towers in scenarios 1 – 3 formed equilateral triangles and were strictly theoretical, while scenario 4 located the three towers and major airports across the state of Florida.

ADS-B Receiver Towers

The FAA does not publicly document where ADS-B Receiver Towers exist for means of national security. Based on this limitation, theoretical locations were chosen across the state of Florida. Much of the simulation focused on a triangular topology with MCO at its center of mass. In the last scenario, a more real-world orientation was explored, based on the performance of the previous simulation iterations, that placed the towers at Jacksonville International Airport, Tampa International Airport and Miami International Airport. This allowed for a plausible real-world topology to be compared.

It was also assumed that each of the towers was equipped with a GNSS uplink to take advantage of the GPS clock signal. This signal provides an extremely high level of accuracy and is vital to provide adequate timestamping.

GPS Location Reporting

ADS-B location relies on the GPS to provide real time data to report an aircraft's location. Based on the government standard, provided by the FAA, less than 2 meters of accuracy must be provided 95% of the time. This signal noise level was sequenced within the simulation under the assumption of a Gaussian probability distribution. To implement a random Gaussian noise into the MATLAB environment, the standard deviation (δ) and mean (μ) was required. Assuming a mean of zero, Equation 2, below, was used to determine the standard deviation of the noise. In this situation, a = -2 meters and b = 2 meters, as shown in Figure 6.

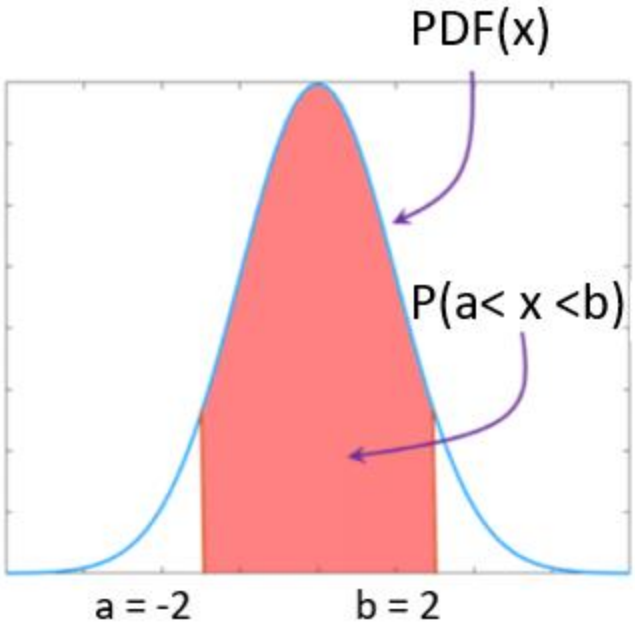

Figure 6: Probability Density Function of GPS Location Noise

$$P(a \leq X \leq b) = 1 - Q\left(\frac{\mu - a}{\delta}\right) - Q\left(\frac{b - \mu}{\delta}\right) \quad (2)$$

$$0.95 = 1 - Q\left(\frac{0 - (-2)}{\delta}\right) - Q\left(\frac{2 - 0}{\delta}\right)$$

$$0.95 = 1 - Q\left(\frac{2}{\delta}\right) - Q\left(\frac{2}{\delta}\right)$$

$$2Q\left(\frac{2}{\delta}\right) = 0.05$$

$$Q\left(\frac{2}{\delta}\right) = 0.025$$

$Q(2.805) \approx 0.025$ (Calculated through Q-tables in APPENDIX A: Q-Function Lookup Table)

$$\delta = \frac{2}{2.805} = 0.713\ meters$$

Using the calculated standard deviation (δ) and assumed mean, the additive noise of the GPS location data could be added into the simulation environment for a more realistic model.

GPS Clock Synchronization

For the MLAT calculations to be performed correctly, specifically the timestamping portion, each of the towers must use the same clock signal. A GPS based clock was used for this simulation, as well as the documented accuracy level. This allowed for real-world noise to enter the simulation to provide a more accurate representation of the application. Based on the published standard, a GPS clock is accurate to within less than 30 ns, 95% of the time [19]. Assuming a Gaussian noise distribution, Equation 2 was used to determine the standard deviation (δ) with a zero mean (μ). In this situation a = -30 ns and b = 30 ns, as shown in Figure 7.

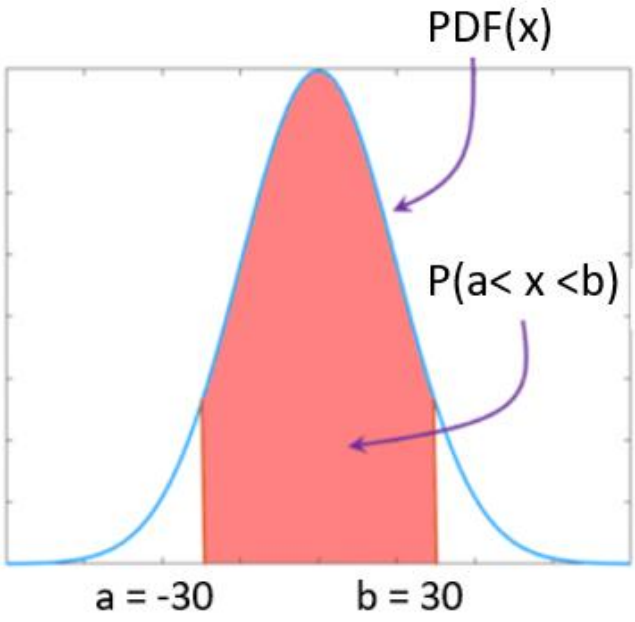

Figure 7: Probability Density Function of GPS Clock Noise

$$P(a \leq X \leq b) = 1 - Q\left(\frac{\mu - a}{\delta}\right) - \left(\frac{b - \mu}{\delta}\right) \quad (2)$$

$$0.95 = 1 - Q\left(\frac{0-(-30)}{\delta}\right) - Q\left(\frac{30-0}{\delta}\right)$$

$$0.95 = 1 - Q\left(\frac{30}{\delta}\right) - Q\left(\frac{30}{\delta}\right)$$

$$2Q\left(\frac{30}{\delta}\right) = 0.05$$

$$Q\left(\frac{30}{\delta}\right) = 0.025$$

$$Q(2.805) \approx 0.025$$

$$\delta = \frac{30}{2.805} = 10.695\ ns$$

This information was then utilized within the MATLAB software to add GPS clock noise into the timestamp creation when the ADS-B packets were received.

It should be noted that many internal latencies (equipment, transmission, calculation, etc) could be added into the simulation model to further ground the system in the real world. Within this simulation, no other latencies or signal noise was taken into account except the GPS location and GPS clock synchronization.

MLAT Calculations

To determine the theoretical 2-D location of a signal, three sets of information is required, including the X-Y coordinates of each tower and the corresponding timestamp of the received signal. Based on the Equation 1, by Chan [17], the location of the signal was derived.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_{2,1} & y_{2,1} \\ x_{3,1} & y_{3,1} \end{bmatrix}^{-1} * \left\{ \begin{bmatrix} r_{2,1} \\ r_{3,1} \end{bmatrix} r_1 + \frac{1}{2} \begin{bmatrix} {r_{2,1}}^2 - K_2 + K_1 \\ {r_{3,1}}^2 - K_3 + K_1 \end{bmatrix} \right\} \quad (1)$$

With:

$\begin{bmatrix} x \\ y \end{bmatrix}$ = the estimated location of the signal

$x_{i,1}$ = the distance between tower i and tower 1 in the x coordinate

$y_{i,1}$ = the distance between tower i and tower 1 in the y coordinate

$r_{i,1}$ = the range between tower i and tower 1

${r_i}^2 = (x_i - x)^2 + (y_i - y)^2$

$K_i = {x_i}^2 + {y_i}^2$

Only two dimensions were chosen to be investigated within this thesis based on the limited effect the altitude of an aircraft would have on the overall MLAT calculation, the significant complexity increase in the required 3-D algorithm and the addition of a fourth receiver tower within the system. For instance, at the start of the modeled approach path studied in this thesis, an aircraft's altitude should be close to 17,000 ft yet is over 500,000 ft away from its destination. This equates to a ~97% weighting on the latitude and longitude (x, y) coordinates versus a ~3% weighting on the altitude (z) value. Additionally, when trying to implement the arithmetic to perform a 3-D MLAT calculation, the complexity of the calculation increases because of the multiple intersections the 3 hyperbolic equations produce [17]. A deterministic model is then required to define which of the intersections is most likely correct, as the location is rarely co-located. An example can be seen in Figure 8 illustrating the most common scenario with 3 possible intersections. Finally, the physical implementation of the simulation would also require an additional tower to be added into the system. Based on these factors and the subjectively limited benefit in determining the altitude of the ADS-B signal origin, only a 2-dimensional simulation was investigated.

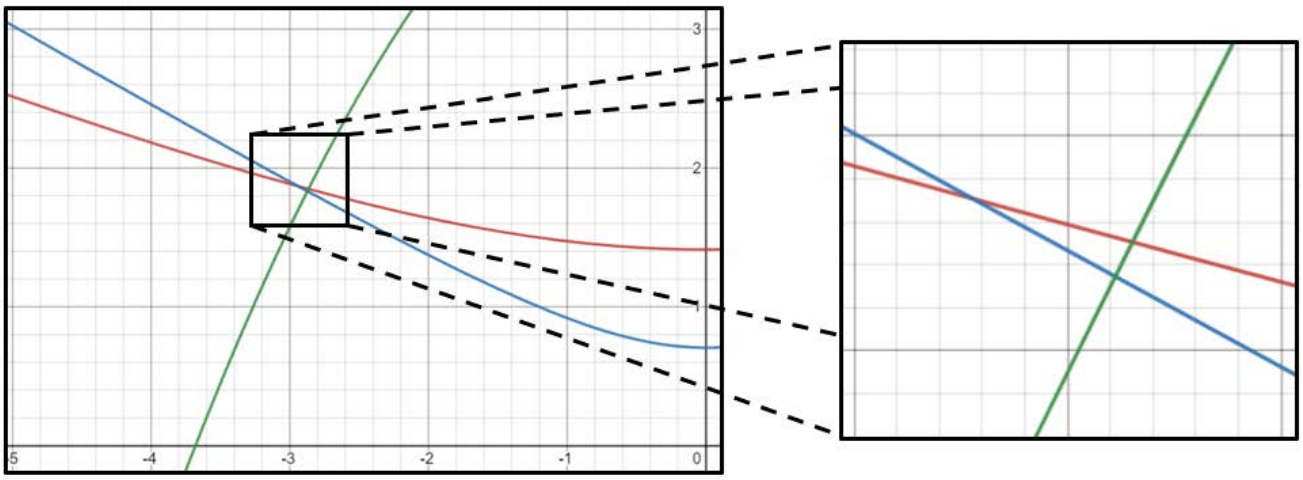

Figure 8: Intersections of 3 Hyperbolic Equations

Approach Path

To ground the simulation in reality, an actual approach path to MCO was chosen as the baseline and adversary vector. PIGLT FOUR, shown in Figure 9, was one of many approach paths that were published in 2016 by the FAA. This vector is now obsolete but is an accurate representation of the type of movement a commercial aircraft takes during its landing. Figure 9, shows the various segments of the approach path by indicating the minimum altitude (ft), nautical heading (degrees) and length (nautical miles).

Additionally, the approach speed was varied based on the portion of the approach. As an aircraft gets closer to landing, the speed is adjusted, based on the recommendations of ATC [20]. This information was utilized within the simulation to help define the data stream of the two ADS-B location packets (2 Hz).

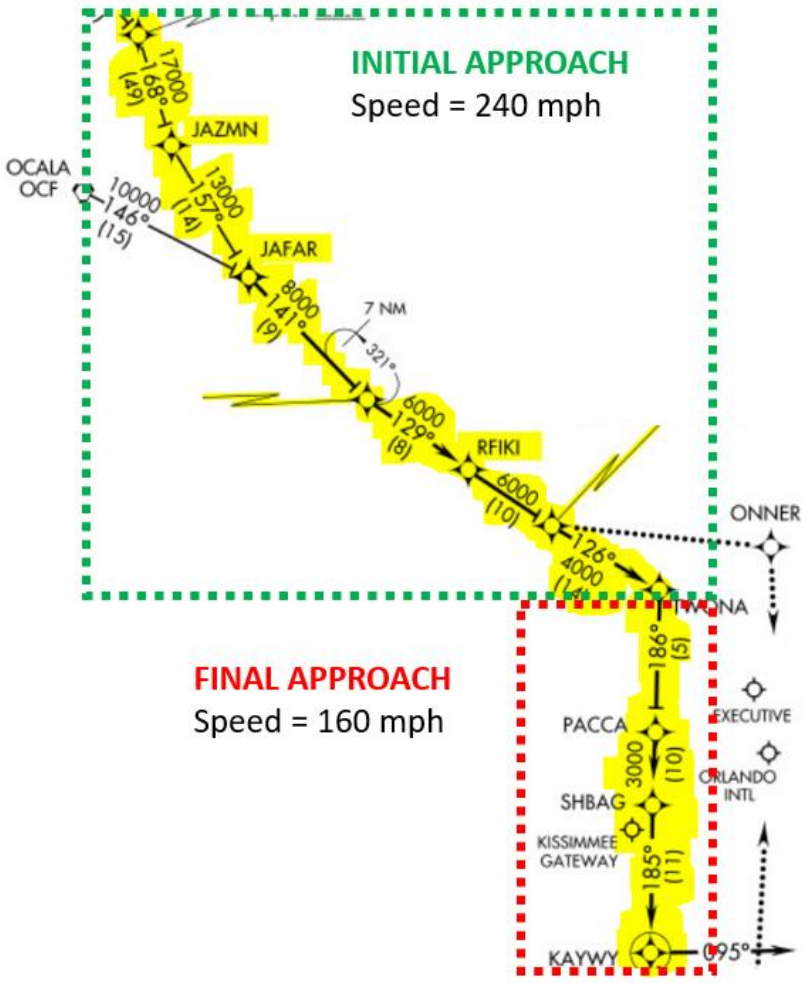

Figure 9: PIGLT FOUR Approach Path [21]

Flagging Threshold

For the adversary injection, a threshold needed to be determined to allow ATC to "Flag" a location data packet as suspicious. This determination would be a comparison between the reported ADS-B location and the MLAT calculated location. If the difference between the two data sets, at a specific timestamp, is greater than the threshold value, then it would indicate that a signal was being spoofed.

To determine the flagging threshold that would be implemented during the adversary injection portion of simulation an investigation into the baseline performance was conducted. Each scenario was iterated 10,000 times to determine

the mean measurement error and standard deviation of the data points. Assuming a baseline performance of 99% (<=1% of false flagging) and confirmed Gaussian probability distribution, the flagging threshold was determined using the Q-function.

$$0.99 = 1 - Q(\frac{b}{\delta}) \qquad (4)$$

Once this value was determined, it was used on the adversary injection to determine how a stationary and moving adversary would affect the performance of the model for each tower topology.

MODEL SIMULATION

Overview

Baseline Performance

A determination was made to calculate the flagging threshold of the baseline approach path with a specific receiver tower topology. Once the threshold was determined, the value was used to determine if a received ADS-B location data packet from an adversary was spoofed. Both a stationary and moving adversary was injected into the model and the performance was observed. Additionally, with each scenario, the topology of the receiver towers changed to determine if there was any performance change. A basic flow chart of the simulation model can be seen in Figure 10.

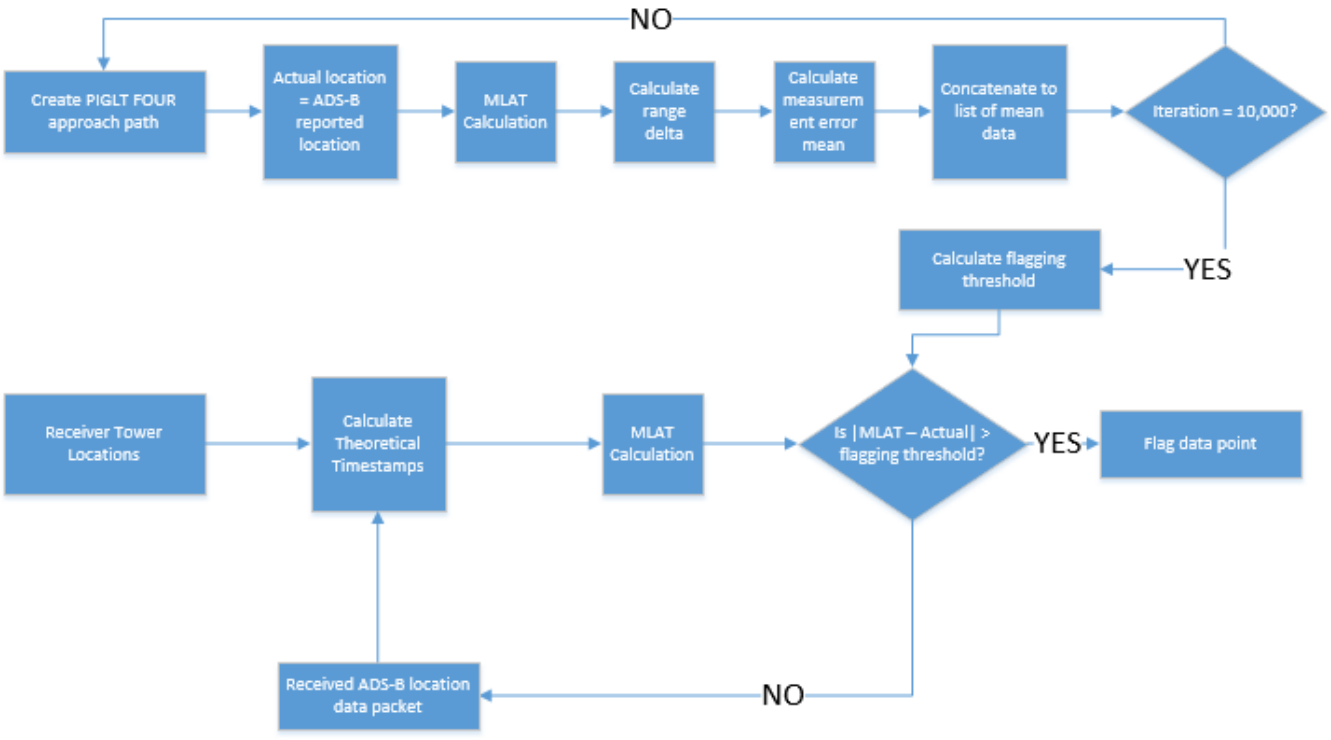

Figure 10: Simulation Flow Chart

As seen in Figure 10 and referenced in APPENDIX B-1: MATLAB Simulation Code - PIGLT FOUR Approach Path Creation, the simulation creates the PIGLT FOUR approach path to use as the "actual" location data. From there, the ADS-B location data is created by adding the GPS location noise to the actual location data. Next, the MLAT algorithm, powered by Equation 1 and referenced in APPENDIX B-2: MATLAB Simulation Code – TDOA Arithmetic, is used to determine the calculated location of each data point, based on the calculated noisy timestamps. Then, the measurement error value at each point is calculated by taking the root-mean-square between the MLAT location and the ADS-B broadcasted location data. The average measurement error is then calculated for the entire approach path.

This entire sequence is iterated 10,000 times, referenced in APPENDIX B-3: MATLAB Simulation Code – Baseline Performance, to obtain a large sample size of information and determine a baseline measurement error mean and standard deviation. The mean and standard deviation are then used to calculate the flagging threshold based on a 99% performance value (< 1% of false positive) for the baseline conditions. As in, if a known real aircraft followed the PIGLT FOUR approach path, then 99% of the time its location data would not be flagged.

Once this flagging threshold was determined, the simulation then processed both baseline and adversary injection data and flagged it accordingly.

Adversary Injection

Based on the flagging threshold determined within the baseline simulation, a stationary adversary and moving adversary was placed into the system, as seen in APPENDIX B-4: MATLAB Simulation Code – Stationary Adversary Injection and APPENDIX B-5: MATLAB Simulation Code – Moving Adversary Injection, respectively. Each adversary was located 10 km directly north of the center of MCO with the stationary adversary remaining in that location and the moving adversary placed on a randomly generated path. The moving adversary was modeled after a commercially available drone. The random path generated utilized a 50-mph constant speed and always began its broadcast at the same origin as the stationary adversary.

At each point in the ADS-B data stream, the model compared the calculated MLAT location and the reported ADS-B location. If the difference between the two points was greater than the specific flagging threshold, then the model would flag the location data as suspicious. Once the simulation provided all location data, the performance was measured as a binary scale (1 = flagged successfully, 0 = unsuccessfully flagged).

Simulation Setup

The setup for each scenario of the simulation was identical except for the location of the receiver towers. This lone variable was modified to determine what kind of trend could be extracted as the tower locations were altered. Based on the triangular nature of the configuration, the simplest shape to create was an equilateral triangle that centered around the center of MCO. Scenario 1 – 3 remained consistent with the same shape but increased the overall size of the triangle to observe how the measurement error of the model changed. Scenario 4 diverged from the shape trend by modeling a more likely topology by placing the towers at 3 major airports within the state of Florida. As seen in Figure 11, this modified the shape of the created triangle, as well as the overall size.

Each topology was baselined to determine the measurement error of the model and the flagging threshold was calculated to determine the performance during the adversary injection.

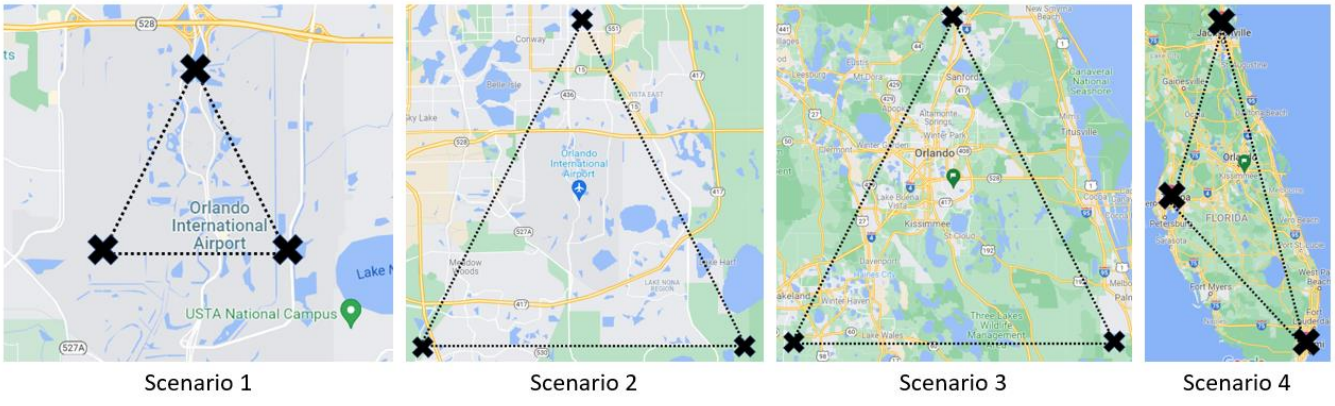

Figure 11: Receiver Tower Topology

Baseline Results

Mean Measurement Error

After 10,000 iterations were run of each scenario, the mean measurement error was recorded to determine if any trend could be derived. Figure 12 shows how each iteration of the baseline approach path varied in mean measurement error. To further illustrate the distribution of the mean measurement error, a histogram was created, as seen in Figure 13.

The distribution seen in Figure 13 is consistent with a normal bell curve, or Gaussian probability distribution, with the mean at the center. The results, seen in Table 3, show a trend in the mean measurement error between scenario 1, 2 and 3. The only modification made between each of these trials was the size of the triangle formed by the receiver towers. These results illustrate that as the towers were spaced further apart, the mean measurement error was reduced.

Scenario 4 diverted from the equilateral triangle shape and placed the receiver towers at 3 major airports across the state. The performance of this topology was similar to scenario 3 but with slightly higher mean measurement error. The size of the created triangle was larger than any other scenarios but contained a different shape. It can be assumed that the size of the created triangle is not the only factor to affect the MLAT calculation but also the shape.

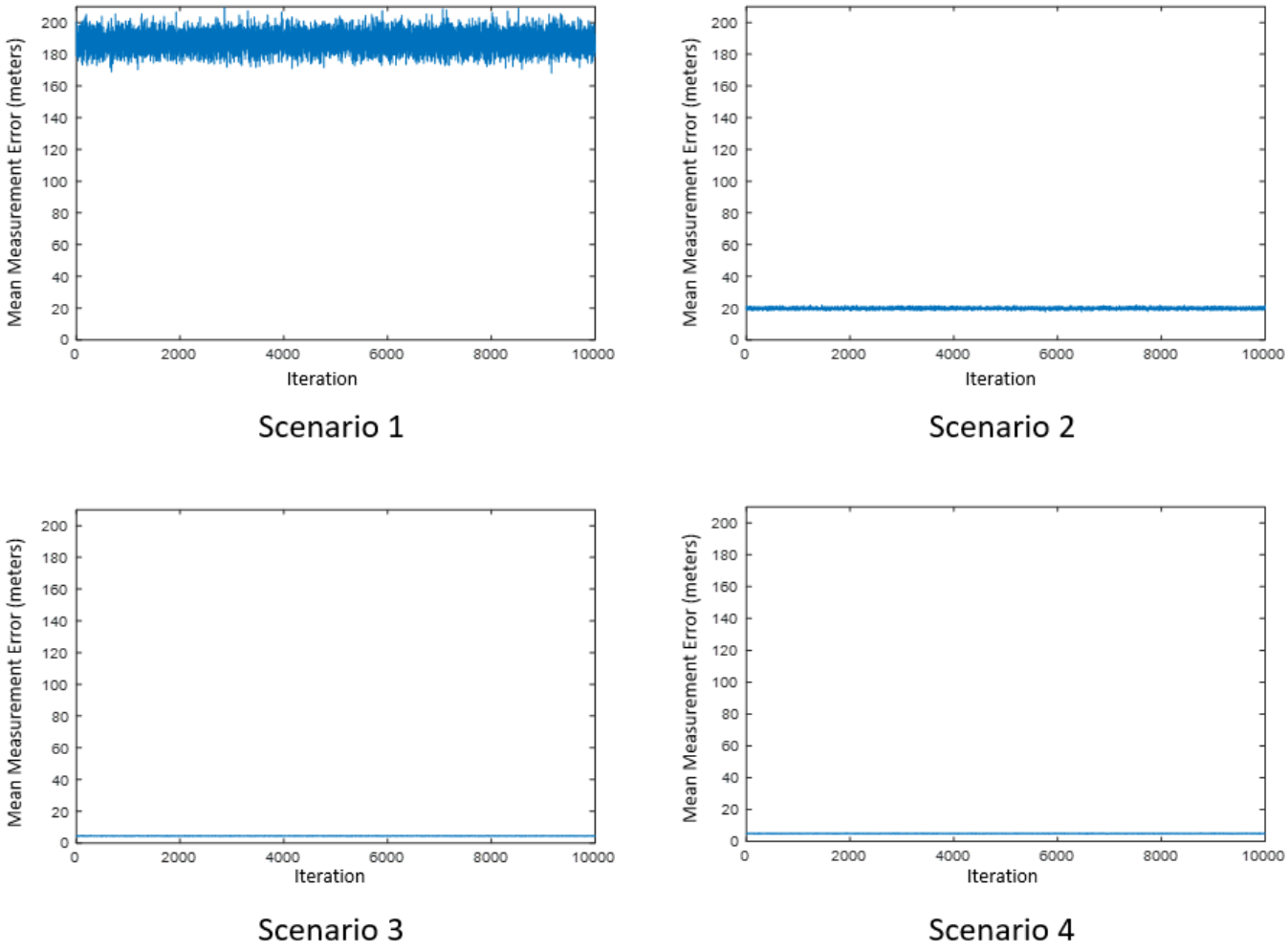

Figure 12: Mean Measurement Error (meters) per Iteration

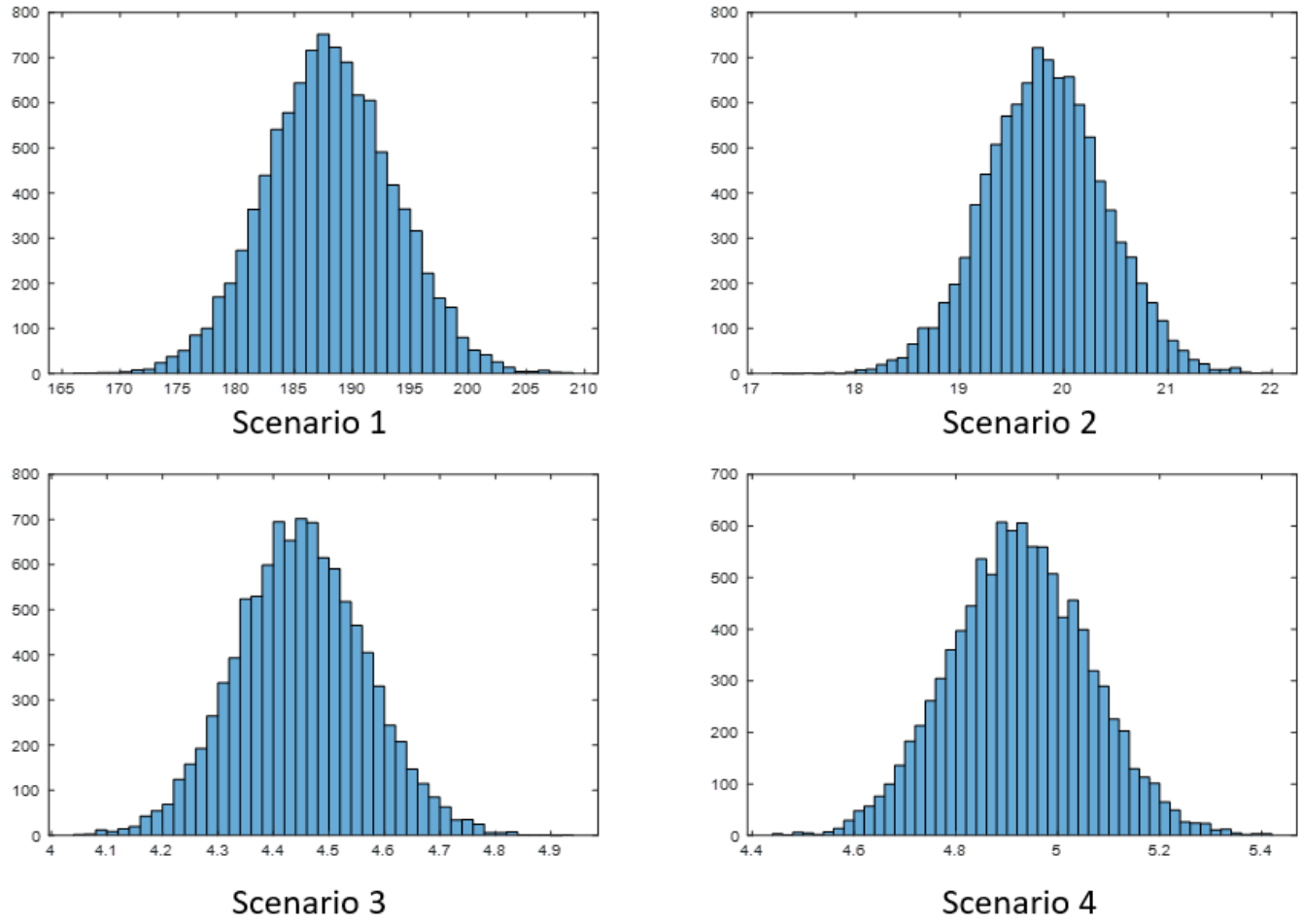

Figure 13: Mean Measurement Error Distribution

Scenario	Measurement Error Mean (m)	Measurement Error Standard Deviation (m)
1	188.08	5.54
2	19.83	0.58
3	4.45	0.12
4	4.92	0.14

Table 3: Baseline Measurement Error Results

Flagging Threshold Results

Once the baseline performance was characterized, the mean measurement error and standard deviation was used to determine the flagging threshold that would be used during the adversary injection portion of each scenario. To

determine the threshold value, a 99% baseline performance level would need to be established. This would ensure that if an actual aircraft was traveling through the airspace and reporting its position over ADS-B, then less than 1% of its location data points would be flagged (i.e., false positives).

To determine the flagging threshold, an offset would be added to the mean measurement error based on the baseline iteration distribution. Based the Gaussian distribution of the mean measurement error data, a Q-function equation was created to determine the applicable offset. Using Equation 3, MATLAB's inverse Q-function algorithm and the known standard deviation (δ), a general equation was created to determine the flagging threshold, as seen in Equation 4.

$$0.99 = 1 - Q(\frac{offset}{\delta}) \quad (3)$$

$$Q\left(\frac{offset}{\delta}\right) = 0.01$$

$$\frac{offset}{\delta} = Q^{-1}[0.01]$$

$$offset = \delta * Q^{-1}[0.01]$$

$$flagging\ threshold = measurement\ error\ mean + offset \quad (4)$$

Using Equation 4, the flagging offset and threshold for each scenario was determined, shown in Table 4. These values were the main outputs of the baseline analysis. These parameters allowed for adversaries to be injected into the simulation to truly determine if MLAT could be a viable means to validate airborne location data from an ADS-B transmission.

Scenario	Flagging Offset (meters)	Flagging Threshold (meters)
1	12.87	200.95
2	1.34	21.17
3	0.27	4.72
4	0.32	5.24

Table 4: Flagging Thresholds

Adversary Injection

Stationary Adversary

Once the flagging threshold for each scenario was determined, the performance of system could be investigated using adversary injection, as shown in Figure 14. First, the stationary adversary was explored by creating an adversary 10 km directly north of the center of MCO. This location was chosen because it is outside of the MCO property line and within an easily accessible area for anyone in the public domain. Each scenario was then synthesized with the signal origin of the ADS-B transmission emanating from the stationary adversary's location. The model compared the calculated MLAT location with the broadcasted ADS-B location and if the two sources varied by more than the scenario's flagging threshold, then the data point would be flagged as suspicious.

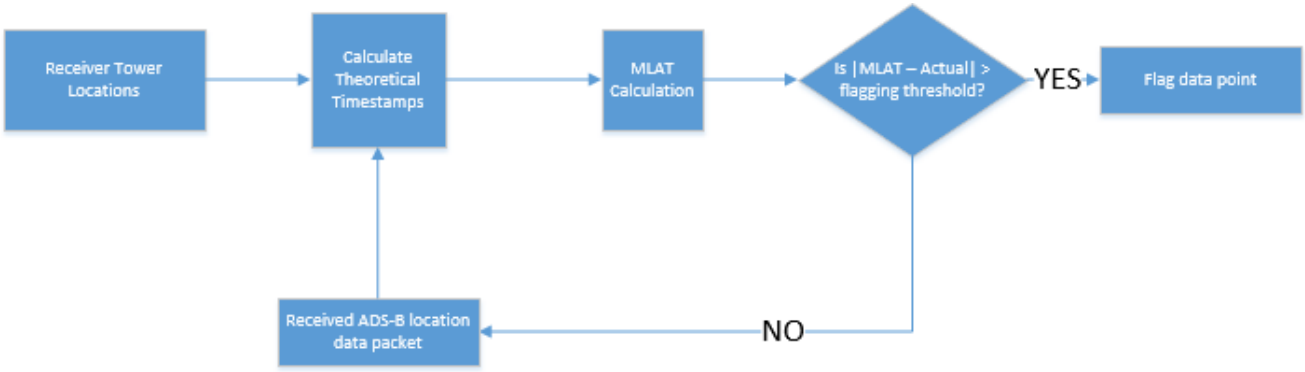

Figure 14: Adversary Injection Flow Chart

At the end of the entire scenario, a "coverage performance" value was determined based on the amount of data points flagged versus the total number of points. Since all data was being broadcasted from the adversary's location then the model should flag all data points for full coverage.

Figure 15 shows the location of the MLAT calculation, in blue, versus the actual location of the adversary, in red. Figure 16 illustrates the complete plot of the simulation, including the broadcasted approach path in blue, the

receiver tower locations in red, the actual adversary location in purple and the calculated signal location in yellow. Hence, Figure 15 is the zoomed-in data of the purple and yellow data points in Figure 16. It should be noted that in Scenario 1, the MLAT receiver towers are located close enough to appear collocated. This is only a limitation of the graphing output of the simulation and not reflective of the actual setup.

Figure 16 shows that the calculated MLAT locations do not overlap with the broadcasted approach path within scenarios 1 – 2. Scenarios 3 – 4 show some overlap but is most likely caused by a limitation in the MATLAB plotting aspect ratio. To objectively determine the coverage performance of the system model, the flagging threshold comparison was completed.

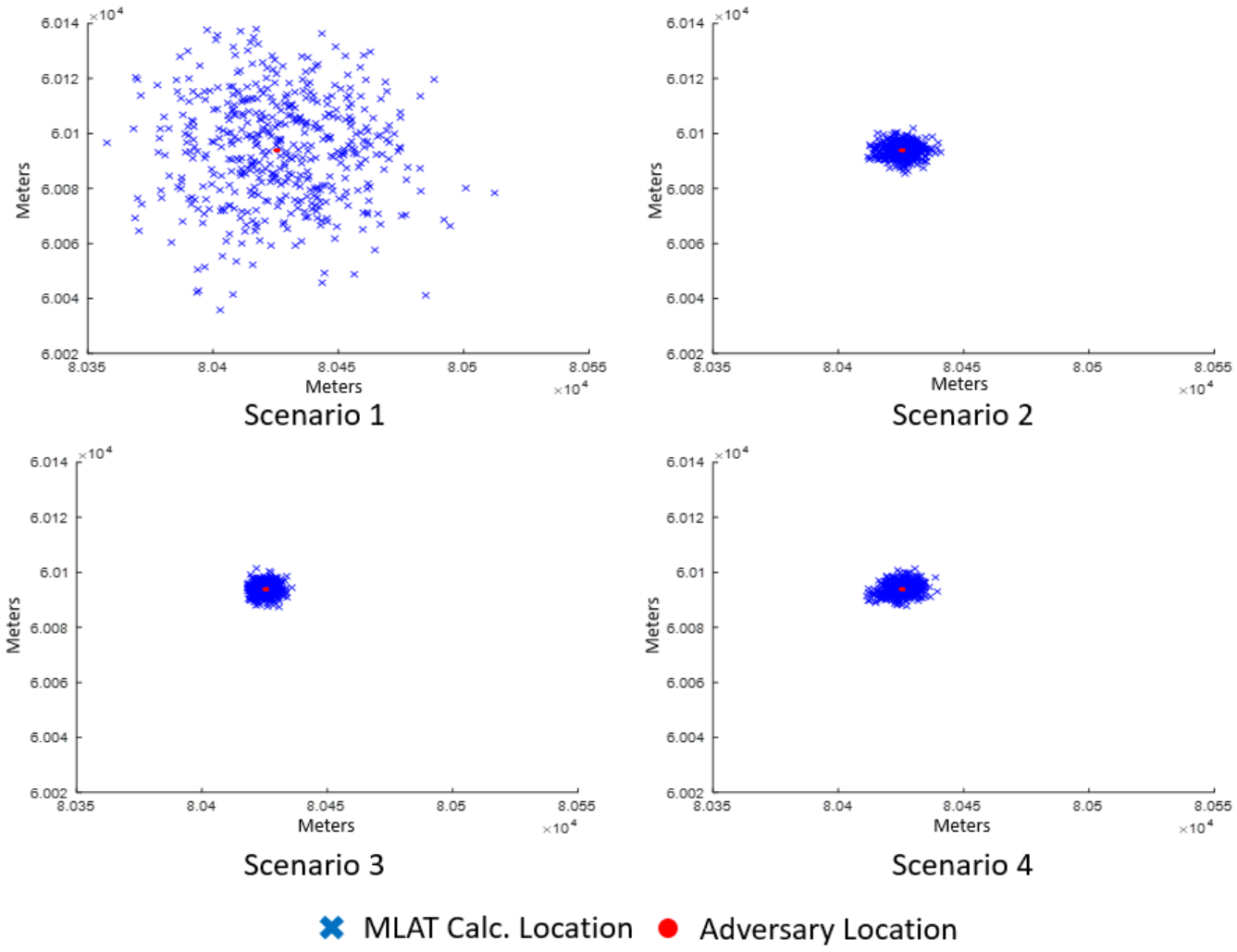

Figure 15: Adversary Location vs MLAT Location

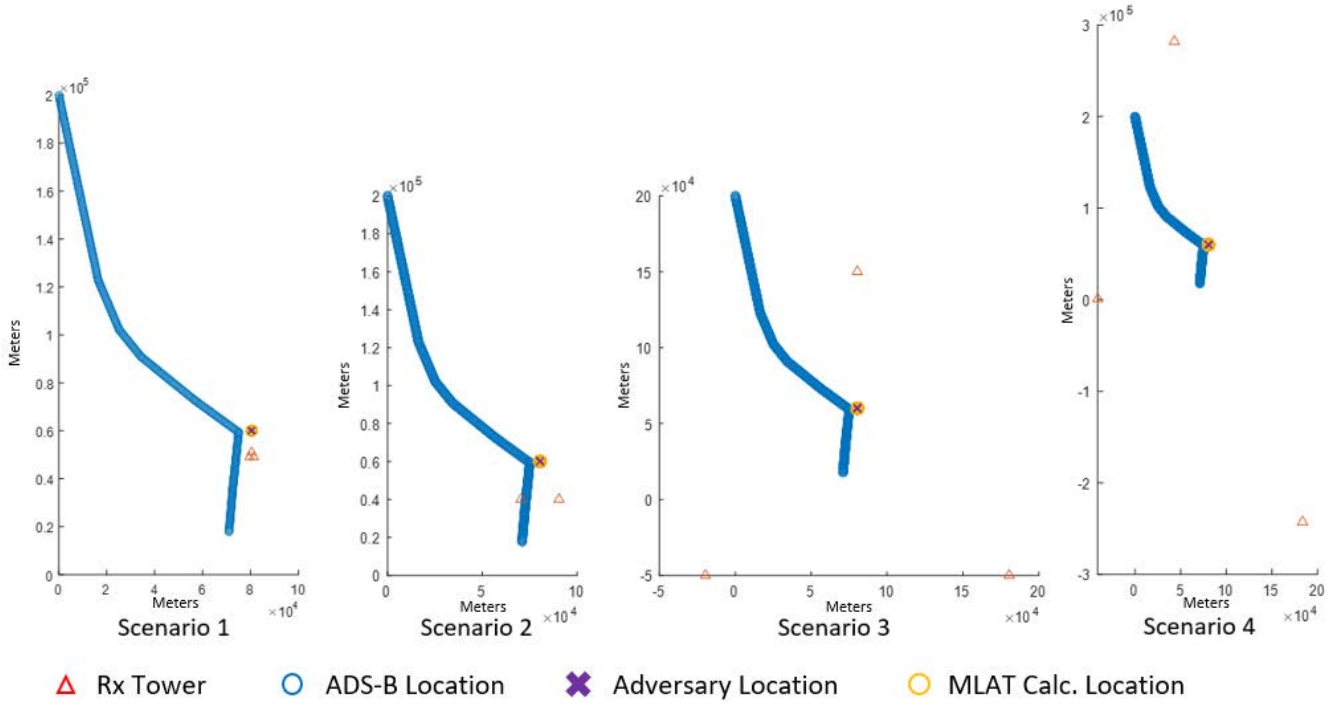

Figure 16: Complete Plot of Stationary Adversary Injection

Table 5 shows the coverage performance of each scenario for a stationary adversary injection. Since the ADS-B location data was being broadcast from a location that is different than the broadcasted location, it is expected that all data points would be flagged, corresponding to a 100% coverage performance. Scenarios 1 – 4 were all able to provide 100% coverage in identifying false location data using MLAT.

Scenario	Location Data Points	Spoofed Data Points	Flagged Data Points	Coverage Performance (%)
1	485	485	485	100
2	485	485	485	100
3	485	485	485	100
4	485	485	485	100

Table 5: Stationary Adversary Coverage

Moving Adversary

Like the stationary adversary, the flagging threshold values were used to gauge the model's performance against a moving adversary. The adversary was modeled after an individual who had the means to modify an existing commercially available drone to allow for mobile ADS-B broadcasting. The path of the drone was modeled as random with a constant speed of 22 m/s (50 mph).

Figure 17 shows the relation between the actual location of the moving adversary versus the calculated location provided by the MLAT algorithm. Scenario 1 – 3 demonstrate the reduction of mean measurement error of each trial through the overlapping of the data points in each plot. Blue triangles represent the actual location of the adversary and red X's represent the calculated location of the MLAT function. Between each scenario, the data points start to converge and nearly overlap by scenario 3. The comparison between actual and calculated location in scenario 4 is near identical to scenario 3. As expected, this is based on the mean measurement error of the two scenarios being near equal.

Figure 18 illustrates the complete plot of the simulation, including the broadcasted approach path in blue, the receiver tower locations in red, the actual adversary location in purple and the calculated signal location in yellow. As with the stationary adversary results, Figure 17 is the zoomed-in data of the purple and yellow data points in Figure 18.

Figure 18 demonstrates that the calculated MLAT locations do not overlap with the broadcasted approach path within scenarios 1 – 2. Like the stationary adversary injection, scenarios 3 – 4 show some overlap but is most likely caused by a limitation in the plotting aspect ratio. The flagging threshold comparison was again completed to objectively determine the coverage performance of the system.

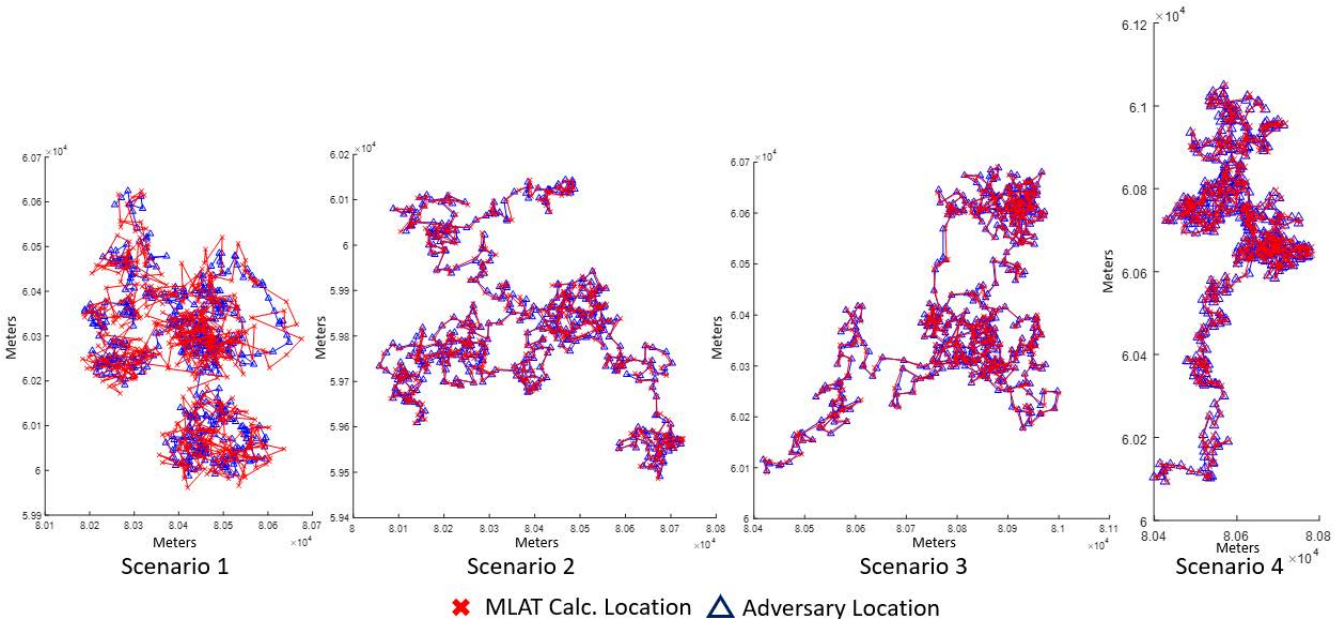

Figure 17: Actual Location vs MLAT Calculated Location of Moving Adversary

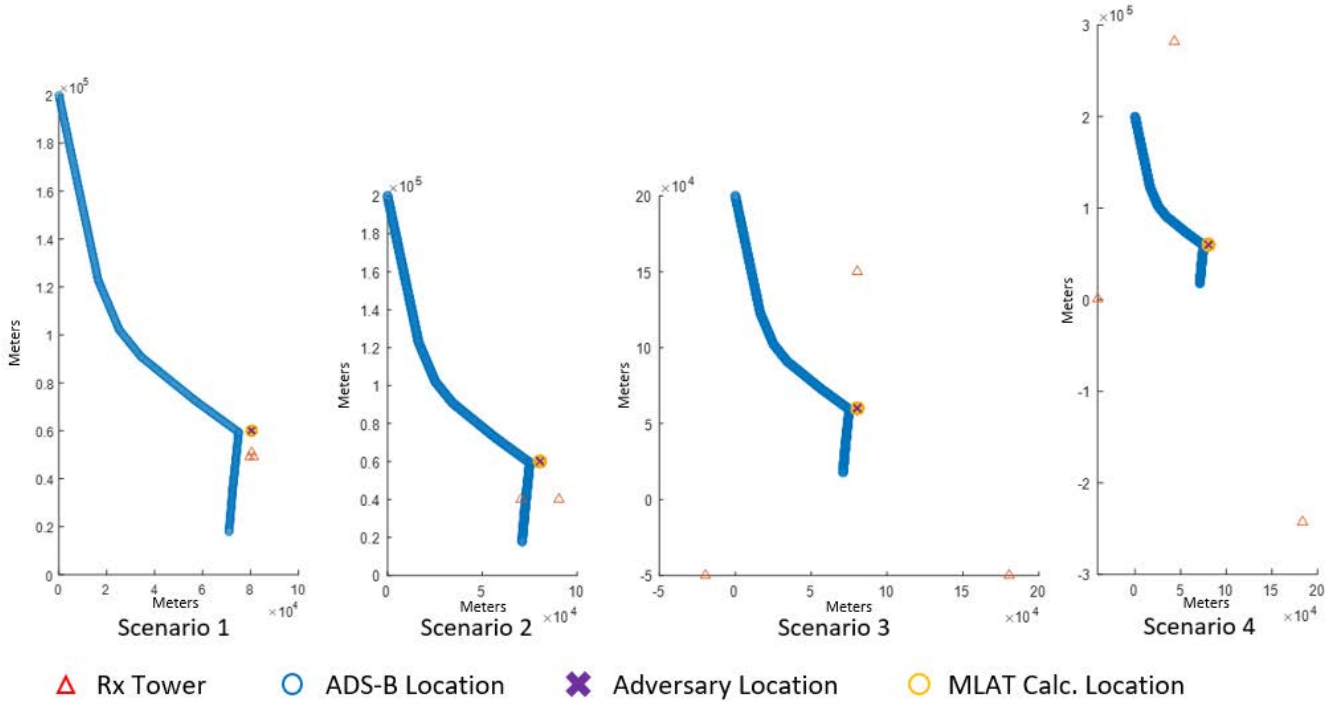

Figure 18: Complete Plot of Moving Adversary Injection

Table 6 shows the coverage performance of each scenario when a moving adversary is injected into the system. Since the ADS-B signal was emanating from a different location than the broadcasted location, it is expected that

all data points would be flagged, corresponding to a 100% coverage performance. Using MLAT, the system model was able to provide 100% coverage in identifying false location data within scenarios 1 – 4.

Scenario	Location Data Points	Spoofed Data Points	Flagged Data Points	Coverage Performance (%)
1	485	485	485	100
2	485	485	485	100
3	485	485	485	100
4	485	485	485	100

Table 6: Moving Adversary Coverage

DISCUSSION

It should be noted that the adversary location was never co-located with the ADS-B location data that was being broadcasted. If this situation had been presented and the location of the adversary was within the flagging threshold, then it would not be flagged properly. This situation was not observed as a stationary adversary would only have one data point that would not be flagged properly, i.e., when the ADS-B location aligned with the actual location of the adversary. By this time, the ATC would have flagged the spoofed aircraft because of the previously flagged data points within its approach path. Additionally, the moving adversary would have a similar performance based on the limited speed of the modeled drone (22 m/s). Based on the initial approach speed (107 m/s) and final approach speed (72 m/s) of a generic commercial jet and the 2 Hz transmission speed, only 1 -2 location data point would be received that would not be flagged. The only realistic way a drone could fool the system would be to travel along its broadcasted path with a speed close to a commercial aircraft.

Finally, because of the highly iterative nature of the simulation, only a small amount of tower topologies were observed. Based on Scenario 1-3, which all contained an equilateral triangle configuration, there is a strong correlation between the distance from the tower center of mass and the measurement error, as shown in Figure 19. Additionally, as the distance between towers was increased within the simulation, the mean measurement error decreased. Based on these two results, it can be concluded that there is likely an ideal distance between towers that will result in the least amount of error and with a center of mass around the received signal origin.

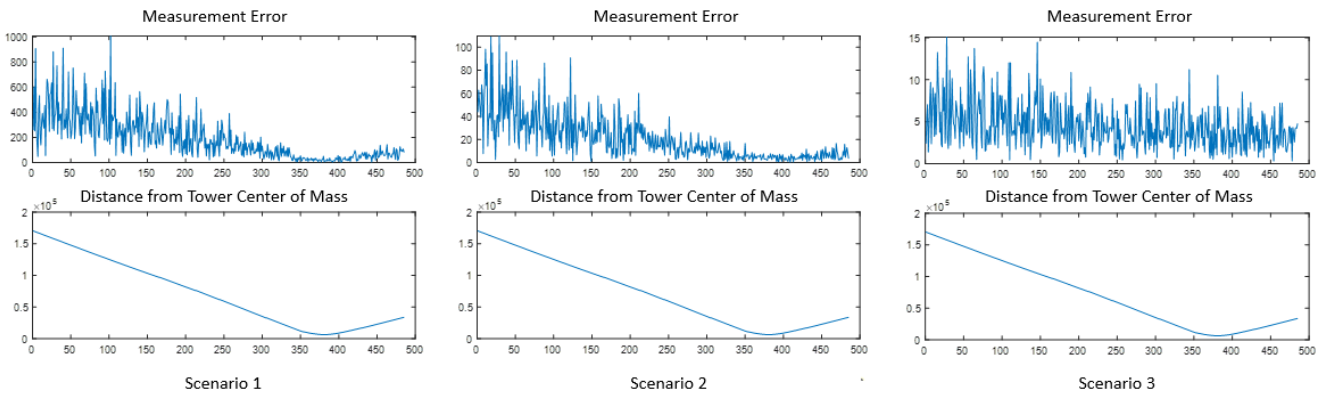

Figure 19: Measurement Error & Distance from Tower Center of Mass Correlation

CONCLUSIONS

The baseline simulation model provided a mean measurement error between 188.08 and 4.45 meters. By simulating the baseline setup over thousands of iterations, a performance distribution was created to determine what value would be used to allow for 99% accuracy in detecting suspicious signals. Once this value was determined, it was then utilized into adversary injection scenarios and was able to correct identify suspicious signals 100% of the time, as seen in Table 7. Based on these results, MLAT is a promising means to validate ADS-B location data.

Scenario	Measurement Error Mean (m)	Measurement Error Standard Deviation (m)	Flagging Threshold (m)	Adversary Flagging Performance (%)
1	188.08	5.54	200.95	100
2	19.83	0.58	21.17	100
3	4.45	0.12	4.72	100
4	4.92	0.14	5.24	100

Table 7: Simulation Performance Results

Based on the performance of the simulated model, a general rule can be assumed. Unless the location of the adversary, whether stationary or moving, is within the flagging threshold level of the broadcasted ADS-B data point, then the model will flag it appropriately. Whether the adversary is moving on a similar path, or stationary has no bearing on the model, since it takes each data point as an independent calculation to determine the validity.

FUTURE WORK

Receiver Tower Topology

Based on the results of the simulation, future investigation should be taken surrounding receiver tower topology. This simulation focused mainly on equilateral triangles, with one non-equilateral triangle observed. Other shape configurations could be investigated to determine if there is an ideal shape for the least amount of measurement error. Furthermore, investigation could be conducted to determine the ideal distance between towers to reduce measurement noise. Based on Scenario 1 – 3, there was a trend for the measurement error to decrease as the distance between the center of mass increased. Investigation should be conducted to determine if there is a convergence value when the measurement error cannot continue to be reduced.

Additionally, research and simulation into the number of towers could be completed to further reduce the measurement error. By finding an ideal tower configuration shape and distance, the ideal case could be sought after for each ADS-B location data packet. By having multiple options to choose from at a given time, an algorithm could be created to determine what towers best met the ideal conditions. This determination could be completed in real time as an aircraft traveled through the airspace to continuously give ATC the least amount of measurement error possible.

Robust System Simulation

One limitation of this research is the elimination of the altitude dimension (z). By reducing the location information to 2-D, it leaves the system susceptible to adversaries that follow the same path as their ADS-B location data but at a different altitude. Research into three-dimension analysis would provide more security to the validation technique of MLAT. Unfortunately, this would require at least 4 signal towers to synthesize the extra dimension of space

[17]. Additionally, more research could be completed surrounding the internal system latencies present in this type of system. Considering equipment, measurement and transmission latencies and their impact on the timestamping of the received signal could greatly impact the accuracy of the MLAT calculation. Finally, research into the latencies surrounding the ADS-B location transmission could be completed. Because the airborne location requires two data packets, additional uncertainty surrounds how the ADS-B location is determined for a moving signal transmitter. An example of a commercial aircraft, moving at a cruising speed of 400 MPH, would be located ~180 meters apart during each of the even and odd location frames. Determining if this is considered by the aircraft transponder and how it could be implemented into the simulation would be ideal. The addition of a third dimension and more realistic system characteristics would provide a more holistic simulation that could model a real-world application more accurately.

Predictive Position Feedback

If locating the source of a spoofed signal was the primary objective, Kalman Filtering could also be introduced into the system model to allow for better predictive modeling of where an adversary could appear next. The simulation presented within this research does not consider any previous information to formulate the result. Kalman filtering would improve this by utilizing assumed data about the system and past location data to give an educated guess on the next state location. This technique could easily be investigated and implemented within a MLAT / ADS-B framework.

In addition to Kalman filtering, additional signal characteristics could be utilized to determine the location of an adversary. By combining TDOA algorithms with other characteristics like signal strength and direction, additional data could be gathered to create a more robust system model.

REFERENCES

[1] F. A. Administration, "14 CFR Part 91 Automatic Dependent Surveillance—Broadcast (ADS–B) Out Performance Requirements To Support Air Traffic Control (ATC) Service; Final Rule," Department of Transportation, 2010.

[2] E. Blasch, P. Paces, P. Kostek and K. Kramer, "Summary of Avionics Technologies," *IEEE A&E SYSTEMS MAGAZINE,* pp. 6-11, September 2015.

[3] A. Helfrick, "The centennial of avionics: Our 100-year trek to performance-based navigation," *IEEE aerospace and electronic systems magazine,* pp. 36-45, September 2015.

[4] H. R. Pilcher, "A Century of Powered Flight," *Nature (London),* 13 December 2003.

[5] H. L. Z. B. a. J. Z. L. Du, "Radar automatic target recognition," *IET Radar, Sonar,* vol. 1, no. 1, pp. 18-26, 2007.

[6] J. Sun, The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals, TU Delft OPEN Publishing, 2020.

[7] R. Francis, R. Vincent, J. Noe, P. Tremblay and D. Desjardins, "The flying laboratory for the observation of ADS-B," *International Journal of Navigation and Observation,* pp. 1-5, 2011.

[8] Brooker and Peter, "SESAR and NextGen: Investing in New Paradigms," *Journal of Navigation,* vol. 61, no. 2, pp. 195-208, 2008.

[9] M. Strohmeier, V. Lenders and I. and Martinovic, "On the security of the automatic dependent surveillance-broadcast protocol," *IEEE Communications Surveys & Tutorials,* vol. 17, no. 2, pp. 1066-1087, 2015.

[10] M. Richards, J. Scheer and W. Holm, Principals of Modern Radar, Edison: SciTech Publishing, 2010.

[11] J. Sun, H. Vû, X. Olive and J. Hoekstra, "Mode S Transponder Comm-B Capabilities in Current Operational Aircraft," *Proceedings,* vol. 59, no. 4, pp. 4-12, 2020.

[12] I. Peppler, From The Ground Up, Ottawa : Aviation Publishers Co, 1996.

[13] R. (. SC-186, "Minimum Operational Performance Standards (MOPS) for 1090 MHz Extended Squitter Automatic Dependent Surveillance - Broadcast (ADS-B) and Traffic Information Services - Broadcast (TIS-B)," RTCA, Incorporated, 2011.

[14] Y. Kim, J.-Y. Jo and S. Lee, "ADS-B Vulnerabilities and a Security Solution with a Timestamp," *IEEE A&E SYSTEMS MAGAZINE,* p. 10, 2017.

[15] D. W. a. P. C. T. Kacem, "Integrity and Authenticity of ADS-B Broadcasts," in *IEEE Aerospace Conference (AeroConf)*, Big Sky, MT, 2015.

[16] K. B. Rasmussen and S. Capkun, "Realization of RF Distance Bounding," in *Proceedings of the 19th USENIX Conference on Security*, Washington, DC, 2010.

[17] Y. T. Chan, "A Simple and Efficient Estimator for Hyperbolic Location," *IEEE TRANSACTIONS ON SIGNAL PROCESSING,* vol. 42, no. 8, pp. 1905-1915, 1994.

[18] Y. Hamdy and S. Mawjoud, "Performance assessment of U-TDOA and A-GPS positioning methods," *International Conference on Future Communication Networks,* pp. 99-104, 2012.

[19] N. a. T. National Coordination Office for Space-Based Positioning, "GPS Accuracy," 2021. [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/. [Accessed 10 January 2021].

[20] SKYbrary, "Approach Speed Categorisation," MediaWiki, 7 October 2017. [Online]. Available: https://www.skybrary.aero/index.php/Approach_Speed_Categorisation. [Accessed 20 March 2021].

[21] K. Hoke, "Air Navigation Name Nonsense," 2 May 2016. [Online]. Available: https://aerosavvy.com/wp-content/uploads/2016/04/MCO-PIGLT-STAR.jpg. [Accessed 2021].

APPENDIX A: Q-Function Lookup Table

x	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.00	5.0000e-01	4.9601e-01	4.9202e-01	4.8803e-01	4.8405e-01	4.8006e-01	4.7608e-01	4.7210e-01	4.6812e-01	4.6414e-01
0.10	4.6017e-01	4.5620e-01	4.5224e-01	4.4828e-01	4.4433e-01	4.4038e-01	4.3644e-01	4.3251e-01	4.2858e-01	4.2465e-01
0.20	4.2074e-01	4.1683e-01	4.1294e-01	4.0905e-01	4.0517e-01	4.0129e-01	3.9743e-01	3.9358e-01	3.8974e-01	3.8591e-01
0.30	3.8209e-01	3.7828e-01	3.7448e-01	3.7070e-01	3.6693e-01	3.6317e-01	3.5942e-01	3.5569e-01	3.5197e-01	3.4827e-01
0.40	3.4458e-01	3.4090e-01	3.3724e-01	3.3360e-01	3.2997e-01	3.2636e-01	3.2276e-01	3.1918e-01	3.1561e-01	3.1207e-01
0.50	3.0854e-01	3.0503e-01	3.0153e-01	2.9806e-01	2.9460e-01	2.9116e-01	2.8774e-01	2.8434e-01	2.8096e-01	2.7760e-01
0.60	2.7425e-01	2.7093e-01	2.6763e-01	2.6435e-01	2.6109e-01	2.5785e-01	2.5463e-01	2.5143e-01	2.4825e-01	2.4510e-01
0.70	2.4196e-01	2.3885e-01	2.3576e-01	2.3270e-01	2.2965e-01	2.2663e-01	2.2363e-01	2.2065e-01	2.1770e-01	2.1476e-01
0.80	2.1186e-01	2.0897e-01	2.0611e-01	2.0327e-01	2.0045e-01	1.9766e-01	1.9489e-01	1.9215e-01	1.8943e-01	1.8673e-01
0.90	1.8406e-01	1.8141e-01	1.7879e-01	1.7619e-01	1.7361e-01	1.7106e-01	1.6853e-01	1.6602e-01	1.6354e-01	1.6109e-01
1.00	1.5866e-01	1.5625e-01	1.5386e-01	1.5151e-01	1.4917e-01	1.4686e-01	1.4457e-01	1.4231e-01	1.4007e-01	1.3786e-01
1.10	1.3567e-01	1.3350e-01	1.3136e-01	1.2924e-01	1.2714e-01	1.2507e-01	1.2302e-01	1.2100e-01	1.1900e-01	1.1702e-01
1.20	1.1507e-01	1.1314e-01	1.1123e-01	1.0935e-01	1.0749e-01	1.0565e-01	1.0383e-01	1.0204e-01	1.0027e-01	9.8525e-02
1.30	9.6800e-02	9.5098e-02	9.3418e-02	9.1759e-02	9.0123e-02	8.8508e-02	8.6915e-02	8.5343e-02	8.3793e-02	8.2264e-02
1.40	8.0757e-02	7.9270e-02	7.7804e-02	7.6359e-02	7.4934e-02	7.3529e-02	7.2145e-02	7.0781e-02	6.9437e-02	6.8112e-02
1.50	6.6807e-02	6.5522e-02	6.4255e-02	6.3008e-02	6.1780e-02	6.0571e-02	5.9380e-02	5.8208e-02	5.7053e-02	5.5917e-02
1.60	5.4799e-02	5.3699e-02	5.2616e-02	5.1551e-02	5.0503e-02	4.9471e-02	4.8457e-02	4.7460e-02	4.6479e-02	4.5514e-02
1.70	4.4565e-02	4.3633e-02	4.2716e-02	4.1815e-02	4.0930e-02	4.0059e-02	3.9204e-02	3.8364e-02	3.7538e-02	3.6727e-02
1.80	3.5930e-02	3.5148e-02	3.4380e-02	3.3625e-02	3.2884e-02	3.2157e-02	3.1443e-02	3.0742e-02	3.0054e-02	2.9379e-02
1.90	2.8717e-02	2.8067e-02	2.7429e-02	2.6803e-02	2.6190e-02	2.5588e-02	2.4998e-02	2.4419e-02	2.3852e-02	2.3295e-02
2.00	2.2750e-02	2.2216e-02	2.1692e-02	2.1178e-02	2.0675e-02	2.0182e-02	1.9699e-02	1.9226e-02	1.8763e-02	1.8309e-02
2.10	1.7864e-02	1.7429e-02	1.7003e-02	1.6586e-02	1.6177e-02	1.5778e-02	1.5386e-02	1.5003e-02	1.4629e-02	1.4262e-02
2.20	1.3903e-02	1.3553e-02	1.3209e-02	1.2874e-02	1.2545e-02	1.2224e-02	1.1911e-02	1.1604e-02	1.1304e-02	1.1011e-02
2.30	1.0724e-02	1.0444e-02	1.0170e-02	9.9031e-03	9.6419e-03	9.3867e-03	9.1375e-03	8.8940e-03	8.6563e-03	8.4242e-03
2.40	8.1975e-03	7.9763e-03	7.7603e-03	7.5494e-03	7.3436e-03	7.1428e-03	6.9469e-03	6.7557e-03	6.5691e-03	6.3872e-03
2.50	6.2097e-03	6.0366e-03	5.8677e-03	5.7031e-03	5.5426e-03	5.3861e-03	5.2336e-03	5.0849e-03	4.9400e-03	4.7988e-03
2.60	4.6612e-03	4.5271e-03	4.3965e-03	4.2692e-03	4.1453e-03	4.0246e-03	3.9070e-03	3.7926e-03	3.6811e-03	3.5726e-03
2.70	3.4670e-03	3.3642e-03	3.2641e-03	3.1667e-03	3.0720e-03	2.9798e-03	2.8901e-03	2.8028e-03	2.7179e-03	2.6354e-03
2.80	2.5551e-03	2.4771e-03	2.4012e-03	2.3274e-03	2.2557e-03	2.1860e-03	2.1182e-03	2.0524e-03	1.9884e-03	1.9262e-03
2.90	1.8658e-03	1.8071e-03	1.7502e-03	1.6948e-03	1.6411e-03	1.5889e-03	1.5382e-03	1.4890e-03	1.4412e-03	1.3949e-03
3.00	1.3499e-03	1.3062e-03	1.2639e-03	1.2228e-03	1.1829e-03	1.1442e-03	1.1067e-03	1.0703e-03	1.0350e-03	1.0008e-03
3.10	9.6760e-04	9.3544e-04	9.0426e-04	8.7403e-04	8.4474e-04	8.1635e-04	7.8885e-04	7.6219e-04	7.3638e-04	7.1136e-04
3.20	6.8714e-04	6.6367e-04	6.4095e-04	6.1895e-04	5.9765e-04	5.7703e-04	5.5706e-04	5.3774e-04	5.1904e-04	5.0094e-04
3.30	4.8342e-04	4.6648e-04	4.5009e-04	4.3423e-04	4.1889e-04	4.0406e-04	3.8971e-04	3.7584e-04	3.6243e-04	3.4946e-04
3.40	3.3693e-04	3.2481e-04	3.1311e-04	3.0179e-04	2.9086e-04	2.8029e-04	2.7009e-04	2.6023e-04	2.5071e-04	2.4151e-04
3.50	2.3263e-04	2.2405e-04	2.1577e-04	2.0778e-04	2.0006e-04	1.9262e-04	1.8543e-04	1.7849e-04	1.7180e-04	1.6534e-04
3.60	1.5911e-04	1.5310e-04	1.4730e-04	1.4171e-04	1.3632e-04	1.3112e-04	1.2611e-04	1.2128e-04	1.1662e-04	1.1213e-04
3.70	1.0780e-04	1.0363e-04	9.9611e-05	9.5740e-05	9.2010e-05	8.8417e-05	8.4957e-05	8.1624e-05	7.8414e-05	7.5324e-05
3.80	7.2348e-05	6.9483e-05	6.6726e-05	6.4072e-05	6.1517e-05	5.9059e-05	5.6694e-05	5.4418e-05	5.2228e-05	5.0122e-05
3.90	4.8096e-05	4.6148e-05	4.4274e-05	4.2473e-05	4.0741e-05	3.9076e-05	3.7475e-05	3.5936e-05	3.4458e-05	3.3037e-05

APPENDIX B-1: MATLAB Simulation Code - PIGLT FOUR Approach Path Creation

```
function segmentPath = approachSegment(start,heading,speed,distance)
% INPUTS
% start: xy coordinate to use as first point of approach segment
% heading: degree measurement to indicate the direction
% speed: speed of movement [m/s]
% distance: naudical miles of travel

% TESTING
% start = [1000,1000];
% heading = 282;
% speed = 200;
% distance = 10;

% set starting point
startX = start(1);
startY = start(2);

% determine points to add
distance_Meters = distance * (1609);
points = round(distance_Meters / (2*speed),0);
distancePerPoint = distance_Meters / points;

% create points
segmentX = zeros(points+1);
segmentY = zeros(points+1);
segmentPath = zeros(points+1,2);

for i=1:size(segmentX)
    if (i==1)
        segmentX(i) = startX;
        segmentY(i) = startY;
        segmentPath = [segmentX(i),segmentY(i)];
    else
            segmentX(i) = segmentX(i-1) + (cos(heading*(pi/180))*distancePerPoint);
            segmentY(i) = segmentY(i-1) + (sin(heading*(pi/180))*distancePerPoint);
            segmentPath = vertcat(segmentPath,[segmentX(i),segmentY(i)]);
    end
end

% return segment
% plot(segmentPath(:,1:2:end),segmentPath(:,2:2:end));
return;
```

```
function [approach,MCO_Central] = MCO_Approach
% Create baseline approach path
% Hardcoded converted heading (aviation -> cartisian), speed & distance
% based on PIGLT FOUR approach documentation


segment1 = approachSegment([0,200000],282,240,49);
approach = segment1;

segment2 =
approachSegment([approach(size(approach,1),1),approach(size(approach,1),2)],293,240
,14);
approach = vertcat(approach,segment2);

segment3 =
approachSegment([approach(size(approach,1),1),approach(size(approach,1),2)],309,240
,9);
approach = vertcat(approach,segment3);

segment4 =
approachSegment([approach(size(approach,1),1),approach(size(approach,1),2)],321,240
,18);
approach = vertcat(approach,segment4);

segment5 =
approachSegment([approach(size(approach,1),1),approach(size(approach,1),2)],324,240
,14);
approach = vertcat(approach,segment5);
MCO_Central = [approach(size(approach,1),1) + 5387,approach(size(approach,1),2) -
9332];

segment6 =
approachSegment([approach(size(approach,1),1),approach(size(approach,1),2)],264,160
,15);
approach = vertcat(approach,segment6);

segment7 =
approachSegment([approach(size(approach,1),1),approach(size(approach,1),2)],265,160
,11);
approach = vertcat(approach,segment7);

%histogram(sqrt((MCO_Central(1,1) - 0)^2 + (MCO_Central(1,2)-200000)^2));


return;
```

APPENDIX B-2: MATLAB Simulation Code – TDOA Arithmetic

```
function location = chanTdoaMath(sensor_LOC)
% Implementation of a 2d estimation based on Chan source

% INPUTS
% sensor_LOC: 3x3 matrix of tower location coordinates and associated time stamps
% [xi yi ti; xj yj tj; xk yk tk; xl yl tl]

% CONSTANTS;
C = 299792458;

% MAIN
t1 = sensor_LOC(1,3);
t2 = sensor_LOC(2,3);
t3 = sensor_LOC(3,3);

x1 = sensor_LOC(1,1);
x2 = sensor_LOC(2,1);
x3 = sensor_LOC(3,1);

y1 = sensor_LOC(1,2);
y2 = sensor_LOC(2,2);
y3 = sensor_LOC(3,2);

K1 = x1^2 + y1^2;
K2 = x2^2 + y2^2;
K3 = x3^2 + y3^2;

r1 = C*t1;
r2 = C*t2;
r3 = C*t3;

r21 = r2 - r1;
r31 = r3 - r1;

x21 = x2 - x1;
x31 = x3 - x1;

y21 = y2 - y1;
y31 = y3 - y1;

location = transpose(([x21 y21;x31 y31])^(-1) * ([r21;r31]*r1 + (1/2)*(([((r21)^2 -
K2 + K1); ((r31)^2 - K3 + K1)])));

return;
```

APPENDIX B-3: MATLAB Simulation Code – Baseline Performance

```
% FUNCTIONALITY
% Main simulation sequenced over time to observe how accuracy changes with
% a moving signal. Historical data used to provide a baseline approach
% vector to compare with adversarial interactions.
clear;
% CONSTANTS
C = 299792458; % meters / second
noise_mean = 0;
noise_std_dev = 10.695*10^(-9); % seconds
gps_noise_std_dev = 0.713;

[Act_Vector,mcoCentral] = MCO_Approach;

% Scenerio 1
tower1 = [mcoCentral(1,1)-1000 mcoCentral(1,2)-1000];
tower2 = [mcoCentral(1,1) mcoCentral(1,2)+1000];
tower3 = [mcoCentral(1,1)+1000 mcoCentral(1,2)-1000];

% Scenerio 2
% tower1 = [mcoCentral(1,1)-10000 mcoCentral(1,2)-10000];
% tower2 = [mcoCentral(1,1) mcoCentral(1,2)+10000];
% tower3 = [mcoCentral(1,1)+10000 mcoCentral(1,2)-10000];

% Scenerio 3
% tower1 = [mcoCentral(1,1)-100000 mcoCentral(1,2)-100000];
% tower2 = [mcoCentral(1,1) mcoCentral(1,2)+100000];
% tower3 = [mcoCentral(1,1)+100000 mcoCentral(1,2)-100000];

% Scenerio 4
% tower1 = [mcoCentral(1,1)-37324 mcoCentral(1,2)+231966]; % Jacksonville
% tower2 = [mcoCentral(1,1)-121253 mcoCentral(1,2)-49246]; % Tampa
% tower3 = [mcoCentral(1,1)+103006 mcoCentral(1,2)-292839]; % Miami

towerX = [tower1(1) tower2(1) tower3(1)];
towerY = [tower1(2) tower2(2) tower3(2)];
ADSB_Vector = zeros(size(Act_Vector,1),2);

% MAIN

% Time sequencing
% Loop based on size of Act_Vector / ADSB_Vector
interval = size(Act_Vector,1);
errorVector = zeros(10000,1);
Calc_Loc_Vec = zeros(interval,2);
CompareXY = zeros(interval,2);
Compare_R = zeros(interval,1);
distance_from_MCO = zeros(interval,1);

for k = 1:size(errorVector)
    for m = 1:size(Act_Vector,1)
      % Add in GPS location noise to each iteration
```

```
        ADSB_Vector(m,1) = Act_Vector(m,1) + normrnd(noise_mean,
gps_noise_std_dev);
        ADSB_Vector(m,2) = Act_Vector(m,2) + normrnd(noise_mean,
gps_noise_std_dev);
    end

    for i = 1:interval
        % Use tower locations to determine theoretical timestamping and actual time
        %stamping based on clock noise
        t1 = (sqrt((tower1(1)-Act_Vector(i,1))^2+(tower1(2)-Act_Vector(i,2))^2))/C;
        t2 = (sqrt((tower2(1)-Act_Vector(i,1))^2+(tower2(2)-Act_Vector(i,2))^2))/C;
        t3 = (sqrt((tower3(1)-Act_Vector(i,1))^2+(tower3(2)-Act_Vector(i,2))^2))/C;

        t1_actual = t1+normrnd(noise_mean, noise_std_dev);
        t2_actual = t2+normrnd(noise_mean, noise_std_dev);
        t3_actual = t3+normrnd(noise_mean, noise_std_dev);



        % Determine calculated location based on MLAT
        MLAT_calc = [tower1(1) tower1(2) t1_actual; tower2(1) tower2(2) t2_actual;
            tower3(1) tower3(2) t3_actual];
        Calc_Loc = chanTdoaMath(MLAT_calc);

        % Compare calculated location with ADSB_Vector
        Compare_ADSB_Loc = [abs(Calc_Loc(1)-ADSB_Vector(i,1)) abs(Calc_Loc(2)-
ADSB_Vector(i,2))];
        Compare_Range = sqrt((Calc_Loc(1)-ADSB_Vector(i,1))^2+(Calc_Loc(2)-
ADSB_Vector(i,2))^2);
        distance_MCO = sqrt((Act_Vector(i,1)-mcoCentral(1,1))^2+(Act_Vector(i,2)-
mcoCentral(1,2))^2);

        % Concat location comparision and calculated location to create vectors
        if(i==1)
            Calc_Loc_Vec = Calc_Loc;
            CompareXY = Compare_ADSB_Loc;
            Compare_R = Compare_Range;
            distance_from_MCO = distance_MCO;
        else
            Calc_Loc_Vec = vertcat(Calc_Loc_Vec,Calc_Loc);
            CompareXY = vertcat(CompareXY,Compare_ADSB_Loc);
            Compare_R = vertcat(Compare_R,Compare_Range);
            distance_from_MCO = vertcat(distance_from_MCO,distance_MCO);
        end

    end
    error = mean(Compare_R);
    if(k==1)
        errorVector = error;
    else
        errorVector = vertcat(errorVector,error);
    end
end
figure;
plot(errorVector);
figure;
histogram(errorVector);
```

```
trialAverage = mean(errorVector);
trialStdDev = std(errorVector);

threshold = trialAverage+(qfuncinv(0.01)*trialStdDev);
thresholdCheck = zeros(size(errorVector));

% for j=1:size(errorVector)
%     if errorVector(j) > threshold
%         thresholdCheck(j) = 1;
%     else
%         thresholdCheck(j) = 0;
%     end
% end
%
% histogram(thresholdCheck);

% Plot calculated location vs ADSB location
Calc_X = Calc_Loc_Vec(:,1:2:end);
Calc_Y = Calc_Loc_Vec(:,2:2:end);
ADSB_X = ADSB_Vector(:,1:2:end);
ADSB_Y = ADSB_Vector(:,2:2:end);
figure;
hold on
plot(ADSB_X, ADSB_Y,'-ob');
plot(Calc_X, Calc_Y,':xr');
plot(towerX,towerY,'^k');
%axis([-10000 100000 0 250000]);
daspect([1 1 1]);
hold off

% Plot location comparison over time
figure;
subplot(2,1,1);
plot(Compare_R);

subplot(2,1,2);
plot(distance_from_MCO);
```

APPENDIX B-4: MATLAB Simulation Code – Stationary Adversary Injection

```
% FUNCTIONALITY
% Main simulation sequenced over time to observe how accuracy changes with
% a moving signal. Historical data used to provide a baseline approach
% vector to compare with adversarial interactions.
clear;
% CONSTANTS
C = 299792458; % meters / second
noise_mean = 0;
noise_std_dev = 10.695*10^(-9); % seconds

% Scenerio 1-4
ADV_Threshold = 200.95; % meters
% ADV_Threshold = 21.17; % meters
% ADV_Threshold = 4.72; % meters
% ADV_Threshold = 5.24; % meters


[ADSB_Vector,mcoCentral] = MCO_Approach;

% Scenerio 1
tower1 = [mcoCentral(1,1)-1000 mcoCentral(1,2)-1000];
tower2 = [mcoCentral(1,1) mcoCentral(1,2)+1000];
tower3 = [mcoCentral(1,1)+1000 mcoCentral(1,2)-1000];

% Scenerio 2
% tower1 = [mcoCentral(1,1)-10000 mcoCentral(1,2)-10000];
% tower2 = [mcoCentral(1,1) mcoCentral(1,2)+10000];
% tower3 = [mcoCentral(1,1)+10000 mcoCentral(1,2)-10000];
%
% Scenerio 3
% tower1 = [mcoCentral(1,1)-10000 mcoCentral(1,2)-10000];
% tower2 = [mcoCentral(1,1) mcoCentral(1,2)+10000];
% tower3 = [mcoCentral(1,1)+10000 mcoCentral(1,2)-10000];

% Scenerio 4
% tower1 = [mcoCentral(1,1)-37324 mcoCentral(1,2)+231966]; % Jacksonville
% tower2 = [mcoCentral(1,1)-121253 mcoCentral(1,2)-49246]; % Tampa
% tower3 = [mcoCentral(1,1)+103006 mcoCentral(1,2)-292839]; % Miami

towerX = [tower1(1) tower2(1) tower3(1)];
towerY = [tower1(2) tower2(2) tower3(2)];

Act_Vector = zeros(size(ADSB_Vector,1),2);

for j=1:size(ADSB_Vector,1)
    Act_Vector(j,1) = mcoCentral(1,1);
    Act_Vector(j,2) = mcoCentral(1,2) + 10000;
end

% MAIN

% Time sequencing
% Loop based on size of Act_Vector / ADSB_Vector
```

```
interval = size(Act_Vector,1);
for i = 1:interval
    % Use tower locations to determine theoretical timestamping and actual time
    %stamping based on clock noise
    t1 = (sqrt((tower1(1)-Act_Vector(i,1))^2+(tower1(2)-Act_Vector(i,2))^2))/C;
    t2 = (sqrt((tower2(1)-Act_Vector(i,1))^2+(tower2(2)-Act_Vector(i,2))^2))/C;
    t3 = (sqrt((tower3(1)-Act_Vector(i,1))^2+(tower3(2)-Act_Vector(i,2))^2))/C;

    t1_actual = t1+normrnd(noise_mean, noise_std_dev);
    t2_actual = t2+normrnd(noise_mean, noise_std_dev);
    t3_actual = t3+normrnd(noise_mean, noise_std_dev);

    % Determine calculated location based on MLAT
    MLAT_calc = [tower1(1) tower1(2) t1_actual; tower2(1) tower2(2) t2_actual;
        tower3(1) tower3(2) t3_actual];
    Calc_Loc = chanTdoaMath(MLAT_calc);

    % Compare calculated location with Actual location
    Compare_Act_Loc = [abs(Calc_Loc(1)-Act_Vector(i,1)) abs(Calc_Loc(2)-
Act_Vector(i,2))];
    Compare_Range = sqrt((Calc_Loc(1)-Act_Vector(i,1))^2+(Calc_Loc(2)-
Act_Vector(i,2))^2);

    % Concat location comparision and calculated location to create vectors
    if(i==1)
        Calc_Loc_Vec = Calc_Loc;
        CompareXY = Compare_Act_Loc;
        Compare_R = Compare_Range;
    else
        Calc_Loc_Vec = vertcat(Calc_Loc_Vec,Calc_Loc);
        CompareXY = vertcat(CompareXY,Compare_Act_Loc);
        Compare_R = vertcat(Compare_R,Compare_Range);
    end

end

ADV_Check = zeros(size(ADSB_Vector,1),1);
ADV_Count = 0;
for k=1:size(ADSB_Vector,1)
   if (sqrt((Calc_Loc_Vec(k)-ADSB_Vector(k))^2) > ADV_Threshold)
       ADV_Check(k) = 1;
       ADV_Count = ADV_Count + 1;
   else
       ADV_Check(k) = 0;
   end
end

ADV_Capture = 100*(ADV_Count / size(ADV_Check,1));
histogram(ADV_Check);

figure;
% Plot calculated location vs ADSB location
Calc_X = Calc_Loc_Vec(:,1:2:end);
Calc_Y = Calc_Loc_Vec(:,2:2:end);
ADSB_X = ADSB_Vector(:,1:2:end);
ADSB_Y = ADSB_Vector(:,2:2:end);
hold on
```

```
plot(ADSB_X, ADSB_Y);
plot(towerX,towerY,'^');
plot(Act_Vector(:,1:2:end), Act_Vector(:,2:2:end),'.r','MarkerSize', 10);
plot(Calc_X, Calc_Y,'x');
%axis([-10000 100000 0 250000]);
daspect([1 1 1]);
hold off;
```

APPENDIX B-5: MATLAB Simulation Code – Moving Adversary Injection

```
% FUNCTIONALITY
% Main simulation sequenced over time to observe how accuracy changes with
% a moving signal. Historical data used to provide a baseline approach
% vector to compare with adversarial interactions.
clear;
% CONSTANTS
C = 299792458; % meters / second
noise_mean = 0;
noise_std_dev = 10.695*10^(-9); % seconds
% Scenerio 1-4
%ADV_Threshold = 200.95; % meters
%ADV_Threshold = 21.17; % meters
%ADV_Threshold = 4.72; % meters
ADV_Threshold = 5.24; % meters


[ADSB_Vector,mcoCentral] = MCO_Approach;

% Scenerio 1
% tower1 = [mcoCentral(1,1)-1000 mcoCentral(1,2)-1000];
% tower2 = [mcoCentral(1,1) mcoCentral(1,2)+1000];
% tower3 = [mcoCentral(1,1)+1000 mcoCentral(1,2)-1000];

% Scenerio 2
% tower1 = [mcoCentral(1,1)-10000 mcoCentral(1,2)-10000];
% tower2 = [mcoCentral(1,1) mcoCentral(1,2)+10000];
% tower3 = [mcoCentral(1,1)+10000 mcoCentral(1,2)-10000];

% Scenerio 3
% tower1 = [mcoCentral(1,1)-10000 mcoCentral(1,2)-10000];
% tower2 = [mcoCentral(1,1) mcoCentral(1,2)+10000];
% tower3 = [mcoCentral(1,1)+10000 mcoCentral(1,2)-10000];

% Scenerio 4
tower1 = [mcoCentral(1,1)-37324 mcoCentral(1,2)+231966]; % Jacksonville
tower2 = [mcoCentral(1,1)-121253 mcoCentral(1,2)-49246]; % Tampa
tower3 = [mcoCentral(1,1)+103006 mcoCentral(1,2)-292839]; % Miami

towerX = [tower1(1) tower2(1) tower3(1)];
towerY = [tower1(2) tower2(2) tower3(2)];

Act_Vector = zeros(size(ADSB_Vector,1),2);

for j=1:size(ADSB_Vector,1)
    if (j==1)
        Act_Vector(j,1) = mcoCentral(1,1);
        Act_Vector(j,2)= mcoCentral(1,2) + 10000;
    else
        % Create random movement
        speed = normrnd(22,10);
        vector = randi([0 360]);
        Act_Vector(j,1) = Act_Vector(j-1,1) + speed*cos(vector);
        Act_Vector(j,2) = Act_Vector(j-1,2) + speed*sin(vector);
    end
```

```
end

% MAIN

% Time sequencing
% Loop based on size of Act_Vector / ADSB_Vector
interval = size(Act_Vector,1);
for i = 1:interval
    % Use tower locations to determine theoretical timestamping and actual time
    %stamping based on clock noise
    t1 = (sqrt((tower1(1)-Act_Vector(i,1))^2+(tower1(2)-Act_Vector(i,2))^2))/C;
    t2 = (sqrt((tower2(1)-Act_Vector(i,1))^2+(tower2(2)-Act_Vector(i,2))^2))/C;
    t3 = (sqrt((tower3(1)-Act_Vector(i,1))^2+(tower3(2)-Act_Vector(i,2))^2))/C;

    t1_actual = t1+normrnd(noise_mean, noise_std_dev);
    t2_actual = t2+normrnd(noise_mean, noise_std_dev);
    t3_actual = t3+normrnd(noise_mean, noise_std_dev);

    % Determine calculated location based on MLAT
    MLAT_calc = [tower1(1) tower1(2) t1_actual; tower2(1) tower2(2) t2_actual;
        tower3(1) tower3(2) t3_actual];
    Calc_Loc = chanTdoaMath(MLAT_calc);

    % Compare calculated location with Actual location
    Compare_Act_Loc = [abs(Calc_Loc(1)-Act_Vector(i,1)) abs(Calc_Loc(2)-
Act_Vector(i,2))];
    Compare_Range = sqrt((Calc_Loc(1)-Act_Vector(i,1))^2+(Calc_Loc(2)-
Act_Vector(i,2))^2);

    % Concat location comparision and calculated location to create vectors
    if(i==1)
        Calc_Loc_Vec = Calc_Loc;
        CompareXY = Compare_Act_Loc;
        Compare_R = Compare_Range;
    else
        Calc_Loc_Vec = vertcat(Calc_Loc_Vec,Calc_Loc);
        CompareXY = vertcat(CompareXY,Compare_Act_Loc);
        Compare_R = vertcat(Compare_R,Compare_Range);
    end

end

ADV_Check = zeros(size(ADSB_Vector,1),1);
ADV_Count = 0;
for k=1:size(ADSB_Vector,1)
   if (sqrt((Calc_Loc_Vec(k)-ADSB_Vector(k))^2) > ADV_Threshold)
       ADV_Check(k) = 1;
       ADV_Count = ADV_Count + 1;
   else
       ADV_Check(k) = 0;
   end
end

ADV_Capture = 100*(ADV_Count / size(ADV_Check,1));
histogram(ADV_Check);

figure;
```

```
% Plot calculated location vs ADSB location
Calc_X = Calc_Loc_Vec(:,1:2:end);
Calc_Y = Calc_Loc_Vec(:,2:2:end);
ADSB_X = ADSB_Vector(:,1:2:end);
ADSB_Y = ADSB_Vector(:,2:2:end);
hold on
plot(ADSB_X, ADSB_Y,'o');
plot(towerX,towerY,'^');
plot(Act_Vector(:,1:2:end), Act_Vector(:,2:2:end),'Marker','.','MarkerSize', 30);
plot(Calc_X, Calc_Y,'x');
%axis([-10000 100000 0 250000]);
daspect([1 1 1]);
hold off;
```