Test Data Generation for False Data Injection Attack Testing in Air Traffic Surveillance

Aymeric Cretin
FEMTO-ST Institute, UBFC, CNRS
Besançon, FRANCE
aymeric.cretin@femto-st.fr

Alexandre Vernotte
FEMTO-ST Institute, UBFC, CNRS
Besançon, FRANCE
alexandre.vernotte@femto-st.fr

Antoine Chevrot
FEMTO-ST Institute, UBFC, CNRS
Besançon, FRANCE
antoine.chevrot@femto-st.fr

Fabien Peureux
FEMTO-ST Institute, UBFC, CNRS
Besançon, FRANCE
fabien.peureux@femto-st.fr

Bruno Legeard
FEMTO-ST Institute, UBFC, CNRS
Smartesting Solutions and Services
Besançon, FRANCE
bruno.legeard@femto-st.fr

***Abstract*—The ADS-B — Automatic Dependent Surveillance Broadcast — technology requires aircraft to broadcast their position and velocity periodically. The protocol was not specified with cyber security in minds and therefore provides no encryption nor identification. These issues, coupled with the reliance on aircraft to communicate on their status, expose air transport to new cyber security threats, and especially to FDIAs — False Data Injection Attacks — where an attacker modifies, blocks, or emits fake ADS-B messages to dupe controllers and surveillance systems. This paper is part of an ongoing research initiative toward FDIA test generation intended to improve the detection capabilities of surveillance systems. It focuses on the mechanisms used to alter existing legitimate ADS-B recordings *as if* an attacker had tempered with the communication flow. We propose a set of alteration algorithms covering the taxonomy of FDIA attacks for ADS-B previously defined in the literature. We experiment this approach by generating test data for an AI-based FDIA detection system [8]. Experimental results show that the proposed approach is straightforward to generate attack situations and provides a efficient way to easily generate sophisticated alterations that were not picked up by the detection system.**

***Keywords*—AI testing, test data generation, false data injection attacks, air surveillance.**

I. Introduction

The world of air transport is facing new challenges as the traffic load keeps growing steadily[1]. With an increasingly congested airspace, Air Traffic Control (ATC) needs surveillance technologies that can support the increasing constraints in terms of simultaneously handled aircraft as well as positioning accuracy. The Automatic Dependent Surveillance-Broadcast (ADS-B) protocol is currently being rolled out in an effort to reduce costs and improve aircraft position accuracy [21]. Communication via ADS-B consists of participants broadcasting their current position and other information periodically (a.k.a. a beacon) in an unencrypted message [14].

However, the fundamental technological changes in ATC needed to support increasing traffic, which consist of a shift from independent and non-cooperative surveillance technologies to dependent and cooperative ones, have rendered the ATC community unable to foresee the new emerging threats related to cyber security. The ADS-B protocol was not designed with security in mind since securing ADS-B communication was not a high priority during its specification. As a consequence, anyone with the right equipment can listen and emit freely. For instance, there is a market for equipping private aircraft with ADS-B transponders using a smartphone and a dongle[2]. The complete freedom of ADS-B both in emission and reception makes it vulnerable to spoofing, and more precisely to a class of attack called *False Data Injection Attack* (FDIA) which purpose is to covertly emit meticulously-crafted fake surveillance messages in order to dupe ATC controllers into thinking, for instance, that some aircraft is dangerously approaching a building, while in reality it is flying normally.

Although it is not the only means for Aircraft tracking — other protocols are also used in conjunction of radar technologies —, ADS-B plays a central role in the current shift regarding aircraft position, collected from radar systems to GNSS [4]. It is so central in fact that it has become a mandatory brick of air traffic surveillance and any observed problem will ground all aircraft in the area[3]. Hence there is a strong need to improve its overall security. Nevertheless, because of the inherent properties of the protocol, current solutions for securing ADS-B communications are only partial or involve an unbearable cost [22].

Instead, ATC should be made more secure by strengthening its logic, but the ability to differentiate attacks from real critical situations still remains a challenge that is being tackled by the ATC community. Indeed, multiple integrity checks or detection approaches are rolled out or under study. These solutions are new and need to be deeply tested. To the best of our knowledge, there is no direct previously published work addressing the assessment of the efficiency of this kind of security. The most related work is a framework proposed by Barreto et. al. [3], which allows for the simulation of an

[1] http://www.boeing.com/commercial/market/current-market-outlook-2017/

[2] https://www.uavionix.com/products/skybeacon/

[3] https://hackaday.com/2019/06/09/gps-and-ads-b-problems-cause-cancelled-flights/

978-1-7281-1075-2/20/$31.00 ©2020 IEEE
DOI 10.1109/ICSTW50294.2020.00034

entire air traffic environment (aircraft, radio relay, network infrastructure, etc.). They perform FDIAs in the simulated environment to evaluate the attack impact on each component. The approach provides substantial information on how components react to an FDIA. Still, implementing all network behaviours of a scenario requires a lot of effort and the approach does not allow for the concretization of the simulated attacks on actual ATC software.

The contribution of this paper is part of an ongoing research initiative about FDIA testing that ultimately led to the creation of a testing framework called *FDI-T* [6] (False Data Injection Testing framework). This framework allows ATC experts to simulate FDIAs by creating, modifying and deleting recorded legitimate ADS-B messages in a fruitful, scalable and productive manner. The generated test scenarios can be executed on ATC systems in order to evaluate their resilience against potential security and safety anomalies related to FDIAs. They can also be used as test data to evaluate machine learning based detection systems, and improve their detection rate. The paper focuses on the alteration mechanisms that take as input one or several air traffic recordings and a list of alteration directives in order to simulate the presence of FDIAs in air traffic surveillance communications.

The paper is organized as follows. Section II briefly provides a basis for common concepts and current practices regarding air traffic surveillance as well as the key aspects to test such systems, especially regarding test scenarios based on FDIAs. Afterwards, Sect. III introduces the proposed approach to perform FDIAs on ADS-B messages and describes the automated process supporting the method. Section IV details the various algorithms used to apply alteration directives on ADS-B messages and explains how this generation is automated. Section V demonstrates the ability of the proposed approach to generate test data to assess the detection capabilities of a machine learning technique from the literature [8]. Finally, Sect. VI recaps the major contributions of the paper and suggests directions for future work.

II. Background and Research Objective

A. ADS-B Protocol

Communication via ADS-B consists of aircraft using a Global Navigation Satellite System (GNSS) to determine their position and broadcast it periodically without solicitation (a.k.a beacons or squitters), along with other information obtained from on-board systems such as altitude, ground speed, aircraft identity, heading, etc. Ground stations pick up on the squitters, process them and send the information out to the ATC system. ADS-B is therefore a cooperative (aircraft need a transponder) and dependent (on aircraft data) surveillance technology.It means for instance that ground stations with antennas positioned at the right angle and direction can receive position information. Aircraft can now receive squitters from other aircraft, which notably improves cockpit situational awareness as well as collision avoidance. For instance, the second generation of the Traffic Alert and Collision Avoidance System (TCAS-II) is based on ADS-B data.

Its introduction also provides controllers with improved situational awareness of aircraft positions in En-Route and TMA (Terminal Control Area) airspaces, and especially in NRAs (Non Radar Areas). It theoretically gives the possibility of applying much smaller separation minima than what is presently used with current procedures (Procedural Separation) [1].

ADS-B has the advantage of being a much cheaper technology as it has minimal infrastructure requirements. For instance, an ADS-B receiver can easily be bought online for a few hundreds euros[4]. ADS-B has a high accuracy and update rate, with a small latency. The major drawback of the technology lies in its lack of encryption and authentication, which is discussed in the following section.

B. False Data Injection Attacks

Extensive research can be found in the literature discussing the cyber security of surveillance communications [17], [22], [23], [25]. The progressive shift from independent and non-cooperative technologies (PSR) to dependent and cooperative technologies (ADS-B) has created a strong reliance on external entities (aircraft, GNSS) to estimate aircraft state. This reliance has brought alarming cyber security issues.

FDIAs were initially introduced in the domain of wireless sensor networks [11]. A wireless sensor network is composed of a set of nodes (i.e. sensors) that send data reports to one or several ground stations. Ground stations process the reports to reach a consensus about the current state of the monitored system. A typical scenario consists of an attacker who first penetrates the sensor network, usually by compromising one or several nodes, and thereafter injects false data reports to be delivered to the base stations. This can lead to the production of false alarms, the waste of valuable network resources, or even physical damage. Active research regarding FDIAs has been conducted in the power sector, mainly against smart grid state estimators [10]. It shows that these attacks may lead to power blackouts but can also disrupt electricity markets, despite several integrity checks.

FDIAs also exist in the domain of air traffic surveillance: because surveillance relies on the information provided by aircraft's transponders to ground stations, aircraft transponders are equivalent to nodes from a wireless network and ground stations are equivalent to base stations (but there is no real effort in the ATC domain to penetrate the sensor network since all communications are unauthenticated and in clear text). Still, performing FDIAs on surveillance communications is no simple task: it requires a deep understanding of the system, its protocol(s) and its logic, to covertly alter (by injecting falsified squitters and deleting genuine ones) the consensus reached by the ground station regarding the air situation picture. These attacks are much more complex to achieve than e.g., jamming, because the logic of the communication flow must be preserved and the falsified data must go unnoticed.

[4]https://flywithscout.com

Considering the attacker has the necessary equipment, he can perform three malicious basic operations [13]:

(i) *Message injection* which consists of emitting non-legitimate but well-formed ADS-B messages.
(ii) *Message deletion* which consists of physically deleting targeted legitimate messages using destructive or constructive interference (message deletion may not be mistaken for jamming as jamming blocks all communications while message deletion drops selected messages only).
(iii) *Message modification* which consists of modifying targeted legitimate messages using overshadowing, bit-flipping or combinations of message deletion or injection.

The above three techniques allow for the execution of several attack scenarios that can be categorized as follows [17]:

- **Ghost Aircraft Injection.** The goal is to create a non-existing aircraft by broadcasting fake ADS-B messages on the communication channel.
- **Ghost Aircraft Flooding.** This attack is similar to the first one but consists of injecting multiple aircraft simultaneously to saturate the air situation picture and generate a denial of service of the controller's surveillance system.
- **Virtual Trajectory Modification.** Using either message modification or a combination of message injection and deletion, this attack aims to modify the aircraft perceived trajectory.
- **False Alarm Attack.** Based on the same techniques as the previous attack, the goal is to modify the messages of an aircraft in order to indicate a fake alarm. A typical example would be modifying the squawk code to 7500, indicating the aircraft has been hijacked.
- **Aircraft Disappearance.** Deleting all messages emitted by an aircraft can lead to the failure of collision avoidance systems and ground sensors confusion. It could also force the aircraft under attack to land for safety check.
- **Aircraft Spoofing.** This attack consists of spoofing the ICAO number of an aircraft through message deletion and injection. This could allow an enemy aircraft to pass for a friendly one and reduce causes for alarm.

One can sense the potential for disaster if one of these attack scenarios was to be executed successfully. It is of the utmost importance that none of the scenarios represent a real threat to such a critical infrastructure with human lives on the line. However, because of the inherent properties of the ADS-B protocol, current solutions for securing ADS-B communications are only partial or involve an unbearable cost [22]. Therefore, ATC systems must become more robust against FDIAs, i.e. being capable of automatically detecting any tempering with the surveillance communication flow.

C. FDIA detection using Machine Learning

Machine Learning techniques have significantly contributed to improving the detection of anomalies in many domains. There are few of many applications of irregularity disclosure using Machine Learning techniques, e.g., detecting abnormal cells or markers on medical imagery [16], [18], detecting intruders on a network using logging information [24], or detecting traffic inconsistencies in bus trajectory data [9].

Regarding anomaly detection in ADS-B data, several experiments can be found in the literature. Using signal discrepancies through different receiver sensors [12] shows the ease of access to physical data through personal antennas. It can be noted that platforms, such as Opensky-Network [19], helped the training of models using the logical aspects of ADS-B [8], [20].

However, a major drawback of using Machine Learning techniques for anomaly detection is the lack of pre-existing altered data, whether to use it for training or testing. While contributions exist using unsupervised models on unbalanced dataset with good detection scores [5], [15], supervised learning will often yield underperforming results. Hence, there is a need of harmonizing dataset to explore new models and improve existing ones. Alteration mechanisms, as proposed in this paper, would enable such goals on ADS-B data.

D. Research Objective and Questions

It is critical to make sure that FDIA detection systems are properly and thoroughly tested. Such a testing campaign needs large sets of test data in the form of air traffic recordings, where ideally half of the recordings present an anomaly, i.e. an FDIA that should be detected. The creation of FDIA test scenarios in the ATC domain can be very complex. This requires at the same time altering or creating false data (e.g., creating ghost aircraft) while ensuring the consistency of all data. It is also necessary to ensure the widest possible coverage w.r.t. the taxonomy of attacks previously given. The creation of a set of algorithms capable of generating FDIA test scenarios thus aims to considerably increase the feasibility of in-depth testing, especially regaring the efficiency and effectiveness of FDIA detection systems. Based on these observations, we have defined the following research objective:

Create a set of alteration algorithms to automate the production of synthetic FDIA scenarios for the testing of supervised learning-based detection models.

From this research objective, we have identified two research questions expressed and developed below. We refer to these questions in Sect. V when evaluating the approach.

RQ1 **To what extent is it possible to automatically generate FDIA scenarios to cover the ADS-B attack taxonomy?**

The generated scenarios shall be computed algorithmically and must integrate protocol and specific application domain issues to ensure its relevance and realism.

RQ2 **To what extent is the approach relevant to generate synthetic data for testing an AI model detecting abnormalities in ADS-B data?**

In other cyber security domains (typically in IT), huge historical databases of past attacks and demonstration of vulnerability exploitation are available(e.g., the National Vulnerability Database of the NIST[5]). Within ATC domain, there is no

[5]https://nvd.nist.gov/

history of FDIAs reported to the general public. Therefore our approach aims to generate complex FDIA scenarios that could contribute to more efficient ML-based detection systems.

III. Alteration Process Overview

The proposed approach aims to generate altered versions of an original recording in order to feed FDIA detection systems to abnormal situations. A recording is a set of stored messages emitted by real aircraft using an antenna or created from data collected by online providers (e.g., OpenSky Network[6]. Figure 1 depicts the alteration process enabling to produce an altered recording from a set of original recordings and alteration directives. The five items of the process are detailed below.

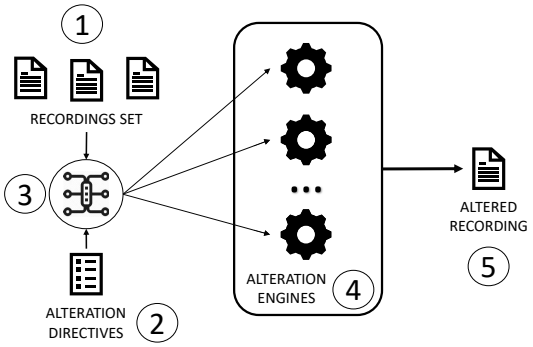

Fig. 1: Alteration process overview

① **A set of original recordings** contains recordings that can be altered or used as a specific parameter of an alteration directive (e.g., a replay attack needs one recording to perform the alteration, and one source recording to be injected in the first one. The source recording is specified in the directive).
② **A list of alteration directives** defines the specification of the test scenario. Indeed, the alteration process relies on a list of alteration directives to be performed on the original recording. An alteration directive, depending on the type of alteration, contains several parameters such as a time window (*when* the alteration takes place in the recording), a list of targeted aircraft, a list of way-points in case of virtual trajectory modification for instance, etc.
③ **The switcher** iterates over the list of alteration directives. For each one, it calls the corresponding alteration engine based on the type of the alteration specified in the processed directive.
④ **The alteration engines** are used to generate altered recordings from original ones. There is an alteration engine for each type of alteration. The engines are build according to the structure of the method they implement (see Sect. IV) where all the implemented algorithms are discussed.
⑤ **An altered recording** is outputted from the process. Its closeness to the original recording depends on the number of targeted aircraft and on the scope defined in the alteration directives.

[6]https://opensky-network.org/

A generated test scenario is formalized as $S = (r, D)$ with r as the targeted recording picked among the set of recordings, and D as a list of alteration directives. Regarding the format of messages taken as input, our prototype accepts surveillance data in the SBS format[7]. SBS messages contain twenty two fields. The first ten fields contain the reception time stamp and static properties about the emitter aircraft or the flight, e.g., its ICAO 24-bit address[8] and its aircraft ID[9].

The last twelve fields contain dynamic properties of the emitter aircraft, i.e. properties that evolve over time such as altitude and ground speed. There are eight types of message, depending on the types of property that are communicated. For instance, messages of type 3 contain position information (altitude, latitude, longitude), while messages of type 4 contain velocity information (ground speed, vertical rate and heading). It should be noted that, for the description of the alteration engines, we ignore the fact that there are various types of message, and consider that messages contain all information. This abstraction considerably reduces the complexity of the algorithms without loosing essential information.

IV. Alteration Engines

This section describes all the types of the alteration directives and introduces the related algorithms. Figure 2 depicts, step by step, the workflow shared between all algorithms.

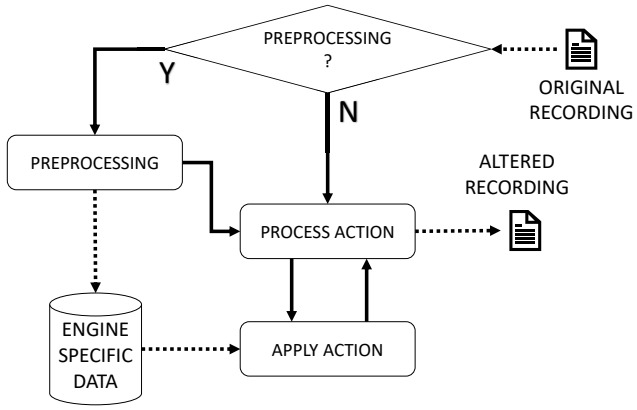

Fig. 2: Process of the alteration engines

The first step of the process consists of determining if a preliminary analysis is required by the current algorithm. Indeed, some algorithms must extract information from the recording to use it later when producing the altered messages. For example, regarding trajectory modification, the dedicated engine needs to generate interpolation functions from the latitude, longitude and altitude of the targeted aircraft. These functions are queried to produce altered values of latitude, longitude and altitude and therefore to alter the initial trajectory of the aircraft.

[7]http://woodair.net/sbs/Article/Barebones42_Socket_Data.htm
[8]https://www.iomaircraftregistry.com/flight-operations/flight-operations/icao-24-bit-aircraft-address-mode-s-coding/
[9]https://www.faa.gov/licenses_certificates/aircraft_certification/aircraft_registry/releasable_aircraft_download/

The workflow of the process is mainly based on *process action* and *apply action* steps. The *process action* method iterates over the recording, and checks if specific conditions are verified (e.g., the message is in the scope of the alteration and the emitter aircraft is targeted by the alteration). Each time it occurs, the message is sent to the *apply action* method that applies the supplied alteration directive to the message. Then the resulting message is returned to the *process action* step and written in the altered recording.

Each alteration engine implements a specific version of the pre-analysis, *process action* and *apply action* methods. We detail each of them below. Interested readers may access the source code on GitHub [10].

A. Property Modification

Most alterations consist of changing the properties of received surveillance messages. It allows users to be quite precise regarding the message properties they want to modify. It also makes it possible to perform *False Alarm* attacks and *Aircraft spoofing* since both consists of changing the value of a single property: squawk code and ICAO address, respectively.

Property modification is represented as an alteration directive $dir_{prop} = (s, t, P)$ where s is the duration of the attack defined as a time interval relative to the recording, t is a list of targeted aircraft, and P is a non-empty set of property value changes. A property value change is represented as a triplet $p = (p.i, p.v, p.o)$ where $p.i$ is the property identifier (e.g., altitude or ground speed), $p.v$ is a value, and o specifies how $p.v$ shall be used to modify the initial value of the propoerty. $p.o$ can be of four types: REPLACE, OFFSET, NOISE, and DRIFT. If $p.o = REPLACE$ then $p.v$ replaces the initial value. If $p.o = OFFSET$ then $p.v$ is added to the initial value. If $p.o = NOISE$ then a random value ranging 0 to $p.v$ is added to the initial value. Finally, if $p.o = DRIFT$ then $p.v$ plus the sum of the previous drift is added to $p.v$.

The *process action* method for the Property modification alteration is described in Algorithm 1. It iterates the messages of the initial recording and, for each message, if it was sent by a targeted aircraft within the alteration time frame (line 3) then the message as well as the alteration directive are sent over to the *apply action* method, which returns the altered version of the message (line 4). Otherwise, the initial message is preserved as it is (line 7). Finally, the obtained message (altered or preserved) is added to the resulting recording al_rec (line 9) and, once all messages are processed, the resulting recording is outputted (line 11).

The algorithm that implements the *apply action* method for the Property Modification algorithm is presented in Algorithm 2. It iterates over the list of property value changes (line 2) and, for each property, it checks in line 3 if the property value change is an offset. If it is an offset then the resulting message is the addition of the initial value of the property and the property value contained in the directive (line 4). Otherwise, the message initial value is replaced by the directive property value (line 6). Note that the dp variable represents the drift progress and, as such, it is only used if $dir.p.o = DRIFT$.

Algorithm 1: *process action* Method for Property Modification

Input: *rec* \ list of genuine surveillance messages
dir \ alteration directive
Result: al_rec \ list of genuine and altered messages

1 $dp \leftarrow 1$
2 **foreach** $old\_msg \in rec$ **do**
3 **if** $old\_msg.ts \in dir.s \wedge old\_msg.icao \in dir.t$ **then**
4 $new\_msg \leftarrow applyAction(old\_msg, dir, dp)$
5 $dp \leftarrow dp + 1$
6 **else**
7 $new\_msg \leftarrow old\_msg$
8 **end**
9 $al\_rec \leftarrow al\_rec \cup \{new\_msg\}$
10 **end**
11 **return** al_rec

Algorithm 2: *apply action* Method for Property Modification

Input: *msg* \ genuine surveillance message
dir \ alteration directive
dp \ drift progress
Result: al_msg \ altered message

1 $al\_msg \leftarrow copy(msg)$
2 **foreach** $p \in dir.P$ **do**
3 **if** $p.o = OFFSET$ **then**
4 $al\_msg.getParam(p.i).value \leftarrow al\_msg.getParam(p.i).value + p.v$
5 **else if** $p.o = REPLACE$ **then**
6 $al\_msg.getParam(p.i).value \leftarrow p.v$
7 **else if** $p.o = NOISE$ **then**
8 $al\_msg.getParam(p.i).value \leftarrow al\_msg.getParam(p.i).value + rand(0, p.v)$
9 **else if** $p.o = DRIFT$ **then**
10 $al\_msg.getParam(p.i).value \leftarrow al\_msg.getParam(p.i).value + p.v \times dp$
11 **end**
12 **end**
13 **return** al_msg

B. Aircraft Disappearance

The goal of this type of alteration is to hide aircraft, i.e. delete messages of certain aircraft at a certain time. It is represented as an alteration directive $dir_{del} = (dir.s, dir.t, dir.n)$ where $dir.s$ is the duration of the attack defined as a time interval relative to the recording, $dir.t$ is the targeted aircraft and $dir.n$ is the number of deleted consecutive messages, e.g., if $dir.n = 0$ then all messages are deleted, while if $dir.n = 3$, for instance then three messages are deleted.

This is certainly the simplest form of alteration because it solely dismisses messages originating from certain aircraft during a certain time frame. The algorithm for this alteration lies in the *process action* method, where messages are written in the resulting altered recording only if the sending aircraft is

[10]https://github.com/aymeric-cr/sbs-generation

not targeted and the message was not sent within the specified time frame, i.e. $\neg(msg.icao \in dir.t \wedge msg.ts \in dir.s)$.

C. Virtual Trajectory Modification

Altering the trajectory of aircraft is a much more complex problem than simply modifying property values at a certain time. This should be done realistically, i.e. with regards to aircraft physical characteristics, because it would be easily detected otherwise. Dynamic properties of aircraft can be formalized as continuous functions of property values related to time, which mimic real-life aircraft physical behaviour as closely as possible. A good candidate to address this issue is interpolation function since it "fills the gap" between each pair of consecutive values.

We opted for the Akima interpolation [2] since local interpolation technique is not subjected to Runge's phenomenon, i.e. a problem of oscillation at the edges of an interval that occurs over a set of equally spaced interpolation points [7] (as opposed to global interpolation). Another benefit of this technique is a faster computation of approximation functions because it only uses the neighbouring points for calculation. Moreover, because the volume of recordings can be substantial (a 30min recording from one sensor may contain around 150000 squitters), a fast interpolation method directly contributes to improve the scalability of the appoach. It should be noted that interpolation is a form of approximation and, as such, there is a certain share of uncertainty in the calculated property values, i.e. interpolation divergence. Since this level of uncertainty would not be acceptable to gain sufficient confidence regarding the efficiency of FDIA detection of critical systems, it appears to be a good fit to demonstrate the capabilities of our approach to rely on a third party "aircraft simulation module" to generate realistic aircraft trajectories.

Algorithm 3: Pre-analysis Method for Trajectory Creation

Input: *recording* \ list of genuine surveillance messages
dir \ alteration directive
Result: trajs \ list of aircraft trajectories

1 $updated \leftarrow zeros(dir.t.size)$
2 **foreach** $msg \in recording$ **do**
3 **if** $msg.icao \in dir.t$ **then**
4 **if** $\neg(msg.ts \in dir.s)$ **then**
5 $trajs(msg.icao)$
6 $.addPos(msg.lat, msg.lon, msg.alt, msg.ts)$
7 **else if** $updated(msg.icao) = 0$ **then**
8 **foreach** $\omega \in dir.\Omega$ **do**
9 $trajs(msg.icao)$
10 $.addPos(\omega.lat, \omega.lon, \omega.alt, \omega.ts)$
11 **end**
12 $updated(msg.icao) \leftarrow 1$
13 **end**
14
15 **end**
16 **end**
17 **return** $trajs$

Virtual Trajectory Modification is given by an alteration directive $dir_{vtm} = (dir.s, dir.t, dir.\Omega)$ with $dir.\Omega$ a nonempty set of way-points defined by $\omega = (\Omega.lat, \Omega.lon, \Omega.alt, \Omega.ts)$, i.e. 3D coordinates and time passage.

The algorithm for trajectory modification is two-fold. First, as a pre-analysis step, it must populate interpolation functions that account for the whole aircraft trajectory, including the part of the trajectory that should be removed and replaced by the supplied way-points. Second, during the *process action* step, it iterates the recording and, for each targeted message, it replaces the property values with the one obtained by querying the interpolation functions. Note that this implementation does not create additional messages, it only edits existing messages. Therefore, if the altered trajectory is longer that the initial one (i.e. more distance is traveled) then the aircraft ground speed must be augmented accordingly since travel time $dir.t$ is not adjustable (as it is bound to the messages). This can lead to erroneous situations where aircraft fly at an impossible speed (either too high or too low) to travel the altered distance within the fixed time $dir.t$.

Population of the interpolation function, as part of the pre-analysis step, is presented in Algorithm 3. We consider a data structure called *Traj* that contains three interpolation functions (for latitude, longitude and altitude) and an identifier (an ICAO). The algorithm iterates over the recording and, for each message emitted by targeted aircraft, if it was not sent within $dir.s$, the received position is added to a *Traj* instance associated to $dir.t$ (lines 4–6). Internally, each value is added to their corresponding interpolation functions. When the first message (of a certain ICAO) sent within $dir.s$ is iterated (line 7), the algorithm iterates $dir.\Omega$ and adds each way-point ω to the *Traj* instance (lines 9–10). Finally, the trajectory is marked as altered (line 12) and all subsequent messages within $dir.s$ are ignored. In other words, all positions of a given trajectory are added to a Traj instance, except the positions that are within the alteration time window $dir.s$ which are replaced by way-points supplied in $dir.\omega$. The result is a list of *Traj* instances, one for each targeted aircraft.

Once all targeted aircraft are associated with a *Traj* instance, the *process action* method performs the alteration. It iterates the recording and, for each message from targeted aircraft sent within $dir.s$, it calls the *apply action* method. The latter replaces property values of the supplied message according to dir as follows:

- For latitude, longitude and altitude properties, the initial values are replaced with interpolated values using the *Traj* instance.
- For ground speed, track and vertical rate properties, new values are computed based on the interpolated position values. To compute ground speed of a certain message m, for instance, the algorithm first computes the horizontal distance (i.e. ignoring altitude) between two near points of the trajectory, the first one scheduled 5 seconds before m and another one 5 seconds after m. The obtained distance, divided by the time it spent to travel it (i.e. 10 seconds), gives the ground speed.

D. Ghost Aircraft Creation

Within this kind of attack, the attacker creates a fake track from scratch, implying that fake messages is created and inserted into the target recording. Ghost Aircraft Creation is represented as an alteration directive $dir_{gac} = (dir.s, dir.t, dir.P, dir.\Omega)$ where $dir.\Omega$ is a nonempty set of way-points. Its implementation relies on the trajectory modification engine. First, a pre-analysis method is called to create *Traj* instances (the interpolation functions use the way-points of $dir.\Omega$) to be sent to the *process action* method, as shown in Algorithm 4. Since ADS-B messages are sent every 400 to 600ms, the *process action* method generates messages randomly, by increasing their time of sending accordingly (line 6), from $dir.s.start$ (line 1) to the end of the alteration time window $dir.s.end$ (line 2). To populate each message, the method calls the *apply action* method of the Virtual Trajectory modification engine, using the list of *Traj* instances.

Algorithm 4: *apply action* Method for Ghost Aircraft Creation

```
   Input: rec \ list of genuine surveillance messages
          dir \ alteration directive
   Result: al_rec \ resulting recording
1  mts ← dir.s.start
2  while mts <= dir.s.end do
3  |  msg ← newMsg(dir.P)
4  |  msg.ts ← mts
5  |  msg ← applyAction(msg)
6  |  mts ← mts + rand(0.4, 0.6)
7  end
8  return al_rec
```

E. Ghost Aircraft Flooding

The initial definition of this attack consists of suddenly creating a lot of ghost aircraft, thus supposedly saturating the Recognized Aircraft Picture (RAP – i.e. what the controller sees). However, it is today quite straightforward for detection systems to recover from this type denial of service. We propose instead to slightly modify the definition of the attack, to be virtual trajectory modification flooding. The goal is to suddenly generate many different trajectories for a targeted aircraft, as if the aircraft was being split in multiple pieces, saturating the detection systems with many conflicting messages.

Aircraft flooding is represented as an alteration directive $dir_{gaf} = (dir.s, dir.t, dir.z, dir.k, dir.\alpha, dir.v, dir.d)$ where:

- $dir.s$ is a time window, where $dir.s.end$ is necessarily the end of the recording;
- $dir.z$ is the number of fake trajectories to create;
- $dir.k$ is the maximum offset value in terms of distance between the fake trajectories and the original trajectory;
- $dir.\alpha$ is a probability distribution ranging 0.1–1 defining the divergence from $dir.k$ of each fake trajectory;
- $dir.v$ is the speed of divergence, i.e. how fast the fake trajectories are offset from the original trajectory; it is expressed as a duration giving the time for the fake trajectory to reach the divergence bound $dir.k \times dir.alpha$;
- $dir.d$ is the direction of divergence of a given fake trajectory; it is a probability distribution that takes its value in a horizontal 90deg cone originating from the aircraft nose and centered on the aircraft track line.

In this way, each fake trajectory is thus created by slowly drifting horizontally from the original trajectory toward direction $dir.d$, up to $dir.k \times dir.\alpha$, at speed $dir.v$. Once the maximum offset has been reached, then the trajectory mimics the original while preserving its horizontal position offset.

Algorithm 5: Trajectory Creation Algorithm for Flooding

```
   Input: initraj \ original Traj instance (up to dir.s.start)
          dir \ alteration directive
   Result: trajs \ set of rogue trajectories
1  for i ← 1 to dir.z do
2  |  α_i ← sample(dir.α) ; ts ← dir.s.start
3  |  d_i ← sample(dir.d) ; traj_i ← initraj
4  |  while ts ≤ dir.s.end do
5  |  |  pCoef ← min((ts − dir.s.start)/dir.v), 1)
6  |  |  step ← pCoef ∗ α_i ∗ dir.k
7  |  |  lat ← fulltraj.interp("lat", ts) + step+
8  |  |       cos(fake_ac.d)
9  |  |  lon ← fulltraj.interp("lon", ts) + step+
10 |  |       sin(fake_ac.d)
11 |  |  traj_i.addPos(lat, lon,
12 |  |       fulltraj.interp("alt", ts), msg.ts)
13 |  |  ts ← ts + dir.
14 |  end
15 |  trajs ← trajs ∪ {traj_i}
16 end
17 return trajs
```

A fake trajectory is represented as a *Traj* instance, built during the pre-analysis phase, and shown in Algorithm 5. We first assume that a *Traj* instance *initraj* has been created, containing all positions of the targeted aircraft (i.e. one position per message) up until $dir.s.start$. Then, for each fake trajectory, the method samples $dir.\alpha$ to determine the divergence magnitude of the trajectory (line 2) and creates the trajectory by duplicating *initraj* (line 3). Next, it creates at regular intervals the way-points that draw the fake trajectory, starting from $dir.s.start$ until the end of the recording. At each interval ts, the algorithm computes the divergence progression percentage $pCoef$ (line 5) by dividing the time distance, from the beginning of the alteration to the current interval, by the specified duration of $dir.v$. Then the euclidean distance from the initial position $step$ is computed by multiplying $pCoef$ to the sampled value of α and the maximum offset $dir.k$ (line 6). Finally, the current latitude and longitude values are obtained using $step$, trigonometric functions *sin* and *cos*, and the sampled direction of divergence $dir.k_i$ (lines 7–8). The algorithm returns all fake trajectories $trajs$.

The last step is to convert the fake trajectories into ADS-B messages to be injected to the initial recording. For each message sent by the targeted aircraft within $dir.s$ and for each fake trajectory, the message is duplicated and its dynamic properties are replaced by values obtained from sampling the interpolation functions of the corresponding *Traj* instance.

F. Replay

Although this type of attack is not part of the given taxonomy, we get from recent discussions with experts that Replay attacks represent a very serious threat as it abstracts itself from realism issues. A typical example of such an attack would be terrorists who collected ADS-B messages of a regular flight on a certain day, then, a few days later, hop on a plane, hijack it and physically interact with the ADS-B transponders to send out the messages they previously recorded. This could allow terrorists to change course of flight without being noticed immediately. In this scenario, there is no need to compute realistic altered values to dupe detection mechanisms, since the fake ADS-B messages that are emitted come from a genuine source. Therefore, replay attacks can certainly bypass FDIA detection systems that rely on realism analysis.

Replay is represented as an alteration directive $dir_{replay} = (dir.s, dir.t, dir.r)$ where $dir.r$ is a source recording from which to extract the targeted aircraft in $dir.t$.

The first step is to extract messages from the source recording. This is done as a first pre-analysis phase by iterating the recording messages and checking whether the emitting aircraft is present in $dir.t$ and if the current message was sent within the directive time frame. If it is the case, the message is then marked to be replayed. The list of all the messages that were marked is eventually returned.

The second step is about adjusting the timestamp of the extracted messages so that they can be correctly inserted in the target recording. For each message msg to be replayed, the algorithm assigns a new time stamp that relates to the target recording, as follows:

$msg.ts \leftarrow first\_ts + msg.ts - first\_rep\_ts + offset$

where $first\_ts$ is the time stamp of the first message of the target recording, $first\_reo\_ts$ is the time stamp of the first message extracted for replay, and $offset$ is a value in second expressing where, in the target recording, the previously extracted messages should be inserted.

Algorithm 6: *process action* Method for Replay attack

```
   Input: reps \ list of messages picked up for replay
          rec \ source recording
   Result: al_rec \ resulting recording
1  cur ← rec.first
2  rp ← reps.first
3  while cur ≠ ∅ do
4  |  while reps ≠ ∅ ∧ cur.ts > rp.ts do
5  |  |  al_rec ← al_rec ∪ {rp}
6  |  |  reps ← reps − {rp}
7  |  |  rp ← reps.next
8  |  end
9  |  al_rec ← al_rec ∪ {cur}
10 |  cur ← rec.next
11 end
12 while reps ≠ ∅ do
13 |  al_rec ← al_rec ∪ {rp}
14 |  reps ← reps − {rp}
15 end
16 return al_rec
```

The last step is the merging between the target recording and the list of extracted messages, as perform by the *process action* method detailed in Algorithm 6. The method iterates over the target recording messages (lines 3–11). If the timestamp of the first of the extracted messages cur is smaller than the timestamp of the current message rp, rp is put in the resulting recording al_rec (line 5) while being removed from the list of extracted messages (line 6). Otherwise, cur is added to the resulting recording. If the target recording has been iterated and the list of extracted messages is not empty, the remaining messages are appended to al_rec (lines 12–14).

In conclusion of this part, looking back at the *RQ1* research question defined in Section II-D (**To what extent is it possible to automate the generation of FDIA scenarios that cover the taxonomy of attack?**), we believe this question can be answered positively. We indeed were able to provide a dedicated algorithm for each type of attack from the taxonomy. Moreover, we covered a new type of attack, *Replay*, and we improved the definition of an existing attack, *Ghost Aircraft Flooding* that becomes *Trajectory Modification Flooding*, an FDIA much more complicated to detect.

V. Experimentation

Since ATC constitutes one of the most critical infrastructures on the planet, it is not possible to demonstrate the use of FDI-T in real situation by feeding falsified data to a deployed FDIA detection system and report results in a public manner.

Therefore, we opted instead to confront our tool to a Machine Learning-based detection approach [8] taken from the literature. First, we elaborate on the constituents of the model and their characteristics. Then we present the dataset that were used to train and test the model, and especially how the proposed approach can allow to generate the anomalies presented in [8] as well as a more complex anomaly of our own making. Finally we discuss the experimentation results and propose an answer to the second research question.

A. The model

The authors of [8] propose a Machine Learning method to discriminate malicious messages from regular ones. They use an LSTM-based encoder-decoder, a deep learning architecture. The model is decomposed in two different sub-models:

① **The encoder** learns to create a representation of its input. In this experimentation, it is based on LSTM cells that have been proven relevant in time series analysis. It cannot learn alone as it has no way to check if its output are correct by itself. That is the reason why it is always coupled with a decoder for its training.

② **The decoder** takes the encoder output, namely the vectorized representation of the original data, and decode it to fit the expected output. In the case of Habler et. al's model, the output is actually the input. That is why this kind of encoder-decoder is called an auto-encoder. The decoder will have a similar architecture, if not identical, to the one used in the encoder.

In this experiment, several complete and isolated flights (from take off to landing) are considered and sliced in windows of 15 messages with a stride of 1. Once the model has been trained with normal data, based on the reconstruction score, an anomaly threshold is set. If trained properly, the model should meet trouble to encode malicious data, as they differ from regular ones, so that the given threshold should be exceeded, revealing a detected anomaly.

B. Dataset

To train the model and test it, we used several flights collected from the OpenSky database [19]. Data concern all European flights and include as many flights morphologies as possible, e.g., Madrid-Moscow that use different routes depending on the weather due to its long journey.

After training the model on legitimate data, we created with our approach alteration scenarios based on the anomalies defined in Habler et. al's paper used to validate their approach. For each anomaly, we present below the corresponding alteration scenario and its associated directives using the notation given in Sect. III, completed with the following notation:

- R_{origin}: a recording containing one flight from one European city to another.
- R_{source}: a recording containing one flight from one European city to another, where $R_{source} \neq R_{origin}$
- a_1: the aircraft contained into R_{origin}.
- a_2: the aircraft contained into R_{source}.
- t_n: a duration in seconds that cannot exceed the duration of R_{origin}, i.e. $t_n < t_{n+1}$.

Random noise (RND) – anomalies are generated by adding random noise to messages. To do so, values from original messages are multiplied by a random number between zero and two. RND requires a scenario using a *Property Modification*. We thus formalize the scenario S_{RND} as following:

S_{RND}	$=$	$(R_{origin}, \{dir_1\})$
dir_1	instance of	dir_{prop}
$dir_1.s$	$=$	$[t_0, t_1]$
$dir_1.t$	$=$	$\{a_1\}$
$dir_1.P$	$=$	$\{p_1\}$
p_1	$=$	$(ALTITUDE, 2, NOISE)$

Different Route (Route) – anomalies consists to replace a whole segment of the initial ADS-B messages with a different route taken from OpenSky. Route requires a scenario that combines a *Replay* and an *Aircraft Disappearance* directive. We thus formalize the scenario S_{Route} as following:

S_{Route}	$=$	$(R_{origin}, \{dir_2, dir_3\})$
dir_2	instance of	dir_{replay}
$dir_2.s$	$=$	$[t_0, t_1]$
$dir_2.t$	$=$	$\{a_2\}$
$dir_2.r$	$=$	R_{source}
dir_3	instance of	dir_{del}
$dir_3.s$	$=$	$[t_0, t_1]$
$dir_3.t$	$=$	$\{a_1\}$
$dir_3.n$	$=$	1

Gradual Drift (DRIFT) – anomalies consist to simulate an altitude drift. The altitude messages on the attacked time window are all raised/lower by an increasing/decreasing multiple of n feet. So if the first message is lowered by a hundred feet, the second will be lowered by two hundreds, etc. DRIFT requires a scenario using a *Property Modification*. It is possible to formalize the scenario S_{DRIFT} as following:

S_{DRIFT}	$=$	$(R_{origin}, \{dir_4\})$
dir_4	instance of	dir_{prop}
$dir_4.s$	$=$	$[t_0, t_1]$
$dir_4.t$	$=$	$\{a_1\}$
$dir_4.P$	$=$	$\{p_2\}$
p_2	$=$	$(ALTITUDE, 25, DRIFT)$

Velocity Drift (VEL) – anomalies consist of a gradual drift applied to the velocity feature in knot. VEL requires a scenario using a Property Modification. We thus formalize the scenario S_{VEL} as following:

S_{VEL}	$=$	$(R_{origin}, \{dir_5\})$
dir_5	instance of	dir_{prop}
$dir_5.s$	$=$	$[t_0, t_1]$
$dir_5.t$	$=$	$\{a_1\}$
$dir_5.P$	$=$	$\{p_3\}$
p_3	$=$	$(SPEED, 1.0, DRIFT)$

Message blocking (BLK) – only the first message out of every five consecutive messages is preserved to simulate a denial-of-service attack or a network delay mode. BLK requires a scenario using an *Aircraft Disappearance* directive. We thus formalize the scenario S_{BLK} as following:

S_{BLK}	$=$	$(R_{origin}, \{dir_6\})$
dir_6	instance of	dir_{del}
$dir_6.s$	$=$	$[t_0, t1]$
$dir_6.t$	$=$	$\{a_1\}$
$dir_6.n$	$=$	4

It must be noted that it was possible to generate all the anomalies presented in the publication associated to the detection model. In addition, we also experimented an anomaly that was not part of the model's validation data set. It consists of a trajectory modification **(TRJ)** of a hundred kilometers throughout a long period of flight time (about 60%).

C. Results

Out of the five addressed anomalies, we obtained similar results to Habler et al's on four of them. Figure 3 depicts the drift down anomaly on a flight from Moscow to Madrid. We can clearly see the pike between the message 180 and 250 going above the anomaly threshold, defined in a similar fashion as the reference experimentation. Similar results can be observed on the 3 other anomalies. However, we do not obtain a comparable pike on the **BLK** anomaly, thus not triggering any anomaly flag. One assumption is about the difference between our data and the ones used in the original experiment.

For instance, results can differ depending on the time gap between messages. If this gap is bigger, deleting 4 messages out of 5 means that the aircraft goes dark for a longer period of time, easing the detection. Overall, the false data injections recreated by FDI-T is also detected as handcrafted ones.

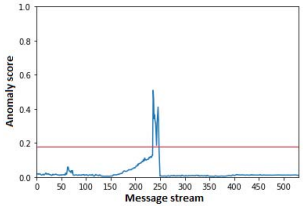

Fig. 3: Drift Down Anomaly Score of FDIA.

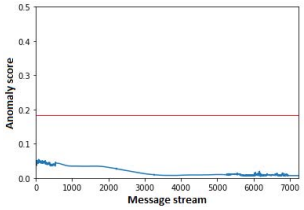

Fig. 4: Evolution of anomaly score for **TRJ**.

As for the results presented in Figure 4 concerning the anomaly **TRJ** generated on a flight between Oslo and Paris, we find that the anomaly score goes way below the detection threshold set for the other five anomalies. It results to an unnoticed attack. However, we can also observe that the use of FDI-T to modify trajectories has the side effect of smoothing the curve, making it detectable for a human eye. A workaround is to add some noise to generate more realistic data.

We show with **TRJ** that our tool is capable to create alterations that would be too cumbersome to create by hand as we are close to 5000 messages modified from the original flight. In addition, these crafted attacks are smart enough to be not detected by a model taken from the litterature, which exposes its limits on real contexts. So, we can now provide an answer to the research question *RQ2* (**To what extent is the approach relevant to generate synthetic data for AI Testing?**). Not only it was possible to generate the anomalies that were used to evaluate Habler et. al.'s model, but the presented approach made it possible to generate a more complex anomaly that was not picked up by the model. Moreover, it is suitable for providing synthetic test data for ML based detection systems.

VI. Conclusion

This paper describes a novel test data generation approach to be part of an existing FDIA testing framework dedicated to ATC systems, called FDI-T. More precisely, the contribution concerns the various generation algorithms, referred to as alteration engines. The paper shows the relevance of these algorithms, making it possible to successfully design and experiment all classes of scenarios of the ADS-B attack taxonomy. Moreover, it was possible to reproduce and extend the FDIA test data of a detection model from the literature. Among future work, we have started to experiment our approach in the context of maritime surveillance, which is based on a communication protocol (AIS), similar to the ADS-B protocol.

VII. Acknowledgement

This work is part of an ongoing research initiative toward the generation of FDIA test scenarios partially carried out in partnership with the french company Kereval. This work is partially supported by a UBFC-ISITE-BFC Grant.

References

[1] EUROCAE Working Group 51. Safety, performance and interoperability requirements document for ads-b/nra application. Technical report, The European Organisation for Civil Aviation Equipment, 2005.

[2] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. 17:589–602, 1970.

[3] A. Barreto, M. Hieb, and E. Yano. Developing a complex simulation environment for evaluating cyber attacks. In *Interservice/Industry Training, Simulation, and Education Conf. (I/ITSEC)*, volume 12248, pages 1–9, 2012.

[4] P. Brooker. Sesar and nextgen: investing in new paradigms. *The Journal of Navigation*, 61(2):195–208, 2008.

[5] Nazly Rocio Santos Buitrago, Loek Tonnaer, Vlado Menkovski, and Dimitrios Mavroeidis. Anomaly detection for imbalanced datasets with deep generative models.

[6] A. Cretin, B. Legeard, F. Peureux, and A. Vernotte. Increasing the resilience of ATC systems against false data injection attacks using DSL-based testing. In *Proc. of the 8^{th} Int. Conf. on Research in Air Transportation (ICRAT'18), Doctoral Symp.*, pages 1–4, Spain, 2018.

[7] J F Epperson. On the runge example. *The American Mathematical Monthly*, 94(4):329–341, 1987.

[8] E Habler and A Shabtai. Using lstm encoder-decoder algorithm for detecting anomalous ads-b messages. *Computers & Security*, 78:155–173, 2018.

[9] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, and A. Tolba. Lotad: long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web*, 21(3):825–847, May 2018.

[10] Yao Liu, Peng Ning, and Michael K Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[11] M. Ma. Resilience against false data injection attack in wireless sensor networks. In *Handbook of Research on Wireless Security*, pages 628–635. IGI Global, 2008.

[12] P. Malhotra, A. Ramakrishnan, G? Anand, L. Vig, P. Agarwal, and G. Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[13] M. Riahi Manesh and N. Kaabouch. Analysis of vulnerabilities, attacks, countermeasures and overall risk of the automatic dependent surveillance-broadcast (ADS-B) system. *Int. Journal of Critical Infrastructure Protection*, 19:16 – 31, 2017.

[14] I. Martinovic and M. Strohmeier. Security of ads- b: State of the art and beyond. *DCS*, 2013.

[15] T. Maruthi Padmaja, Narendra Dhulipalla, Raju S. Bapi, and P. Radha Krishna. Unbalanced data classification using extreme outlier elimination and sampling techniques for fraud detection. pages 511–516. IEEE.

[16] H. R. Roth, L. Lu, J. Liu, J. Yao, A. Seff, K. Cherry, L. Kim, and R. M. Summers. Improving computer-aided detection using convolutional neural networks and random view aggregation. *IEEE Transactions on Medical Imaging*, 35(5):1170–1181, May 2016.

[17] M. Schäfer, V. Lenders, and I. Martinovic. Experimental analysis of attacks on next generation air traffic communication. In *Int. Conf. on Applied Cryptography and Network Security*, pages 253–271. Springer, 2013.

[18] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, 2017.

[19] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm. Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research. In *Proc. of the 13th Int. Sym. on Information Processing in Sensor Networks*, IPSN '14, pages 83–94, Germany, 2014.

[20] Z. Shi, M. Xu, Q. Pan, B. Yan, and H. Zhang. Lstm-based flight trajectory prediction. In *2018 Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1–8, July 2018.

[21] A Smith, R Cassell, T Breen, R Hulstrom, and C Evers. Methods to provide system-wide ADS-B back-up, validation and security. In *25th Digital Avionics Systems Conf.*, pages 1–7. IEEE, 2006.

[22] M. Strohmeier, M. Schäfer, R. Pinheiro, V. Lenders, and I. Martinovic. On perception and reality in wireless air traffic communications security. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1338–1357, 2017.

[23] K. D Wesson, T. E Humphreys, and B. L Evans. Can cryptography secure next generation air traffic surveillance? *IEEE Security and Privacy Magazine*, 2014.

[24] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5:21954–21961, 2017.

[25] R. Zhang, G. Liu, J. Liu, and J P Nees. Analysis of message attacks in aviation datalink communication. *IEEE Access*, 2017.