An ADS-B Intrusion Detection System

Thabet Kacem, Duminda Wijesekera, Paulo Costa, Alexandre Barreto
Radio and Radar Engineering Laboratory
Volgenau School of Engineering, George Mason University
Fairfax, Virginia 22030-4444, USA
Email: [tkacem,dwijesek,pcosta]@gmu.edu, adebarro@c4i.gmu.edu

***Abstract*—Advances in radio and RADAR research have considerably contributed to the emergence of novel air traffic control (ATC) protocols. In particular, Automatic Dependent Surveillance Broadcast (ADS-B) [21] has proven to be a viable option that could complement and extend current technologies. Despite the benefits offered by ADS-B, such as more precise location, easier deployment, and lower costs, cyber-security issues have arisen due to the open, clear-text broadcasting of ADS-B messages making it vulnerable to many attacks. In this paper, we propose an intrusion detection system (IDS) specifically designed to detect malicious or questionable ADS-B messages. We leverage prior work on cyber-defense mechanisms against ADS-B attacks and physical aspects of aircraft motion to classify received ADS-B traffic as potential attacks and retain digital artifacts that might support post-attack forensic investigations.**

I. Introduction

ADS-B has emerged in recent years as a viable alternative to current radio and RADAR standards in Aircraft signaling. Its advantages include superior location accuracy due to the use of Global Position System (GPS), ease of deploying and maintaining ADS-B stations compared to traditional RADAR stations, and lower purchasing costs.

Software-defined radio (SDR) is a technology that migrates hardware components to software, which enables developers to customize the desired functionality a radio should have. This technology has been applied to various fields and its use in air traffic control (ATC) research is causing the emergence of new promising protocols, including in ADS-B applications. Piracci et al. [21] describe the technical and hardware challenges to build an SDR device dedicated to ADS-B. However, these benefits did not eliminate ADS-B's vulnerabilities related to its open clear-text broadcasting of messages. Several research efforts [1], [3], [18] demonstrated in a controlled and simulated environment that it is possible to launch attacks on ADS-B.

These efforts were focused in demonstrating the vulnerabilities, and did not provide a systemic approach to detecting the attacks and focused instead on specific types of attacks. Few researchers aimed at proposing a comprehensive approach to thwart ADS-B attacks from a cyber-security view, such as in [4] and in [17]. Unfortunately, as we discuss below, neither approach is currently implementable.

In previous work [6], [8], [9], we proposed different variants of a cyber-security module that verifies the authenticity and integrity of transmitted ADS-B messages. The module thwarts several message-injection attacks by using keyed-hash message authentication code (HMAC). The difference between the versions is the way the HMAC digest is generated and used to create the security metadata in place of the Cyclic Redundancy Check (CRC) field. The latter is used for bit-error verification and can be easily broken. Also, in [7], we proposed a mechanism to securely exchange the keys used for the HMAC algorithm. We leveraged the fact that aircraft need to obtain authorizations from the entities governing the ATC zones included in its flight path before taking off. We made a simplifying assumption that these entities are referred to as the ATC centers. Then, we delegate the key negotiation to source ATC center, which initiates secure handshakes with the ATCs that control other zones in the flight path to exchange the secret keys over public key infrastructure (PKI) schemes.

In this paper, we propose an Intrusion Detection System (IDS) for ADS-B signals by combining above described previous work with new techniques acting both at the ADS-B sender and receiver in order to strengthen the attack detection. This is achieved by assessing the cyber-physical environment of supposedly signaling aircraft. This IDS detects potential attacks and provides as a result artifacts that could be used for digital forensic analyses.

The rest of the paper is organized as follows. Section II discusses the related work while section III describes the background information in ADS-B. Section IV describes the details of our proposed intrusion detection system. Section V provides experimental evaluation of our IDS system.

II. Related Work

Costin and Francillon [1] showed the possibility of exploiting the vulnerabilities of ADS-B to launch injection attacks on a simulated environment. In that study, they did not propose a systematic way to classify different attacks and thwart them, and provided high-level recommendations but stopping short of getting into implementation details. Similarly, McCallie et al. [18] presented an attack taxonomy based on the difficulty of the attack and developed some attacks in a simulated environment to demonstrate the feasibility of launching ADS-B attacks. The authors however did not provide an implementation of this that detects and classifies attacks while providing forensics about the attacks.

Pan et al. [17] proposed a scheme to thwart ADS-B injection attacks by combining elliptical curve cipher with X.509 digital signatures. However, this scheme suffered from poor scalability because it requires sending extra messages to fit the digital signature for each ADS-B message, which is not feasible in

2324-9013/16 $31.00 © 2016 IEEE
DOI 10.1109/TrustCom.2016.107

the narrow and overly used 1090 MHz band assigned to ADS-B. Strohmeier et al. [4] presented a location verification-based approach to detect ADS-B attacks by elaborating schemes that rely on statistical analysis of the time difference of arrival (TDOA). Our approach uses cryptography to detect attacks, classify them and, more importantly, to provide the system expert with digital forensics about the attacks.

In [10], we proposed a lightweight method to locate malicious ADS-B transmitters by leveraging data obtained from multilateration. We computed the time difference of timestamps, where each timestamp is composed of the arrival time of the ADS-B message and processing time, which is the time to demodulate and timestamp the ADS-B message. The difference, in this paper, is that we use this method to obtain the real location of the attacker, which we use as one artifact to be used as digital forensic about the attack.

Liu et al. [19] proposed a logical-based model to extend NIST National Vulnerability Database (NVD) by leveraging evidence databases and anti-forensics databases in order to map evidence to vulnerability which would facilitate network forensics analysis. Our IDS produces an evidence database of performed attacks that could help performing forensics analysis for ADS-B.

III. ADS-B Background

ADS-B operates in two modes, namely, ADS-B Out and ADS-B In as shown in Figure 1. The former is deployed on aircraft that gets its location from the nearby satellite to make an ADS-B messages, modulate it using Pulse Position Modulation (PPM) before broadcasting it every second in the 1090 MHz band. ADSB-In is mostly deployed on ground components, i.e. ADS-B stations and ATC centers, that receive ADS-B messages from air crafts that are within transmission range, which is in the order of 150 nautical miles [12].

In practice, 1090 Mode S Extended Squitter is an implementation that supports ADS-B, where ADS-B data is encapsulated

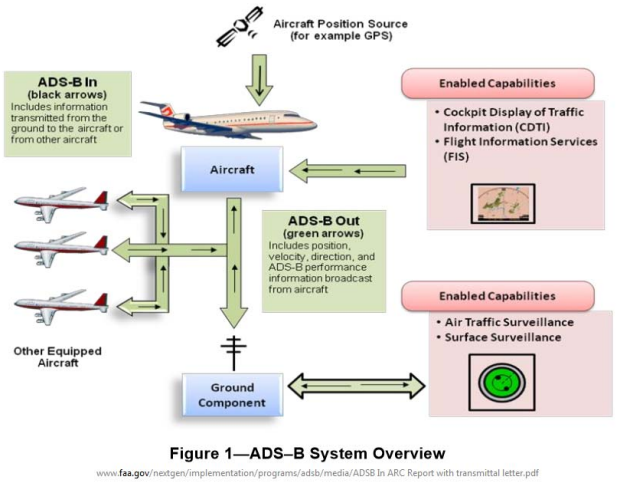

Fig. 1. ADS-B Modes of Operations

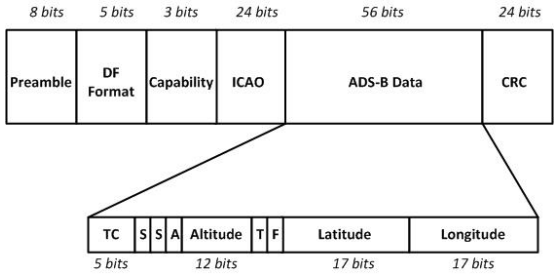

Fig. 2. ADS-B Packet Format

in Mode S frames as shown in Figure 2. The preamble is used for synchronization and the DF Format field indicates the protocol being used, where the 17 decimal represents ADS-B. The capability bits indicate the sub-protocol in use and the International Civil Aviation Organization (ICAO) identifier uniquely identifies each aircraft. The CRC field is used for bit error verification. The ADS-B data occupies 56 bits of the 112 bits of Modes S frames and provide several information such longitude, latitude, altitude, velocity, etc.

Each GPS location is encoded using two ADS-B messages: even and odd. This is due to the use of the Compact Position Reporting (CPR) [2] in order to encode and decode latitude and longitude using an optimal number of bits (17 bit for longitude and 17 bits for latitude, where 23 bits owuld be needed if plain binary encoding is used). The even/odd parity is used to divide the earth in longitudinal and latitudinal lines to determine the exact location.

However, due to clear-text ADS-B messages broadcast, it is possible to use cheap hardware and launch ADS-B attacks. It is possible to download an open-source ADS-B receiver and relatively inexpensive radios to at least eavesdrop on air traffic. Besides, if the attacker has the ability to craft ADS-B messages and broadcast them on the 1090 MHz band, this can cause injection attacks that would create more problems.

We developed an attack taxonomy to study the threat model of ADS-B, where we used two classification criteria: the difficulty of implementation of the attack and the location of the radio used for the attack. Therefore, we concluded that there are three categories:

- **Medium-level attacks:** includes most traditional message injection attacks where the geo-location information used to create the ADS-B message is randomly generated and the radio used for the attack is immobile.
- **Advanced-level attacks:** the geo-location is generated from a flight simulator program which creates realistic trajectories while the radio is also immobile.
- **Expert-level attacks:** with proliferation of manned and unmanned aerial vehicles, the radio used for the attack could be located on an aircraft or drone.

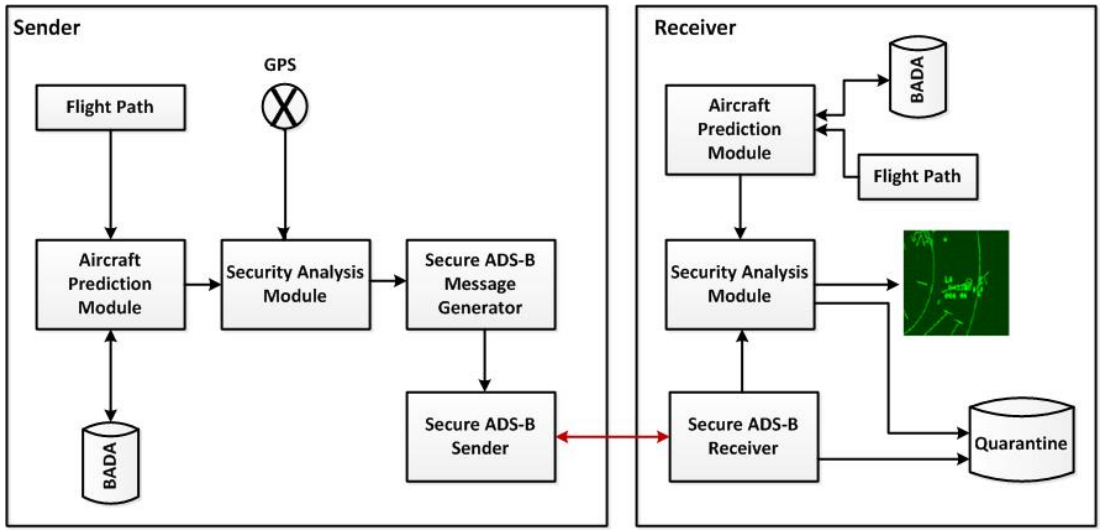

Fig. 3. ADS-B IDS Overview

IV. ADS-B IDS Framework

A. Overview

An overview of the proposed ADS-B IDS is provided in Figure 3. First, at the sender, the goal is to craft secure ADS-B messages based on valid location data. Therefore, the aircraft passes the obtained GPS location at time t along with the corresponding predicted one. The latter is obtained from the aircraft prediction module [11] that computes the flight path in term of series of GPS locations every second by leveraging the Base of Aircraft Data (BADA) that defines specific parameters for each aircraft according to its make and model.

Then, the security analysis module at the sender compares both location to check if there is an inconsistency, and if the pilot is very far from its supposed flight path it could indicate a GPS spoofing attack. Consequently, the result is the location that would be used to generate and send the secure ADS-B message.

At the receiver, the goal is to verify the integrity and authenticity of the transmitted ADS-B messages, double-check that with data obtained from the physical environment of the aircraft and finally locate the attacker and create a corresponding entry in the quarantine.

B. Security Analysis Module at the Sender

Once the aircraft obtain its GPS location from nearby satellite, the security analysis module at the sender checks for conformity of the position with regards to the flight path that is generated by the aircraft prediction module [11] before takeoff. This module generates a series of geo-coordinates for a specific make and model of an aircraft by leveraging parameters from BADA.

If the GPS location deviates, at a time t, from the corresponding predicted one beyond a certain threshold, this indicates that the reported position is not within the aircraft safe zone at time t. In the next subsection, we describe how we defined this safe zone. In this case, there could be a potential GPS spoofing attack or inappropriate maneuver by the pilot.

Algorithm 1: ADS-B Sender Security Analysis Algorithm

```
1  Initial Conditions: GPS available, pre-set threshold and
   flight path is determined;
2  String icao = getICAO();
3  String makeModel = GetMM();
4  Position predicted =
   getGeoPoint(icao,makeModel,flightPath,t);
5  Position actual = getGPSPosition(t);
6  int distance d = computeDistance(predicted, actual);
7  if d ≤ threshold then
8      Generate ADS-B message from GPS position;
9  else
10     Check if there is a GPS Spoofing attack;
```

Algorithm 1 describes process of validating the location information to use in order to generate ADS-B messages. In line 1, we set the initial conditions assuming that GPS data is available, that there is a pre-set threshold beyond which a GPS position is flagged suspicious and that the flight path is available. In lines $2-3$, we extract the ICAO number and the make and model of the aircraft in order to use them along with the flight path to determine the predicted position, in line 4, i.e. where the aircraft should be at a time t. In line 5, we extract the GPS location of the aircraft from the nearby satellite. Finally, in lines $7-10$, we check if the distance between the actual and predicted positions is less than the pre-set threshold, in which case we generate the ADS-B message. Otherwise, we check if a the aircraft is under GPS spoofing attack.

Details about how to detect if there is a GPS spoofing attack is beyond the scope of this paper and approaches such as in [20] could be integrated within the IDS in the future. In case

of wrong pilot maneuver, the voice line could be leveraged to communicate with him or her and solve the issue.

C. Safe Zone of an Aircraft

This subsection describe the safe zone of an aircraft, as shown in Figure 4, that sets the upper and lower bounds for a reported position in the sender or receiver. We define this safe as a sphere having as diameter the overall length of the aircraft. For a reported position to be valid, its distance from the predicted position at an instant t, is less than or equal to the diameter of the sphere.

In order to compute this distance between two points in the geographic coordinate system, we use the Haversine formula stated in Equation 1, where Δ Lat, Δ Lon, Δ Alt correspond, respectively, to the difference between the reported and predicted latitude, longitude and altitude :

$$\begin{aligned} a &= \sin^2(\Delta Lat/2) + \cos(Lat_{predicted}) \\ &\quad * \cos(Lat_{Reported}) * \sin^2(\Delta Lon/2) \\ c &= 2 * atan2(\sqrt{a}, \sqrt{(1-a)}) \\ d &= \sqrt{(R*c)^2 + \Delta^2 Alt} \end{aligned} \quad (1)$$

Then, if this distance is less to the value of the diameter of the sphere representing the safe zone, then it is a valid position update as the green marker in Figure 4. Otherwise, it is not a valid one such as the red marker in Figure 4. The diameter of the sphere depends on the make and model of the aircraft and can be gotten from BADA database.

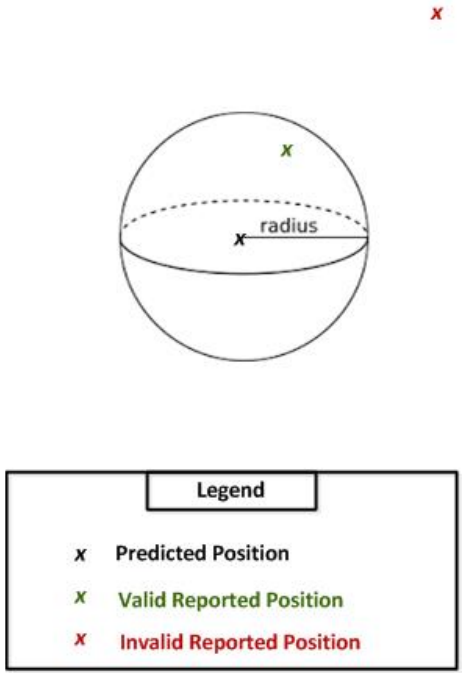

Fig. 4. Aircraft Safe Zone

D. Secure ADS-B Message Generator

After the security analysis module checks the validity of the location to be used, the secure message generator encodes it into the two corresponding even and odd ADS-B messages. However, in order to enforce authenticity and integrity of these messages, there is a challenge of fitting the HMAC-based metadata without changing the packet format or adding extra messages because the smallest size of HMAC is 128 bits which is greater than the whole Mode S frame.

Algorithm 2 describes the process of generating secure ADS-B messages from plain ones. In line 1, we present the initial conditions stating that the GPS position to use have already been validated by the security analysis module and the HMAC key to use in the zone where the aircraft flies is available. In line $2-3$, we generate the plain ADS-B messages, i.e. the even and odd ones, to encode a GPS position. In line 4, we run the HMAC algorithm based on the two ADS-B messages in question and the key before splitting the resulting digest in multiple of 8. For example, we split a 128 bit digest into 8 portions and we split a 256 bit digest into 16 portions.

Then, in lines $6-7$, we makes two even-sized sets out the portions before running several rounds of a chosen bit manipulation algorithm to each set in lines $8-12$. The number of rounds is basically half of the set's size. Then, we delete the previous elements from each set and we replace then by the output of this algorithm for each set after running all the rounds. In lines $13-14$, we generate the metadata for the even and odd messages based on the content of set1 and set2. Each metadata has 8 bits sequence number and 16 bits resulting from the bit manipulation algorithm.

Algorithm 2: Secure ADS-B Message Generation Algorithm

```
1  Initial Conditions: Valid GPS position and HMAC key are available.
2  String adsbEven = generate(gps,0);
3  String adsbOdd = generate(gps,1);
4  String digest = hmac(adsbEven,adsbOdd,key);
5  String[] portions = digest.split();
6  String[] set1 = choose(portions,4);
7  String[] set2 = choose(portions,4);
8  int nbRounds = set1.length() / 2;
9  int index =0;
10 while index ≤ nbRounds do
11     bitManipAlgo(set1);
12     bitManipAlgo(set2);
13 String metaEven = genMetadata(set1[0]);
14 String metaOdd = genMetadata(set2[0]);
15 String secMsgEven =
   concat(payload(adsbEven),metaEven);
16 String secMsgOdd = concat(payload(adsbOdd),metaOdd);
```

E. Secure ADS-B Message Receiver

Once ADS-B messages are received, they need to be checked for integrity and authenticity before further checks are performed by the security analysis module.

Algorithm 3 describes the process of validating incoming messages that are first grouped according to their ICAO identifier then to their sequence number. Thus, we used a doubly nested hashmap to that effect which facilitates the computation of the HMAC digest.

In line 1, of Algorithm 3, several variables are initialized. Line 2 marks the start of the while loop checking for incoming ADS-B messages. Line 3 extracts the ICAO identifier and the sequence number while lines $4-7$ checks for existence of the hashmaps and create them accordingly. The first hashmap *bucket* has as key the ICAO number and the second hashmap *sequence* as mapped content. This latter has the sequence number as key and a queue containing the ADS-B messages as mapped content. Line 8 enqueues each message in the corresponding queue's hashmap.

In line 9, we check the size of the queue for a given ICAO and sequence number. If it is equal to 2, that means we can start verifying the HMAC. Lines $10-14$ extract parts of the metadata to assemble the HMAC digest while computing the HMAC based on the payloads and the used key. In lines $15-18$, we check if the computed versus assembled HMACs match and we store the result in a binary variable which we pass to the security analysis module, in line 19, along with miscellaneous information about the aircraft and ADS-B messages involved.

In lines $20-21$, we resolve the case where the size of the hashmap is less than 2 meaning that one or both messages are not received. Therefore, in lines $22-26$, we apply a 30 seconds timeout and if the messages are received we return to line 9 to check the HMAC. Otherwise, we pass the result to the security analysis module that can determine if this is due to transmission power such as weak signal power. This cannot undue the failure to compute the safe zone but can solve this problem with the next transmission by ordering the aircraft to apply some adaptation mechanisms such as increasing its signal power or avoiding a zone where risk factors affect ADS-B transmissions.

Algorithm 3: Secure ADS-B Receiver Algorithm

```
1  Initiate icao hashmap "bucket", sequence hashmap
     "sequence", longMessage, ch_even, ch_odd, rh_even, rh_odd;
2  while new ADS-B message is received do
3      Extract icao and sequence numbers from ADS-B
       message;
4      if icao already exists in icao hashmap then
5          Retrieve the sequence map and queue;
6      else
7          Create new icao and sequence hashmaps;
8      Enqueue the ADS-B message in the corresponding
       sequence Hashmap's queue;
9      if size(sequence(icao))==2 then
10         Extract received hash portion from the even and
           odd messages and store them in rh_even, rh_odd,
           respectively;
11         Concatenate the payloads of the even and odd
           messages and store it in longMessage;
12         Apply the HMAC algorithm using longMessage
           and the appropriate key;
13         Split the HMAC digest into 8 portions;
14         Apply several rounds of the bit manipulation
           algorithms to get the last two and store them in
           ch_even, ch_odd;
15         if ch_even==rh_even and ch_odd==rh_odd then
16             match = 1;
17         else
18             match = 0;
19         Pass the result of the comparison to the Security
           Analysis Module;
20     else
21         if size(sequence(icao)) < 2 then
22             Wait 30 seconds timeout for the remaining
               message;
23             if timeout elapsed and other message is not
               received then
24                 Move the previously received message in
                   Quarantine;
25             else
26                 Return to line 9;
27         else
28             Pass the messages to the Security Analysis
               Module;
```

F. Security Analysis Module at the Receiver

The objective of this module is to run another check on received messages regardless whether their HMAC verification failed or succeeded. Therefore, it computes the location encoded by the ADS-B messages in question and checks if it is within the safe zone, according to guidelines defined in subsection IV-C.

Algorithm 4 describes the logic of this component. Line 1 defines the initial conditions assuming that most of ADS-B are received and that the flight path is already known. Lines $2-5$ extract from the ADS-B message and the BADA database, respectively, the ICAO number, the sequence number, the queue corresponding to an ICAO and a sequence number, and the make and model of the aircraft. Lines $6-7$ compute the predicted position, at a instant t, based on the flight path of the aircraft in question and the declared position based on decoding the ADS-B messages in question. Line 8 calculates the distance between both positions using the Haversine formula defined in subsection IV-C. Line 9 computes the threshold which determines the final decision if the ADS-B position is

Algorithm 4: ADS-B Receiver Security Analysis Algorithm

```
1  Initial Conditions: Most ADS-B messages are received and flight path is determined;
2  String icao = getICAO(adsbMsg);
3  String seqNb = getSeqNb(adsbMsg);
4  Queue q = fetch(icao,seqNB);
5  String makeModel = GetMM();
6  Position predicted = getGeoPoint(icao,makeModel,flightPath,t);
7  Position actual = getADSBPosition(t, q);
8  int distance d = computeDistance(predicted, actual);
9  threshold = getOverallLength(makeModel);
10 if d ≤ threshold then
11     Validate the position report;
12     if HMAC verification failed then
13         Apply adaptation techniques to avoid receiving incorrect in ADS-B messages;
14 else
15     Confirm the existence of a message injection attack;
16     Create an entry in the quarantine;
17     Use multilateration to trace the real location of the attacker;
18     if HMAC verification succeeded then
19         Change the key because it is compromised;
```

really valid or not. This is obtained by extracting the overall length of the aircraft. For a ADS-B position to be valid, its distance from the corresponding one at a time t has to be within the safe zone, i.e. less than the threshold.

In lines $10-13$, explore the cases where the measured distance is less than the threshold. Therefore, we validate the position report. In addition, if the HMAC verification failed, we conclude it is due to risk factors affecting ADS-B broadcast. Therefore, we initiate some adaptation techniques to fix that according to [9].

Conversely, in lines $14-19$, we explore the cases where this distance is greater than the threshold. In this case, we confirm the existence of message injection attack. Therefore, we create an entry for the message(s) in question specifying its time of arrival, the type of attack and we leverage multilateration techniques to determine the real location of the attacker. This is done by computing the time difference of arrival from different sensors at the ATC center such as in [10]

G. Quarantine

The quarantine is a database where artifacts, related to attacks or isolated ADS-B messages, are stored. These artifacts can serve as input for digital forensics experts to bridge the gap between ADS-B vulnerabilities and ADS-B attacks. The structure of the quarantine is as follows:

- **ADS-B Message:** the message to be stored in quarantine.
- **Timestamp:** time of arrival of the ADS-B message.
- **Type of Attack:** specifies the type of suspected attack if applicable.
- **Sender's Position:** specifies the longitude, latitude and altitude of the attacker if any, which is obtained by leveraging multilateration techniques such as in [10].

V. Evaluation

A. ADS-B Test Bed

In order to conduct our experiments, we setup a test bed shown in Figure 5 to send and receive ADS-B messages through the radio nodes. The components are described as follows:

- **Track Source:** the tracks are obtained from the output Excel files created by the Aircraft Trajectory Predictor module. We extract the coordinates, create and broadcast the corresponding secure even and odd ADS-B messages.
- **Nodes:** Every Node is composed of a PC/Laptop running Linux Ubuntu that has the GNU Radio platform [14] and uses an Ettus N200 radio device [15] to broadcast and receive the messages. There are two kinds of nodes in our test bed:
  - **Sending Nodes:** Transmit legitimate or malicious ADS-B messages.
  - **Receiving Nodes:** we extended gr-air-modes to comply with the our security framework, i.e. when ADS-B messages are received by these nodes, they are decoded to determine their geographical location and then forwarded to the ADS-B server.
- **Anechoic Chamber:** We use an Anechoic chamber to isolate transmissions from exterior noise and interference.
- **Database:** We use a Sqlite database to store the locations of the generated tracks that are received by the ADS-B server.
- **Radar Display:** We use Google Earth interface to visualize the locations of all aircraft in real-time.

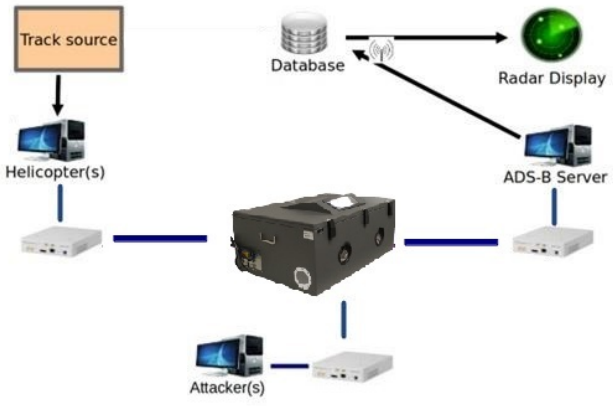

Fig. 5. ADS-B Test Bed

B. Experimental Results

In this subsection, we present the results that we obtained by running a set of experiments on the test bed to evaluate our system. We used two metrics to collect data:

- **Sending Time:** It is the sum of the time to generate the plain ADS-B message after the sanity check of the location, the time to generate the security metadata, the time to write the resulting secure ADS-B message into the Field-Programmable Gate Array (FPGA) of the radio and the time to send it.
- **Receiving Time:** It is the sum of the time to receive the incoming messages, the time to verify the security metadata, the time to perform the additional security analysis check and the time to create possible entries in the quarantine.

First, we measured the sending time, in seconds, of plain ADS-B versus secure ADS-B for every pair of even/odd ADS-B messages, shown in Figure 6. We observed that the secure ADS-B has a very small overhead, with a maximum value of 0.03 seconds, while the sending time has an average of 2.35 seconds for ever pair of messages.

Second, we measured the receiving time, in seconds, of plain ADS-B versus secure ADS-B for every pair of even/odd ADS-B messages, shown in Figure 7. We observed that secure ADS-B introduces a jitter ranging from 0.004 seconds to 0.008 seconds, which is negligible compared to the 12 seconds jitter of Secondary Surveillance Radar (SSR).

From this experiments, we can conclude that our secure ADS-B framework presents good performance measurements and operates with negligible jitter.

VI. Conclusions

We proposed a novel intrusion detection system to secure ADS-B and prevent attackers from exploiting its open-text open broadcast nature in order to launch attacks against ATC operations. We showed that this IDS presents promising performance values and operates with minimal overhead.

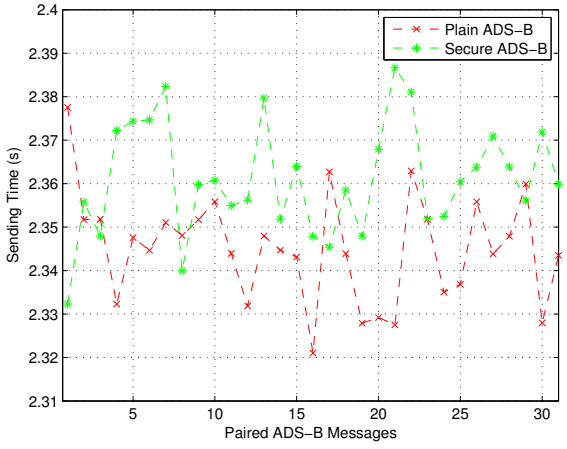

Fig. 6. ADS-B Sending Time

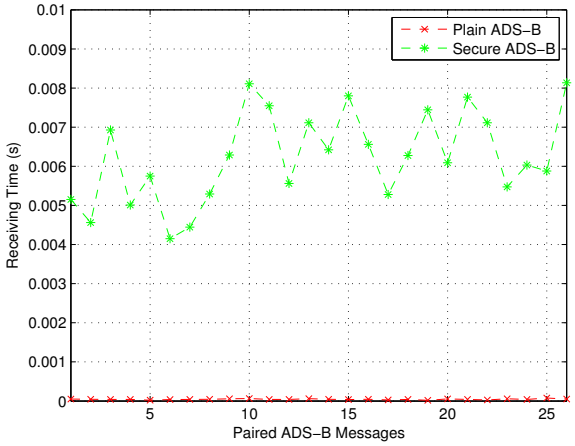

Fig. 7. ADS-B Sending Time

Acknowledgment

This work is funded by Department of Homeland Security Grant 2012-ST-104-000047 and Federal Railroad Administration Grant FR-TEC-0010-2015.

References

[1] A. Costin and A. Francillon, “Ghost in the Air(Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices,” 2012.
[2] Marshall, “ADS-B 1090 MOPS, Revision B.
[3] M. Strohmeier, V. Lenders, and I. Martinovic, ”On the Security of the Automatic Dependent Surveillance-Broadcast”, Communications Surveys & Tutorials, IEEE 17.2 (2015): 1066-1087.
[4] M. Strohmeier, Martin, V. Lenders, and I. Martinovic, ”Lightweight location verification in air traffic surveillance networks”, Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, pages 49-60, April 2015.
[5] Bradford, Steve. ”NextGen progress and ICAO”. Proceedings of the Integrated Communications, Navigation and Surveillance Conference (ICNS), 2014. IEEE, 2014. p. 1-22.
[6] T. Kacem, D. Wijesekera, and P. Costa, “Integrity and authenticity of ADS-B broadcasts, *IEEE Aerospace Conference (AeroConf)*, Big Sky, MT, USA, March 2015.
[7] T. Kacem, D. Wijesekera, P. Costa, J. Carvalho, M. Monteiro, and A. Barreto, ”Key distribution mechanism in secure ADS-B networks”, in *Integrated Communication, Navigation and Surveillance Conference (ICNS)*, Herndon, VA, USA, April 2015.
[8] T. Kacem, D. Wijesekera, P. Costa, J. Carvalho, M. Monteiro, and A. Barreto, ”Secure ADS-B Design & Evaluation”, IEEE International Conference on Vehicular Electronics and Safety (ICVES), November 2015, Yokohama, Japan.
[9] T. Kacem, J. Carvalho, D. Wijesekera, P. Costa, M. Monteiro, and A. Barreto, ”Risk-Adaptive Engine for Secure ADS-B Broadcasts”, SAE AeroTech, September 2015, Seattle, WA.
[10] M. Conte, A. Barreto, T. Kacem, D. Wijesekera, P. Costa, ”Detecting Malicious ADS-B Transmitters Using a Low-Bandwidth Sensor Network”, INternational Conference on Information Fusion, July 2015, Washington, D.C.
[11] Robert Pastor, Aircraft Trajectory Predictor Project Available at https://github.com/RobertPastor/AircraftTrajectoryPredictor
[12] ADS-B Frequently Asked Questions (FAQs). [Online]. Available: http://www.faa.gov/nextgen/implementation/programs/adsb/faq
[13] Eurocontol, ”Base of Aircraft Data (BADA)”, Available at http://www.eurocontrol.int/services/bada

[14] "GNU Radio - gnuradio.org" Available: http://gnuradio.org/redmine/projects/gnuradio/wiki
[15] "Ettus Research - Product Category". Available: https://www.ettus.com/product/category/USRP-Networked-Series
[16] "IQ Format". Available: https://gnuradio.org/redmine/projects/gnurad\\/wiki/UsrpFAQIntroFPGA
[17] W.J. Pan, Z.L. Feng, and Y. Wang, ADS-B Data Authentication Based on ECC and X.509 Certificate, Journal of Electronic Science and Technology, vol. 10, pp. 5155, Mar. 2012.
[18] D. McCallie, J. Butts, and R. Mills, "Security analysis of the ADS-B implementation in the next generation air transportation system", *International Journal of Critical Infrastructure Protection*, vol. 4, no. 2, pp. 78–87, Aug. 2011. Available: http://www.sciencedirect.com/science/article/pii/S1874548211000229
[19] C. Liu, A. Singhal, and D. Wijesekera. "A Logic-based Network Forensic Model For Evidence Analysis" Advances in Digital Forensics XI. Springer International Publishing, 2015. 129-145.
[20] Warner, Jon S., and Roger G. Johnston. "GPS spoofing countermeasures." Homeland Security Journal 25.2 (2003): 19-27.
[21] E. Piracci, G. Galati, and M. Pagnini. "ADS-B signals reception: a Software Defined Radio approach." Metrology for Aerospace (MetroAeroSpace), 2014.