MAVPro: ADS-B Message Verification for Aviation Security with Minimal Numbers of On-Ground Sensors

Ala' Darabseh
ala.darabseh@nyu.edu
New York University Abu Dhabi

Hoda AlKhzaimi
hoda.alkhzaimi@nyu.edu
New York University Abu Dhabi

Christina Pöpper
christina.poepper@nyu.edu
New York University Abu Dhabi

ABSTRACT

Automatic Dependent Surveillance Broadcast (ADS-B) centrally contributes to aircraft traffic control in the US and Europe since 2020. ADS-B messages contain information about aircraft location and tracks to provide better real-time traceability of aircraft in space. However, the lack of security mechanisms will be an obstacle for trusting the ADS-B technology. Thus, countermeasures should be integrated to secure the communication and evaluate the integrity and trustworthiness of received messages. In this paper, we design a message verification protocol called `MAVPro` to evaluate the trustworthiness of received ADS-B messages whose authenticity and integrity could otherwise not be verified. The main idea behind `MAVPro` is to compare location claims in received ADS-B messages with expected aircraft locations, which are computed using predicted trajectory information (e. g., velocity, elapsed time, aircraft acceleration, heading information) and a set of pre-trusted, continuously updated anchors. Our protocol is able to evaluate the trustworthiness of received messages if as little as *one* ADS-B receiver obtains a message — as opposed to four receivers required for using multilateration-based techniques to verify position claims. Thus we are able to considerably extend the coverage area where security checks can be applied compared to existing solutions. We evaluate `MAVPro` based on real-time data from the OpenSky network, analyze its performance, and verify its applicability to address ADS-B security concerns. `MAVPro` is backwards compatible and does not require changes to the ADS-B infrastructure.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**.

KEYWORDS

ADS-B, Aviation Security, Location Verification, Spoofing Detection

ACM Reference Format:

Ala' Darabseh, Hoda AlKhzaimi, and Christina Pöpper. 2020. MAVPro: ADS-B Message Verification for Aviation Security with Minimal Numbers of On-Ground Sensors. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20), July 8–10, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3395351.3399361

WiSec '20, July 8–10, 2020, Linz (Virtual Event), Austria

1 INTRODUCTION

Automatic Dependent Surveillance-Broadcast (ADS-B) [1] will centrally contribute to Aircraft Traffic Control (ATC) as of 2020 in the US and Europe in compliance with the DO-260B standard. ADS-B plays a significant role in enhancing ATC due to its benefits over traditional radar communication systems in terms of reliability and cost-effectiveness. Aircraft are configured by ADS-B Out and ADS-B In to send and receive ADS-B messages, respectively, that contain information on the location, velocity, and status of airplanes and are received by sensors in ground stations. ADS-B enables an aircraft to use GPS information specifying its position in real time and share this data with ATC for situational awareness, reflecting a consistent and precise view of the location of the aircraft with respect to the traffic surrounding it. However, the lack of security mechanisms caused by the open nature of ADS-B broadcast communication is an obstacle for trusting the system and for the widespread adoption of ADS-B technology. Various types of attacks can exploit the security weaknesses in ADS-B to gain control over the communicated messages and interfere with the system: signal jamming, message injection, message modification, and message deletion threaten the security of ADS-B [22]. The attacks are facilitated by the fact that ADS-B messages are broadcast as plaintexts without encryption and integrity protection. Thus countermeasures and mechanisms should be integrated to secure the communication and check the trustworthiness and integrity of received messages.

Security mechanisms can be divided into two approaches: *i*) Securing broadcast messages and *ii*) verifying received messages. Regarding *i*), cryptographic approaches have been discussed to secure ADS-B [26]. They promise to prevent unauthorized parties from accessing and tampering with ADS-B messages. This would, however, result in a closed system where keys would not be publicly disclosed and in the practical difficulty of keeping the keys confidential. The threat of key leaks and the burden of key management render symmetric-key encryption and integrity protection for ADS-B impractical. Asymmetric-key digital signatures, on the other hand, are the only viable cryptographic enhancement for ADS-B [26]. They, however, require changes to the protocol that are not backwards compatible (increased message sizes), plus they require high computational power and time for encryption and decryption operations. Moreover, since the open nature of the communication is a main feature of the ADS-B system, security solutions are needed that enable checks of trustworthiness of ADS-B messages without the need to change messages structures or contents.

Regarding message verification (approach *ii*), multilateration (MLAT) [14] is the de-facto technology to ensure the integrity of location claims in ADS-B messages based on determining the actual location of the message sender and comparing it to the claimed

location in the message. Checks based on MLAT require the broadcast message to be received by at least four sensors on the ground. However, the geographic area which is covered by four (or more) sensors based on data from OpenSky is small compared to the whole area covered by sensors on the ground [21]. This implies that MLAT can only be used for a small set of locations.

In this paper, we propose a new ADS-B message verification protocol called `MAVPro` to check the trustworthiness of ADS-B messages that are received by three, two, or only one sensor(s) on the ground. The main idea behind our approach is to use a rich set of observations to predict the expected location of aircraft and compare the measured value with the received value from the aircraft. When the difference is larger than a predefined threshold value, this is considered an indication of an attack. By using this approach, we significantly extend the applicability of the MLAT verification technique to broader coverage areas and enable its use for areas that were not applicable for verification at all before.

In more detail, `MAVPro` is derived by building a predefined base for all possible aircraft tracks, i. e., a set of all possible routes from the same source airport to the same destination airport for all trips between the airports that are covered by the ADS-B network. We use the real-world traffic from the OpenSky network [16, 24]. Furthermore, we define an anchor matrix which holds the latest trusted ADS-B message from each aircraft that is currently flying (with an active route in airspace). Then we use track predictions along with anchors for ADS-B message verification in the verification stage. The tracks are used to predict which tracks the received message's location belongs to. The performed security verification to distinguish legitimate ADS-B messages from spoofed or tampered messages includes location check, callsign check, and aircraft check.

The design of our proposed solution is non-trivial because we have to deal with the following constraints and challenges:

- **Lossy Data**: Due to the wireless channel, not all ADS-B messages are received. Thus, we only get the full routes that have messages from sources to destinations and discard the ones that send part of its route.
- **Imprecise Data**: ADS-B messages may be transmitted at unpredictable times, speeds, or they may contain inconsistent information (compared to previously received messages). This requires a filter mechanism to extract these messages from the data for consistency purposes.
- **Limited Sensor Coverage**: The receiving sensors on the ground only observe a part of the aircraft trajectory from the source to the destination airport. This means that every receiver has only a restricted view on the entire trajectory.

The main contributions of this work are:

(1) We design an ADS-B message verification protocol that can evaluate the trustworthiness of received messages. Compared to previous proposals, our protocol provides results even if only a single ADS-B receiver receives messages.
(2) We evaluate the proposed protocol with real-world data from the OpenSky network for a variety of attack scenarios and demonstrate the feasibility of our detection approach.
(3) We provide real-world evidence by observations relating to the coupling of aircraft routes and aircraft speed and rates.

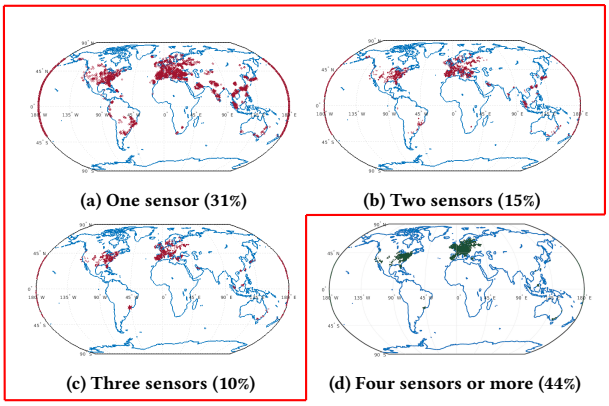


Figure 1: Global coverage areas of the deployed OpenSky sensors and the density of received ADS-B messages by sensors on ground, where the border around (a), (b), and (c) shows where MLAT *cannot* be applied, but `MAVPro` is applicable as message verification tool. The percentages are based on the received messages from all aircraft within a 5-min time interval on Feb 20, 2020.

2 PRELIMINARIES

The protocol we propose is based on fundamental systems and technologies: ADS-B, OpenSky, and MLAT. We provide an overview of these systems in this section. Moreover, our threat model and attack scenarios are described in detail.

2.1 ADS-B System

Automatic Dependent Surveillance-Broadcast (ADS-B) is a new Air Traffic Control (ATC) surveillance technology. It is part of the Next Generation Air Transportation System plan launched as a replacement of the radar systems. In this system, each aircraft obtains its location from navigation satellite systems (GPS) and broadcasts messages periodically to ATC stations on ground and to other surrounding aircraft to provide better location awareness and self-separation [22]. ADS-B consists of two services: ADS-B Out and ADS-B In. The former is used to transmit ADS-B messages to other ADS-B devices that are covered by the network and the latter is used to receive ADS-B messages that are transmitted by other ADS-B participants.

ADS-B messages contain information about the aircraft. Each message is structured into several data blocks and contains information about the aircraft ID (icao24), location (latitude, longitude, altitude), velocity, heading information, callsign (combination of airline and flight number), and an on_ground flag to indicate whether the aircraft is on the ground or not.

2.2 OpenSky

OpenSky [24] is a non-profit association and network of receivers. It is a collaborative research project aiming to improve security, reliability, and efficiency of air space usage by providing public access to real-world air traffic control data. The OpenSky network consists of a large number of connected sensors that are operated by volunteers, academic organizations, and industrial supporters; these sensors collect real-time traffic from ADS-B aircraft.

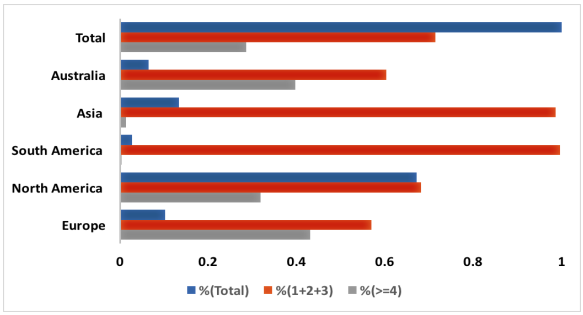


Figure 2: Relation between the total number of received ADS-B messages and the number of received messages by one/two/three and four+ sensors for different geographic areas. Ratios in red (1+2+3) indicate where our proposed technique will be applicable.

The current sensor coverage is predominant in Europe and North America. Figure 1 shows the coverage area of deployed sensors with the percentage of received ADS-B messages by one or more sensors. As the plots show, the percentage of messages that are received by four or more sensors on the ground represents only around 44% from all the messages, where the rest are received by three, two, or only one sensor(s). Moreover, Figure 2 shows percentages for the density of received messages by sensors across the world for the OpenSky network: More than twice as many messages ($\approx 70\,\%$) are received by less than four sensors and only $\approx 30\,\%$ by four or more.

For our purposes, we retrieve selected data from the publicly accessible OpenSky database. Our algorithm uses the data listed in Table 1.

2.3 Multilateration (MLAT)

Multilateration (MLAT), or hyperbolic positioning, is a co-operative independent surveillance technique that has been successfully deployed in military and civil contexts and the OpenSky network[24]. It is used to determine the location of a signal sender. Basically, if a message is received by four or more stations with known locations on the ground, they can make use of their location and the signal reception times (Time Difference of Arrival—TDOA) to find intersection points of all obtained 3D hyperboloid relations [14], which approximately reflects the position of the targeted sender.

One of the main advantages of MLAT is that it can verify the location claims in ADS-B messages without the need to change the existing infrastructure. However, it requires messages to be received by four or more sensors. Moreover, in practice, the accuracy of MLAT deteriorates over long distances and in noisy environments. For more open problems behind MLAT we refer to [22].

Table 1: ADS-B data used from the OpenSky database.

Data field	Meaning
icao24	Aircraft identification number (unique ID for each aircraft)
lat,lon,alt	Aircraft location at the time when the ADS-B message is transmitted in terms of latitude, longitude, and altitude
time	Time in seconds when the message was received
on_ground	Boolean value which indicates if the message is transmitted while the aircraft is on the ground
velocity	Aircraft speed (in miles/sec) at the time of sending the message.
callsign	Combination of the airline and flight number (unique value during aircraft itinerary)
heading	Direction of movement of aircraft (track angle as clockwise angle from the geographic north)
serials	All sensors that received the same message

2.4 Threat Model

We consider active attacks where the attacker manipulates ADS-B messages and behaves as legitimate sender and broadcasts messages in an attempt to modify existing traffic. For active attacks, we in particular consider the following attack strategies:

- **Packet Injection:** The attacker creates and transmits new packets that appear as legitimate ones or replays recorded messages. The attacker uses public knowledge about the ADS-B communication protocol and the message structure. E. g., in *aircraft hijacking* attacks, an attacker creates ghost aircraft by injecting new packets. Ghost aircraft traffic may affect the decisions of ATC and impact traffic collision.
- **Packet Modification:** The attacker uses existing packets and modifies their content. *Location spoofing* and *aircraft spoofing* attacks are two examples of packet modifications which we are considering in this work. We count as aircraft spoofing if any data fields in ADS-B messages are modified for existing icao24 values—location spoofing is the particular case where the position data is modified.

For evaluating the security of our proposed protocol, we consider and implement several attack scenarios that the attacker can follow. We classify the scenarios into three dimensions depending on how an attacker can actively interfere with transmitted ADS-B messages.

D1–Claimed message location: The attacker interferes with the location reports in ADS-B messages:

- *Off-track attack* (simple attacker): We assume the attacker does not act based on the details of the proposed approach. She will modify the location to another (e. g., random) location that is located out-bound the aircraft victim track.
- *On-track attack* (smart attacker): The attacker will replace the location within the message by another location within the range of the aircraft victim track. In other words, we assume the attacker is smart and knows how the proposed detection approach is working so she will try to modify the location in a way that is difficult to detect by the proposed approach. For example, in a Frog Boiling attack [2], the attacker applies small changes over time, within the threshold value, in a way that leads the aircraft to change its actual path after an extended time window.

D2–Claimed message icao24: The attacker may modify the possible values of a claimed aircraft icao24 in two ways:

- *In-airspace icao24*: The icao24 will be replaced by another one that is in airspace (it is currently flying) and is currently listed and tracked by the ATC system.
- *Off-airspace icao24*: The modified icao24 in this situation is not recognized by ATC as currently flying in airspace.

D3–Claimed message time: This dimension defines possible values of the claimed time. The attacker will delay the transmission of the packet for a while before transmitting it again.

Achieving such types of attacks requires having tools and devices to record and inject signals on the network—they are available for affordable prices on the market. Hence, simple (undirected) attacks can be easily achieved if these devices are available to the attacker. More sophisticated attacks, however, require knowledge about the exact location of victim aircraft at particular points in time in order to then obscure their claimed routes. But still such knowledge will not help a lot (more details in Section 5).

3 MAVPRO: ADS-B VERIFICATION PROTOCOL

Our goal is to design a protocol that can test the trustworthiness of received ADS-B messages and detect if they have been modified by an attacker before they reach the receiving sensors on the ground.

3.1 System Requirements and Notation

To achieve this goal, we define a set of requirements and classify them into functional, non-functional and security requirements.

Functional Requirement.

- **FR–**Message Authentication: The main purpose of our proposed protocol is to provide an algorithm and procedure to evaluate the trustworthiness of received ADS-B messages.

Non-Functional Requirements.

- **NR1–**Performance: Our protocol should be able to detect attacks in a fast way to satisfy real-time system needs.
- **NR2–**Compatibility: It should be configured and applied on the ADS-B network (in particular on the ATC backend) without any need to modify the existing infrastructure.
- **NR3–**Interoperability: It should be possible to integrate the protocol with other security techniques, like MLAT, to provide a comprehensive aviation security solution.

Security Requirements. Additionally, our protocol should work under attacks as defined in Section 2.4:

- **SR1–**Location Integrity: It should be able to check the correctness of claimed aircraft location and ensure that the ADS-B message carries the correct position.
- **SR2–**Source Integrity: It should be able to check aircraft identity and ensure that it is genuine.
- **SR3–**Data Integrity: It should be able to verify the consistency with previously received ADS-B messages.

Table 2 explains the main notation used to describe our protocol.

Table 2: Notation and Definitions

Notation	Meaning	Notation	Meaning
s	Source airport	acc	Aircraft acceleration
loc	Location	v_i	Initial velocity
msg	Message	v_f	Final velocity
$\mathbf{T}$	Track matrix	$\mathbf{SD}$	Source-destination matrix
T	Track	THV	Threshold value
$dist$	Distance	$Pr[\cdot]$	probability of …
$disp$	Displacement	n	Number of airports
$diff$	Difference	m	Number of aircraft
R	Route	$\mathbb{A}$	List of aircraft
r	Earth radius	$\mathbb{R}$	List of routes
$C(\cdot)$	claimed …	d	Destination airport
$E(\cdot)$	expected …	cs	Callsign
ic	icao24	$\mathbf{CS}$	Callsign matrix
AN	Anchor	$\mathbf{Anch}$	Anchor matrix

3.2 MAVPro Overview

Generally, using MLAT, at least four receiving sensors are required for verification. This means that we can apply MLAT only to a small geographic area that satisfies this condition. Our approach loosens this strict requirement by allowing verification checks if as little as one receiver only obtains a message. Thus, we are able to significantly increase the coverage area where the security checks can be applied. The main idea behind our protocol is based on comparing the claimed location in the ADS-B message with the expected location of the aircraft, which is computed using prediction of airport tracks, while resorting to a set of trusted values. More precisely, our protocol is based on processing three types of data:

(1) Information from ADS-B messages (location, time, direction, speed, callsign).
(2) Predicted trajectory information based on previously observed flights, called *tracks* in our protocol (see Sec. 3.3.1).
(3) A set of *anchors* as originally trusted observed (initialized) tokens retained on the ATC system. These anchors are verified points of trajectories. They are updated during the execution of our protocol (more details in Sec. 3.3.2). The anchors at the beginning are either initialized based on verifying the location of received message by MLAT or — if this is not available — are based on the assumption that no attack was happening at the time when the protocol was initialized.

In our approach, we distinguish *routes*, which individual airplanes take when flying from a source to a destination airport, from *tracks*, which represent 3-dimensional areas (coverage) between two airports that contain the individual routes that airplanes take.

Based on these components, we propose a technique to extend the trustworthiness of verifiable ADS-B messages received by four ADS-B receivers and propagate this level of trust to geographic areas where fewer than four receivers (up to only one receiver) obtain an ADS-B message. For testing the trustworthiness of messages received from different aircraft, we pursue a two-step approach:

(a) First (see Sec. 3.3), the offline (initialization) phase initializes the required data structure and is used to prepare the ground truth for later usage during the protocol execution. In particular, a comprehensive matrix of tracks **T** between source-destination airports is built as a base for later verification purposes; the entries of **T** consist of all possible routes of all aircraft $\mathbb{A}$ directly flying from the source s to the destination d. Also, an anchor matrix **Anch** for each route is instantiated; the entries of **Anch** are the latest received messages for each aircraft in airspace and are frequently updated based on the current flight data.
(b) Second (see Sec. 3.4), the actual protocol execution and verification process is run and uses the instantiated matrices. **Anch** is updated during the verification process.

3.3 Initial Setup

The initial data structures of MAVPro are built in two steps. We build the **T** matrix for all observed *tracks* in Step I, and we build the **Anch** matrix for all aircraft in airspace in Step II.

3.3.1 ***Step I (Build Tracks).*** To build **T**, we use information from the OpenSky sensors. First, based on one week of ADS-B traffic where the "onground" flag is set to true, we get the longitude and latitude information from aircraft on ground which roughly correspond to the airport locations.

We extract around 3400 airports from the collected traffic. In addition, for each airport location, we list all the aircraft that started or landed at this airport.

Second, we observe that there is a common list of aircraft and callsigns that are used frequently from the same airport s to the destination airport d. Thus, we get all aircraft and callsigns that these aircraft use to fly between pairs of airports. We organize the corresponding data in two matrices **SD** and **CS**, respectively, and use them for establishing **T**. More details are given below.

Source-destination Matrix (SD). We build the $(n \times n)$ source-destination matrix **SD** to store all intersections between airports (n is the total number of observed airports), where $\mathbf{SD}(s, d)$ contains all aircraft (icao24) that go from source s to destination d along with the probability that this route is taken:

$$\mathbf{SD}(s,d) := \begin{cases} \{ic, Pr[ic]\}_{sd}, & \text{if } s \neq d \\ 0, & \text{otherwise} \end{cases} =$$

$$\begin{Bmatrix} 0 & \{ic, Pr[ic]\}_{12} & .. & \{ic, Pr[ic]\}_{1n} \\ \{ic, Pr[ic]\}_{21} & 0 & : & : \\ : & : & 0 & : \\ \{ic, Pr[ic]\}_{n1} & .. & .. & 0 \end{Bmatrix},$$

$$Pr[ic]_{sd} = \frac{TripsCount(ic)_{sd}}{\sum_{i=1}^{m} TripsCount(\forall ic)_{sd}},$$

where $\{ic\}_{sd}$ is the list of aircraft that are flying from source to destination $(\mathbb{A}_s \cap \mathbb{A}_d)$[1] and m denotes the total number of aircraft of a specific track. For each entry in the matrix, there is a list of up to m possible tuples (aircraft) consisting of $\{ic_i, \Pr[ic_i]\}$ for $i \in \{1, m\}$. $\{ic\}_{sd}$ can be empty if there are no direct flights from s to d, and the probability of ic will be zero.

We assign a probability for each aircraft within each track based on its distinct number of trips within one week because we observe that not all aircraft fly at the same frequency between each pair of airports. We will use the probability of ic to derive a trust value for a claimed aircraft ID in the received ADS-B message. This way observing ADS-B messages from more frequently flying aircraft get assigned a higher trust score. For instance, based on the OpenSky data, we noticed that the number of ic that go from Frankfurt to Berlin is 154 within one week where 53 ics are unique.

Callsign Matrix (CS). We build the $(n \times n)$ callsign-matrix **CS** to store the extracted callsigns between airports, where $\mathbf{CS}(s, d)$ contains all callsigns of different aircraft that go from source airport s to destination airport d. Moreover, the *callsign* is also changing over different aircraft, thus we get the frequency of each *callsign* and store its value as follows:

$$\mathbf{CS}(s,d) := \begin{cases} \{cs, Pr[cs]\}_{sd}, & \text{if } s \neq d \\ 0, & \text{otherwise} \end{cases} =$$

$$\begin{Bmatrix} 0 & ... & ... & \{cs, Pr[cs]\}_{1n} \\ \{cs, Pr[cs]\}_{21} & 0 & : & : \\ : & : & 0 & : \\ \{cs, Pr[cs]\}_{n1} & ... & ... & 0 \end{Bmatrix},$$

where

$$Pr[cs]_{sd} = \frac{TripsCount(cs)_{sd}}{\sum_{i=1}^{m} TripsCount(\forall cs)_{sd}}.$$

For instance, for the same track from Frankfurt to Berlin we found that there were 30 unique callsigns out of 154 for the same week. The **CS** matrix will later be used in our verification (more details in Section 3.4).

Track Matrix (T). Eventually, we extract the full trajectory data (not only on-ground data) from OpenSky for each aircraft for the same week which used to get the list of airports for Europe, and classify it into several routes $\mathbb{R}$. Each aircraft can take a set of routes between different source-destination airports. The trajectory for an aircraft consists of the ic and the list of routes this aircraft can take, where each route R contains the recorded locations of this route

$$Trajectory(ic) := [icao, \mathbb{R}], \text{where } \mathbb{R} = \{R_{sd}\}$$
$$\text{and } R_{sd} = \{(lat, lon, alt)_s, \ldots, (lat, lon, alt)_d\}.$$

Finally, we store the routes between each airport pair in the $(n \times n)$ track matrix **T**. Each entry of **T** contains the track (set of routes for all aircraft from s to d) and the computed *coverage* for each track. The coverage of a track is the area that is bounded by all routes from s to d. The coverage will be used to identify which track and which route a received ADS-B message originates from (more details will follow in Section 3.4). In particular,

$$\mathbf{T}(s,d) := \begin{cases} Track_{sd}, & \text{if } s \neq d \\ 0, & \text{otherwise} \end{cases} =$$

$$\begin{Bmatrix} 0 & Track_{12} & ... & Track_{1n} \\ Track_{21} & 0 & : & : \\ : & : & 0 & : \\ Track_{n1} & ... & ... & 0 \end{Bmatrix},$$

where

$$Track_{sd} = [\mathbb{R}_{sd}, coverage(\mathbb{R})].$$

For our evaluation, we initialized all introduced variables and matrices with values from the one week of data that we downloaded from OpenSky. In a real-world ATC-based deployment, if changes on routes are proposed then **T** should be updated accordingly by the ATC. For the protocol to meet its objective, **T** should come from a reliable source and be part of the trusted computing base.

We next introduce an auxiliary data structure called *anchor* that we will use and update in our protocol as specified in Section 3.4.

3.3.2 ***Step II (Initiate Anchor per Route).*** For verification purposes our protocol will make use of *anchors* that we define as the latest trusted ADS-B message from aircraft that has been successfully mapped to a specific route and that has a source-destination airport related to it. For such a purpose, an $(n \times n)$ **Anch** matrix is build to store these anchors.

$$\mathbf{Anch}(s,d) := \begin{cases} \{R, anchor\}, & \text{if } s \neq d \\ 0, & \text{otherwise} \end{cases} =$$

[1]For test purposes we assume that aircraft appearing on the list of both source and destination airports means that there is a direct flight between these two airports.

$$\begin{Bmatrix} 0 & \{R, AN\}_{12} & ... & \{R, AN\}_{1n} \\ \{R, AN\}_{21} & 0 & : & : \\ : & : & 0 & : \\ \{R, AN\}_{n1} & ... & ... & 0 \end{Bmatrix}.$$

The initial value of the anchor should be assigned at the beginning of the aircraft itinerary (at an airport) and it is updated frequently over real-time traffic. MLAT can be used to verify the initially claimed location of an ADS-B message as described in Section 2.3 (assuming each airport is covered by at least four sensors). Once the location of the received ADS-B message is verified then we can make sure the remaining information is also not modified if all received broadcasted messages have identical information.

As we will explain in Section 3.4, when the aircraft is flying, the anchors get updated either following a successful verification check of received ADS-B messages in our proposed protocol or by using MLAT in case the message is received by a sufficient number of sensors. Figure 9 (Appendix) presents the process of initializing and updating the anchors during the flight track. Once the aircraft starts its trip, the anchor value will be assigned by a received ADS-B message that has passed the MLAT verification check. After that, and during the aircraft trip, `MAVPro` will be used to verify the received messages, and based on the verification test, the anchor value will be updated either by the received message in case it is trusted or by the computed coordinates in case there was something wrong detected. MLAT could be used also for verification in case the message is received by 4+ sensors. By doing so, `MAVPro` realizes the interoperability property since it does not enforce to change the current security checks, while it in fact complements them, i. e., `MAVPro` integrates with MLAT to improve the security of the evaluation process.

3.4 Verification Protocol

In this section, we describe our verification protocol and explain how our initialized matrices are used to evaluate the trustworthiness of received ADS-B messages. Our goal is to assign a trust score to the received message based on a number of weighted verification checks. The overall trust score of the message is computed based on all checks by the following formula:

$$Totalscore(msg) = \sum_{i=1}^{3} check(i)_{score} \cdot check(i)_{weight} \quad (1)$$

where $check(i)$ denotes location, aircraft, or callsign checks, based on the parameter i. $check(i)_{score} \in [0, 1]$ reflects the score of the check and $check(i)_{weight} \in [0, 1]$ is the weight of the check, where $\sum_{i=1}^{3} weight(i) = 1$, and $Totalscore(msg) \in [0, 1]$. A total score of 1 denotes a fully trusted message.

The verification process then consists of three correlated checks:

(1) *Location_check* is performed in three steps: *i*) lookup which track the *ic* in a received ADS-B message belongs to (in **T**(**s**, **d**)), *ii*) obtain the associated route and anchor of this route from the observed track (in **Anch**(**s**, **d**)), and *iii*) compute the expected location of aircraft and then verify the claimed position of the received ADS-B message by computing and comparing the claimed distance and expected distance that the aircraft has passed during the elapsed time.

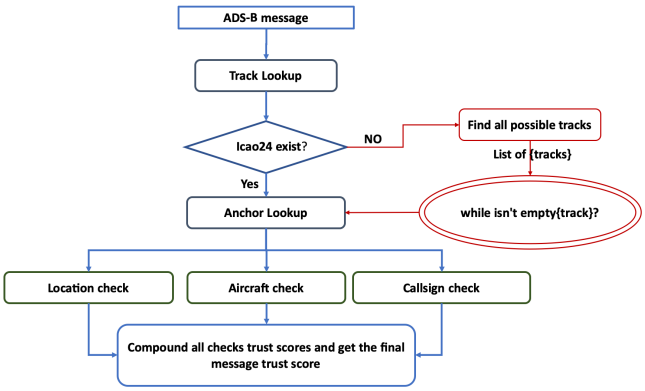


Figure 3: The overall workflow for the `MAVPro` protocol.

(2) *Aircraft_check* verifies whether the *ic* of the received ADS-B message exists in **SD**(**s**, **d**) and extracts its probability if it does exist in **SD**(**s**, **d**), or set to 0 if is *ic* does not exist .
(3) *Callsign_check* checks the callsign of the received ADS-B message for existence in **CS**(**s**, **d**) and extracts its probability.

Figure 3 shows the whole process of the `MAVPro` protocol and its checks in general. Now we will explain these checks in more detail.

Location_check: For the location check, each message is tested against its claimed location $C(lat, lon, alt)$ at time t with the following procedure (more details in Algorithm 3 in the Appendix):

(1) A list of all tracks to which the received message may belong is extracted. In other words, we find all possible tracks $\mathbf{T}_{sd}$ whose coverage area contains the claimed position from the ADS-B message. (Algorithm 1 in the Appendix illustrates the pseudo-code of this process.) If the received message contains an icao24 then getting the track to which this aircraft belongs is straightforward by mapping the icao24 with an existing one in **T**. If, on the other hand, the icao24 is omitted from the message then the *inpolygon()* function is used to get possible tracks of a claimed location; it checks if the received ADS-B message is inside or on the edge of the track polygonal region or not. If it is inside the polygon then this track should be considered in our next verification process:

$$msg_{tracks} = \{T_{sd}\}, \forall T \text{ where msg} \in \text{coverage}(T).$$

(2) After getting all possible tracks, we check all routes of each track in matrix **Anch** to identify which track this message belongs to and to map it to the appropriate route's anchor (Algorithm 2 in the Appendix explains the procedure of getting the anchor). If the returned anchor is empty then this means there is something wrong: either the message was actively modified or it is a false-positive message. In both cases there is no need to proceed to the following check and we evaluate the trustworthiness of this message at this stage. In case the attacker succeeds to modify the *ic* to the new one which exists in the extracted track list or modifies the message location to a new one that is still located in the same coverage area of the victim aircraft, then the next step is required to evaluate the message.

(3) We compute the expected displacement which defines how far an object moves vertically under the effect of gravity over a time period from one point to another. In our protocol this means the distance that the aircraft moves over the elapsed time from the anchor's location to the location of the newly received ADS-B message. This value is computed based on velocity, elapsed time, aircraft acceleration, and heading information from the claimed message and the anchor of the observed route by applying the following equation:

$$E(disp) = v_i \cdot \Delta t + \frac{1}{2} acc \cdot (\Delta t)^2, \text{where} \quad (2)$$

$\Delta t = (Time_{claimed} - Time_{anchor})$, $acc = \frac{v_f - v_i}{\Delta t}$; here v_i is the initial (anchor's) velocity, Δt the elapsed time, $Time_{claimed}$ the time when the claimed message is received by the sensor (the time inside the received ADS-B message that we want to verify), $Time_{anchor}$ the time when the anchor message is received (the time inside ADS-B message that we trusted previously and attached to the route), acc the aircraft route acceleration, which defines the rate at which aircraft change its velocity over the elapsed time, and v_f the final velocity.

(4) We compute the expected position coordinates:

$$E_{lat} = disp \cdot \cos(\alpha) + AN_{lat}$$
$$E_{lon} = disp \cdot \sin(\alpha) + AN_{lon}$$
$$E_{alt} = disp \cdot \tan(\alpha) + AN_{alt}$$

where $\alpha = AN_{heading}$.

(5) We compute the claimed distance using Euclidean distance, which defines the distance between the claimed position in the received ADS-B message and computed position ($[E_{lat}, E_{lon}, E_{alt}]$), by applying the following formula:

$$C(dist) = \|E(lat, lon, alt) - msg(lat, lon, alt)\| . \quad (3)$$

(6) We compute and compare the difference of distance between the claimed location from the received message and the expected location from the obtained coordinates:

$$diff = |C(dist) - E(disp)| . \quad (4)$$

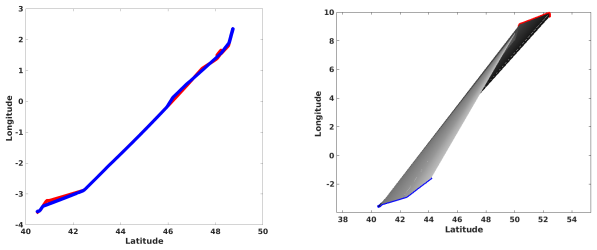
(7) Finally, we assign a trust score for the received message based on this check. The scores range from 0 to 1, where 1 reflects the high trusted message that is sent from an acceptable range, predefined threshold value (THV), and 0 is assigned to an untrusted message:

$$check(i)_{score} = \begin{cases} \frac{THV - diff}{THV}, & \text{if } 0 \leq diff < THV \\ 0, & \text{otherwise.} \end{cases}$$

We determine the THV by computing the $diff$ between $E(disp)$ and $C(dist)$ of routes for a number of tracks (Section 5).

Aircraft_check: As mentioned, each aircraft may have a certain probability to go from source s to destination d, thus, we built this check to evaluate the claimed aircraft ic and compare it with the expected aircraft list **SD**(s, d) that frequently travels from source s to destination d of the associated track. Based on observed results we can assign a trust score for the received ADS-B message by this test (the details are presented in Algorithm 4 in the Appendix):

$$check(i)_{score} = probability(msg.icao).$$



(a) same track (cm=0.13) **(b) different track (cm=17.8)**

Figure 4: Fréchet distance (coupling measure) of different routes.

Callsign_check: The probability of the callsign of the received ADS-B message is checked at this stage to assign a callsign trust score for this message. The **CS**$(\mathbf{s}, \mathbf{d})$ of the associated track is checked to know if the received callsign is one of the frequently recorded callsign that fly from source s to destination d. Based on the observed results we can assign a callsign trust score for the received ADS-B message. The score for each message is determined by the computed probability (as described in Algorithm 5 in the Appendix):

$$check(i)_{score} = probability(msg.callsign).$$

The obtained score from this check and Aircraft_check will not be assessed like the obtained one from Location_check since they support the proposed protocol by adding an extra check on the received message, while Location_check worth more in term of its complexity and procedure. Also, in case the message is received by new ic or new callsign then this message will get zero score, as we said even this case happens, still we cannot base on it by itself to assess the message, we have to consider all checks together.

Assigning weights for tests: We assign weights for all checks based on their difficulty and the procedure that they follow to check the trustscore, where $\sum_{i=1}^{3} Weight_{check(i)} = 1$. Since Location_check represents the core procedure and main idea behind our proposed protocol, it should weigh more than Aircraft_check and Callsign_check and thus we give it a higher weight (value range from $0 \rightarrow 1$, where 1 denotes high weight).

4 REAL-WORLD EVIDENCE

By working with OpenSky data, we observe patterns and evidenc that support the idea of `MAVPro` and provide real-world support for its main approach.

Coupling vs. Decoupling routes: `MAVPro` makes use of the idea that aircraft use similar routes to reach destination airport d from source airport s. To validate this claim, we measure the similarity between several routes from the same source s to the same destination d by using Fréchet distance [10] and the coupling measure (cm) algorithm where cm is zero if the two routes are identical and grows positively as the routes become more dissimilar. Figure 4 illustrates how the coupling measure between two routes from the same track is close to zero while there is no coupling between two routes from two separate tracks. We use this algorithm to make sure that the observed routes within each track belong to this track. Moreover, as the expected displacement is used to compute the

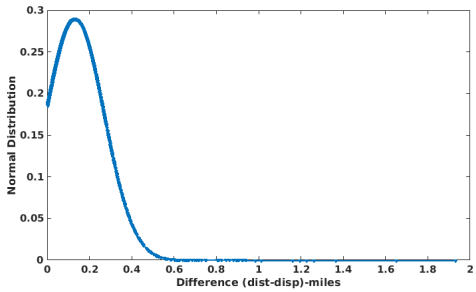


Figure 5: The distribution of the difference between the claimed distance and the expected displacement (THV).

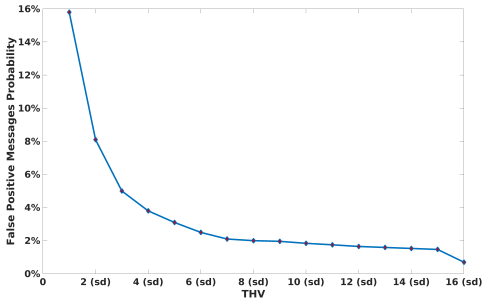


Figure 6: Percentage of false positive messages after applying MAVPro without any attack.

expected coordinates of the next movement of the aircraft, we compare the computed locations of one route with the received ones. The results in Figure 10 (Appendix) show a coupling between the two routes by Fréchet distance that is almost 0.5 cm.

Aircraft speed vs. Vertical rate: The speed of aircraft changes during their flights to reach the destination. Several factors influence the speed of aircraft at each coordinate on the track. From the obtained data we observe that all aircraft which go from the same source to the same destination follow similar speed pattern through their routes. Figure 11 (Appendix) represents how the aircraft velocity and the vertical rate change over time for three routes from the same track. It could be possible that aircraft follow different patterns over the year due to different reasons, i. e., climate change. But these patterns should be known for ATC. Accordingly, if any changes occur, the ATC should be aware of the expected behavior of aircraft. Thus, the aircraft patterns are restricted to set of known events and the proposed matrices should be updated frequently to reflect these events to keep the system robust and secure.

Claimed distance vs. Expected displacement: MAVPro computes the claimed distance and expected displacement (see Section 3). In an ideal scenario, their difference is zero, which means the aircraft follow a normal acceleration. However, in real-time systems, such an optimal scenario is not applicable, thus we measure the amount of variation in the difference over several routes. Figure 5 illustrates the results of the experiment and shows that most of difference values are distributed around 0.13 miles (which means MAVPro's accuracy is around 210 meters) difference, which means this is the minimum *THV* value that MAVPro could use.

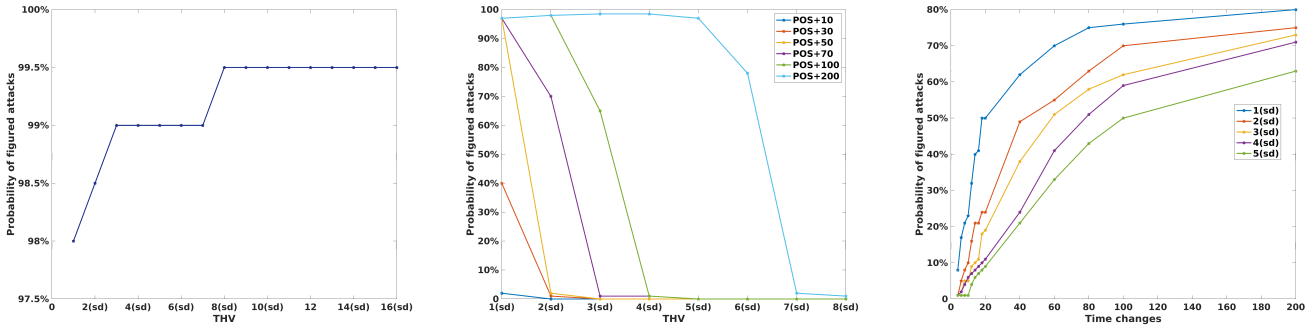
5 SECURITY EVALUATION

To evaluate the effectiveness of our proposed protocol, we define several attack scenarios to investigate the success of an attacker according to our threat model in Section 2.4. In this section, we test how MAVPro deals with these scenarios and check if it is able to detect these attacks. More precisely, we check the accuracy of the Location_check to compute the aircraft coordinates compared to the actual location and then use this derived information as a tool to check the trustworthiness of the received messages. The scenarios are based on ADS-B message spoofing. We built tests based on real-world data (100 route from different tracks) and crafted attacks through normal messages and checking if the tests were able to find them by measuring the success rate of catching these attacks.

One of the challenges we have to deal with is imprecise data. Thus, first of all, we try to capture this data, calling it "errors" (false-positive messages). We run the proposed protocol without any attacks, and then capture the points that result in errors caused by the large inconsistency of received ADS-B messages, e. g., suddenly receiving a message with 0 altitude after getting a message with 1000 m of altitude. Figure 6 shows the percentage of obtained errors over several *THV* values. As the plot shows, increasing the standard deviation (sd) reduces the number of experimental errors (the larger sd, the more false-positive messages will become normal ones). Selecting the particular sd to be used depends on the MAVPro deployers to choose at which level/distance the message location is considered valid and acceptable. At the same time, after roughly 5 sd, the number of errors remains almost constant, based on which we use 5 sd as threshold *THV* for MAVPro's verification tests. For test purposes, we filter the data from these error packets to give a precise indication for MAVPro's accuracy against the defined attacks.

D1–Claimed message location: We consider attacks where the attacker is able to modify the position coordinates by inserting new values, either as a simple attacker who spoofs ADS-B messages in a random way, or as a smart attacker who knows how the proposed protocol is working and tries to avoid the verification checks. Figure 7a reflects how MAVPro is able to capture the simple attacks with high probability over different *THV* values. MAVPro performs well in capturing the simple type of attacks where the modified location is chosen randomly in a way that makes it be far away from the actual aircraft route.

For the smart attacker, we test the effect of changing the received position by different in-advance position values within the range of the aircraft's actual location from the same trajectory where the attacker is able to predict the position of the aircraft. Figure 7b presents the probability of capturing the attack over several *THV* values. As shown, if the attacker changes the position *POS* by only 10 messages in advance, it will be hard to detect, while the probability increases whenever the attacker is farther away by 200 messages in advance. In all cases, 10 messages in advance means the attacker is still within the range of the victim aircraft and the impact of such spoofing is relatively small compared to further away claimed locations. Moreover, in the case of a frog



(a) simple attacker (b) smart attacker (c) message delay attacks

Figure 7: Probability of capturing different attacks over different THV values. For (a) position changes are selected randomly. For (b) position changes are given in number of messages (POS+X messages). For (c) the messages are delayed (playback) after several seconds.

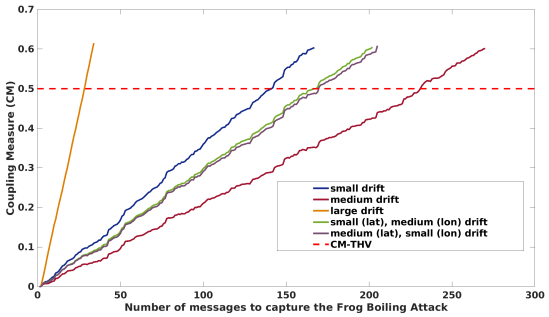


Figure 8: Fréchet distance (coupling measure) between the observed routes from the T matrix and the routes of the Frog Boiling attack across location drifts from the time the attack begins until the coupling measure reaches the THV.

boiling attack, Figure 8 shows how MAVPro is able to capture the attack after a few seconds when the route of the attacker is slowly drifting from the eligible route across several drift levels (i. e., drift levels range from small to large, where small means the attacker adds small changes into the actual aircraft position and large drift means large changes). The the required time that is required to capture it depends on the step that is chosen by the attacker; the larger it is the faster the attack will be captured. As a results, for the simple attacks a higher threshold value is better, while for the smart attacks a lower threshold value is better, more details on how to choose the THV value have been discussed in Section 6.

D2–Claimed message icao24: The attacker might spoof the icao24, changing it to another one. Capturing this type of attack is straightforward and can be achieved by checking the **Anch** matrix and see if the received icao24 exists or not. If it does not exist then the probability of capturing the attack is 100% since it is a simple look-up process. On the other hand, if the icao24 exists in the **Anch** matrix, then the Location_check which we follow by scenario (D1) will be applied for this aircraft with the anchor of claimed icao24. Since the location verification gives an estimation of the aircraft location, such type of attack can be easily detected (simple attack scenario) as long as it is out of the range of victim aircraft.

D3–Claimed message time: A classical time spoofing attack does not exist in the real system since the ATC assigns the received time of the message and the attacker has no control over this information. However, the attacker can take the message and resend it after a period of time in a replay attack. Thus, we evaluate the effect of sending the message with a delay on our approach. Figure 7c shows the probability of capturing messages delayed by different time periods across different standard deviation (sd) values. As shown, the probability of detection is getting higher with increased delay time and choosing lower (stricter) sd values.

6 DISCUSSION

MAVPro applications and effectiveness. Securing ADS-B communication becomes a must. MAVPro is proposed to be deployed by ATC central/fusion servers to verify the ADS-B messages. MAVPro tackles the discussed issues in existing solutions in the literature, more precisely for MLAT. Table 3 shows the advantages and effectiveness of MAVPro over MLAT. As illustrated, MAVPro succeeds to tackle the important coverage limitation of MLAT by increasing the coverage area by verifying messages that are received by three or less sensors. However, the accuracy of MLAT is better than MAVPro, MAVPros' accuracy is still acceptable within this range, also we should remember that MAVPro could be integrated with MLAT to verify the messages where MLAT is not applicable.

Dynamics of the track base. The track base **T** used in this paper is built for testing purposes to prove that this protocol can be helpful for the verification process. However, for the protocol to meet its objective, **T** should ideally come from a reliable source and more importantly be part of the trusted computing base. In case a new

Table 3: MLAT vs. MAVPro

	MLAT	MAVPro
Min number of sensors for verification	4	1
Relative coverage area	31%	100%
Location spoofing detection	Yes	Yes
icao24 spoofing detection	No	Yes
Message time (replay attack) detection	No	Yes
Affected by GDOP noise	Yes	No
Accuracy	30 m	200 m

route is introduced, then this new route should be reflected in the **T** matrix by ATC. The frequency of chaining the routes depends on several conditions, such as weather, season, aircraft congestion, and other. Such dynamics will not impact the effectiveness of MAVPro since at the beginning (before departure) the ATC will know which route the aircraft will follow and thus this selected route will be used later to verify the received messages.

THV selection. The results of Section 5 show the effect of changing THV on the MAVPro accuracy to capture the attacks. Selecting the THV depends on the application, ATC in this scenario. We present how the effectiveness of MAVPro to capture the attacks across several THV values. ATC can make a trade off and use the proper value for the verification process, i. e., relax the THV to capture only the claimed locations that are far away from the actual one, or tighten the THV value with the risk of getting more false-positive reports. The experiments provide a good illusion to decide which one is the proper value to be used.

MAVPro time complexity. One of the major concerns is how fast MAVPro is and if it is able to satisfy the real-time requirements. In this section, we analyze the time complexity of MAVPro. Basically, MAVPro has four main matrices: **T**, **SD**, **CS**, and **Anch** matrix. The running time is based on the selected data structure for those matrices. More precisely, selecting the list of tracks from **T** where the received message can belong to is the most critical decision. Once we select the list of tracks the rest will be straightforward. One simple and fast solution could be building a 3D sorted array whose dimensions are the longitude, latitude, and altitude of the sphere. Then this 3D array could be divided into several equal regions (small cubes) where each region should contain the list of tracks for this one. In this case, the process of getting the list of tracks will take $O(\log x)$, where x is the array dimension. The same applies to all lookup functions. Once we get the message track then the rest of verification checks can be done in a constant time $O(1)$.

7 RELATED WORK

ADS-B security associated problems and challenges have become an important topic [13, 15]. Security solutions and countermeasures have been discussed and proposed to address them and offer a secure system. Existing solutions are divided into two approaches: *i*) Securing broadcast messages and *ii*) verifying received messages.

Securing broadcast messages: Cryptography has been considered in [7, 20, 27] to address authentication in ADS-B and encryption of ADS-B messages [22]. A number of encryption solutions have been introduced [5]. However, the challenges of key disruption and management with the worldwide deployment of ADS-B make the symmetric cryptography based solutions hard options for securing the communication. In addition, Public Key Infrastructure (PKI) [3, 4] based on elliptic-curve cryptography and retroactive key publication solutions [22] are suggested as a way to publish the keys between network nodes. Nonetheless, such solutions require changes to the existing ADS-B infrastructure. Strohmeier et al. [21] discuss more challenges of adapting cryptography as a solution to secure ADS-B messages and how the lightweight encryption techniques [25] are analyzed to handle these challenges.

Verifying received messages: Signal verification mechanisms have been deployed successfully. MLAT is a popular surveillance technology that has been used to verify the sender location. Such solutions require 4+ receivers for verification (details in Sec. 2.3). [14, 17, 19] suggest using MLAT to verify ADS-B messages even without time synchronization between base stations [11]. However, Strohmeier et al. [21] show that the coverage area where the message can be received by four sensors is too small compared to the total ADS-B coverage area. Thus, a lightweight location verification approach [21] is proposed as an improvement on MLAT. It computes the TDoA for each position as a 2D grid and then compares the computed one with the received one. It increases the coverage area where the message verification can be achieved since the message should be received by only two sensors. Nonetheless, these solutions are not sufficient to verify the received messages.

Kacem et al. [9] design an intrusion detection system against malicious ADS-B by leveraging physical principles of aircraft motion. Any position report that falls outside a predicted 'safe zone' is considered anomalous. Their proposal enables each sending aircraft to issue secure ADS-B messages by pre-validating its GPS position before embedding it within the ADS-B message. This, however, incurs additional overhead on the message generation time and does not prevent a dishonest aircraft from cheating on its own location. Other solutions based on machine learning models [6, 23], Long Short-Term Memory (LSTMs) networks [8], Doppler effect [18], and timestamp [12] are also investigated to detect anomalies in communication flow through learning accident patterns.

8 CONCLUSION

We proposed a new verification protocol called MAVPro to assess the trustworthiness of ADS-B messages received by at least one sensor. Our proposal allows to significantly improve the coverage area where ADS-B messages can be verified compared to existing solutions. By relying on a set of pre-trusted and continuously updated anchors, MAVPro achieves this verification by comparing the claimed location in each received ADS-B message with the expected or predicted location of the aircraft, and subsequently assigning it a trust score. We point out that MAVPro does not require any changes to the current ADS-B message structure or infrastructure. Our security analysis for two types of adversaries (simple attacker spoofing messages randomly, smart attacker with knowledge about the verification protocol) has revealed that detecting a D2-type attack (on the icao24) is straightforward. Also, while a D1-type attack (on the message location) can be captured in 99.5 % of the cases when originating from a simple attacker, it is more challenging when dealing with a smart attacker. We still obtain high detection rates for the smart attacker for spoofed messages on the same track as long as the spoofed location is not too close to the real aircraft location (which would hardly classify as attack). D3-type attacks (on the time) can as well be effectively detected by MAVPro as long as the average time delay is higher than 60s with 1 standard deviation.

ACKNOWLEDGMENTS

The authors acknowledge assistance received by the ADEK Award for Research Excellence (AARE) 2017 under grant AARE17-236. This work is partially supported by the Center for Cyber Security at New York University Abu Dhabi (NYUAD).

REFERENCES

[1] W Blythe, H Anderson, and N King. 2011. ADS-B Implementation and Operations Guidance Document. *International Civil Aviation Organization. Asia and Pacific Ocean* (2011).

[2] Eric Chan-Tin, Daniel Feldman, Nicholas Hopper, and Yongdae Kim. 2009. The Frog-Boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinate Systems. In *Security and Privacy in Communication Networks*, Yan Chen, Tassos D. Dimitriou, and Jianying Zhou (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 448–458.

[3] Andrei Costin. 2012. Ghost is in the Air(traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices. Black Hat USA.

[4] Ziliang Feng, Weijun Pan, and Yang Wang. 2010. A data authentication solution of ADS-B system based on x. 509 certificate. In *27th International Congress of the Aeronautical Sciences, ICAS*. 1–6.

[5] Cindy Finke, Jonathan Butts, Robert Mills, and Michael Grimaila. 2013. Enhancing the security of aircraft surveillance in the next generation air traffic control system. *International Journal of Critical Infrastructure Protection* 6, 1 (2013), 3–11.

[6] Edan Habler and Asaf Shabtai. 2018. Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages. *Computers & Security* 78 (2018), 155–173.

[7] D. He, N. Kumar, K. R. Choo, and W. Wu. 2017. Efficient Hierarchical Identity-Based Signature With Batch Verification for Automatic Dependent Surveillance-Broadcast System. *IEEE Transactions on Information Forensics and Security* 12, 2 (Feb 2017), 454–464. https://doi.org/10.1109/TIFS.2016.2622682

[8] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 387–395.

[9] Thabet Kacem, Duminda Wijesekera, Paulo Costa, and Alexandre Barreto. 2016. An ADS-B intrusion detection system. In *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 544–551.

[10] Richard J Kenefic. 2014. Track clustering using Fréchet distance and minimum description length. *Journal of Aerospace Information Systems* 11, 8 (2014), 512–524.

[11] Sangdeok Kim and Jong-Wha Chong. 2015. An efficient TDOA-based localization algorithm without synchronization between base stations. *International Journal of Distributed Sensor Networks* 11, 9 (2015), 832351.

[12] Y. Kim, J. Jo, and S. Lee. 2017. ADS-B vulnerabilities and a security solution with a timestamp. *IEEE Aerospace and Electronic Systems Magazine* 32, 11 (November 2017), 52–61. https://doi.org/10.1109/MAES.2018.160234

[13] W. Li and P. Kamal. 2011. Integrated aviation security for defense-in-depth of next generation air transportation system. In *2011 IEEE International Conference on Technologies for Homeland Security (HST)*. 136–142. https://doi.org/10.1109/THS.2011.6107860

[14] Ivan A. Mantilla-Gaviria, Mauro Leonardi, Gaspare Galati, and Juan V. Balbastre-Tejedor. 2015. Localization algorithms for multilateration (MLAT) systems in airport surface surveillance. *Signal, Image and Video Processing* 9, 7 (01 Oct 2015), 1549–1558. https://doi.org/10.1007/s11760-013-0608-1

[15] Donald McCallie, Jonathan Butts, and Robert Mills. 2011. Security analysis of the ADS-B implementation in the next generation air transportation system. *International Journal of Critical Infrastructure Protection* 4, 2 (2011), 78 – 87. https://doi.org/10.1016/j.ijcip.2011.06.001

[16] Xavier Olive and Jérôme Morio. 2019. Trajectory clustering of air traffic flows around airports. *Aerospace Science and Technology* 84 (2019), 776–781.

[17] Matthias Schäfer, Vincent Lenders, and Jens Schmitt. 2015. Secure track verification. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 199–213.

[18] Matthias Schäfer, Patrick Leu, Vincent Lenders, and Jens Schmitt. 2016. Secure motion verification using the doppler effect. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 135–145.

[19] A. Smith, R. Cassell, T. Breen, R. Hulstrom, and C. Evers. 2006. Methods to Provide System-Wide ADS-B Back-Up, Validation and Security. In *2006 ieee/aiaa 25TH Digital Avionics Systems Conference*. 1–7. https://doi.org/10.1109/DASC.2006.313681

[20] Matthew Smith, Daniel Moser, Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. 2017. Economy Class Crypto: Exploring Weak Cipher Usage in Avionic Communications via ACARS. In *Financial Cryptography and Data Security*, Aggelos Kiayias (Ed.). Springer International Publishing, Cham, 285–301.

[21] Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. 2015. Lightweight Location Verification in Air Traffic Surveillance Networks. In *CPSS@ASIACSS*.

[22] M. Strohmeier, V. Lenders, and I. Martinovic. 2015. On the Security of the Automatic Dependent Surveillance-Broadcast Protocol. *IEEE Communications Surveys Tutorials* 17, 2 (Secondquarter 2015), 1066–1087. https://doi.org/10.1109/COMST.2014.2365951

[23] Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. 2016. A localization approach for crowdsourced air traffic communication networks. *arXiv preprint arXiv:1610.06754* (2016).

[24] Martin Strohmeier, Ivan Martinovic, Markus Fuchs, Matthias Schäfer, and Vincent Lenders. 2015. Opensky: A swiss army knife for air traffic security research. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. IEEE, 4A1–1.

[25] M. Strohmeier, M. Schafer, V. Lenders, and I. Martinovic. 2014. Realities and challenges of nextgen air traffic management: the case of ADS-B. *IEEE Communications Magazine* 52, 5 (May 2014), 111–118. https://doi.org/10.1109/MCOM.2014.6815901

[26] Kyle D Wesson, Todd E Humphreys, and Brian L Evans. 2014. Can cryptography secure next generation air traffic surveillance? *IEEE Security and Privacy Magazine* (2014).

[27] A. Yang, X. Tan, J. Baek, and D. S. Wong. 2017. A New ADS-B Authentication Framework Based on Efficient Hierarchical Identity-Based Signature with Batch Verification. *IEEE Transactions on Services Computing* 10, 2 (March 2017), 165–175. https://doi.org/10.1109/TSC.2015.2459709

APPENDIX

A SUPPORTING FIGURES AND PLOTS

Figure 9 presents the process of initializing and updating the anchors before and during the flight track from the source s to destination d.

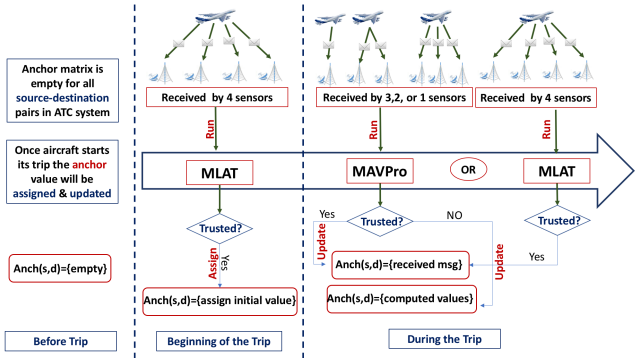


Figure 9: The steps of the initial setup of our proposed protocol. First, the *anchor* value is assigned/initialized by MLAT, then it is updated frequently by our proposed protocol if the messages are received by less than four sensors and by MLAT if they are received by 4+ sensors.

Figure 10 shows the coupling as Fréchet distance, between the received route and the computed route by MAVPro.

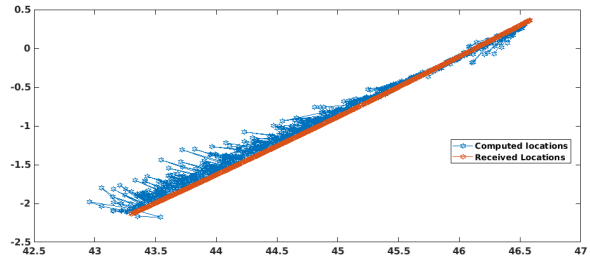
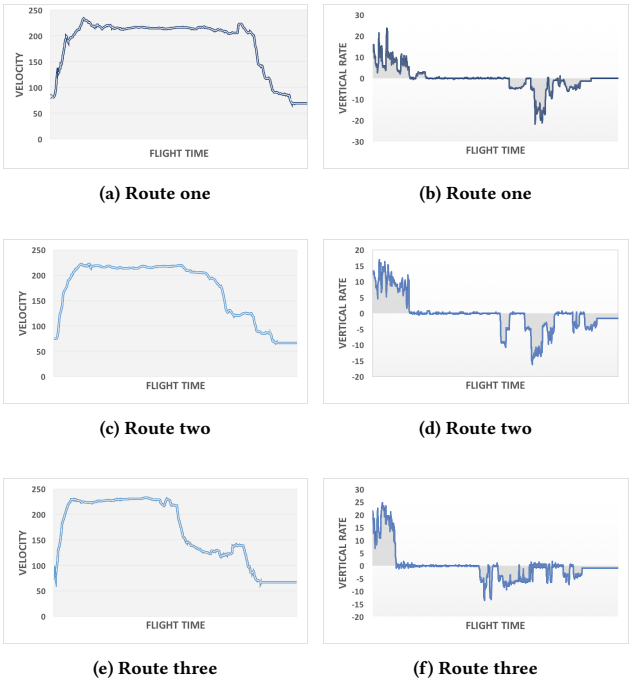


Figure 10: Comparison between the received route and the computed route based on the expected displacement. The Fréchet distance is almost equal to 0.5 cm (coupling measure).

Figure 11 represents how the aircraft velocity and the vertical rate change over time for three sample routes from the same track.



(a) Route one (b) Route one

(c) Route two (d) Route two

(e) Route three (f) Route three

Figure 11: Velocity and vertical rates of three routes from the same source to destination airports.

B PSEUDO CODE ALGORITHMS

The pseudo codes of all MAVPro procedures and checks are provided in this section: Algorithm 1 to find the track, Algorithm 2 to find the anchor, Algorithm 3 to evaluate the location, and Algorithms 4 and 5 to check the aircraft and callsign of the received ADS-B message, respectively.

Algorithm 1 Track_Lookup

procedure FIND_T(msg, T)
  $C\_loc \leftarrow$ msg(lat,lon,alt)
  $C\_icao \leftarrow$ msg.icao
  $tracks \leftarrow$ null
  if $IsEmpty(C\_icao)$ **then**
    for $\forall T \in$ T **do**
      if $inpolygon(C\_loc, T)$ **then**
        $tracks \leftarrow T$
  else
    $tracks \leftarrow \mathrm{T}_{C\_icao}$
  return $tracks$

Algorithm 2 Anchor_Lookup

procedure FIND_AN(msg, $tracks$, Anch)
  $C\_icao \leftarrow$ msg.icao
  $C\_cs \leftarrow$ msg.callsign
  $anchor \leftarrow$ null
  for $\forall T \in tracks$ **do**
    if $C\_icao \exists$ Anch(T) & $C\_cs \exists$ Anch(T) **then**
      $anchor \leftarrow$ Anch$(T).R.anchor$
      $msgT \leftarrow T$
  if $IsEmpty(anchor)$ **then**
    $GetAlarm$
    $break$
  return $anchor$, $msgT$

Algorithm 3 Location_check

procedure CHECK_LOC(msg, $anchor$, THV)
  $C\_lat \leftarrow$ msg.lat
  $C\_lon \leftarrow$ msg.lon
  $C\_t \leftarrow$ msg.time
  $C\_v \leftarrow$ msg.velocity
  $AN\_lat \leftarrow$ anchor.lat
  $AN\_lon \leftarrow$ anchor.lon
  $AN\_t \leftarrow$ anchor.time
  $AN\_v \leftarrow$ anchor.velocity
  $\Delta t = C\_t - AN\_t$
  $\alpha =\leftarrow$ anchor.heading
  $acc = \dfrac{C\_v - AN\_v}{\Delta t}$
  $E(disp) = AN\_v \cdot \Delta t + \dfrac{1}{2} acc \cdot (\Delta t)^2$
  $E_{lat} = disp \cdot \cos(\alpha) + AN_{lat}$
  $E_{lon} = disp \cdot \sin(\alpha) + AN_{lon}$
  $E_{alt} = disp \cdot \tan(\alpha) + AN_{alt}$
  $C(dist) = \|E(lat, lon, alt) - msg(lat, lon, alt)\|$
  $diff = |C(dist) - E(disp)|$
  if $0 < diff < THV$ **then**
    $check(i)_{score} \leftarrow \dfrac{THV - diff}{THV}$
  return $check(i)_{score}$

Algorithm 4 Aircraft_check

procedure CHECK_AIR(msg, $msgT$, SD)
  $C\_icao \leftarrow$ msg.icao
  for $\forall icao \in$ SD$(msgT)$ **do**
    if $C\_icao = icao$ **then**
      $check(i)_{score} \leftarrow$ SD$(msgT).Pr[C_icao]$
  return $check(i)_{score}$

Algorithm 5 Callsign_check

procedure CHECK_CS(msg, $msgT$, CS)
  $C\_cs \leftarrow$ msg.cs
  for $\forall cs \in$ CS$(msgT)$ **do**
    if $C\_cs = cs$ **then**
      $check(i)_{score} \leftarrow$ CS$(msgT).Pr[C_cs]$
  return $check(i)_{score}$