OpenSky Report 2018: Assessing the Integrity of Crowdsourced Mode S and ADS-B Data

Matthias Schäfer[¶‡*], Martin Strohmeier[¶†], Matthew Smith[¶†], Markus Fuchs[¶‡], Vincent Lenders[¶§], Ivan Martinovic[¶†]

[¶]OpenSky Network, Switzerland
lastname@opensky-network.org

[*]TU Kaiserslautern, Germany
schaefer@cs.uni-kl.de

[†]University of Oxford, UK
firstname.lastname@cs.ox.ac.uk

[‡]SeRo Systems, Germany
fuchs@sero-systems.de

[§]armasuisse, Switzerland
vincent.lenders@armasuisse.ch

***Abstract*—Air traffic tracking platforms such as the OpenSky Network use crowdsourcing to track air traffic world wide. While crowdsourcing allows for unprecedented coverage, it comes with challenges concerning the integrity of the data. More specifically, by delegating the data collection task to a mostly unknown group of individuals, the network becomes vulnerable to data integrity breaches by inexperienced or malicious actors. Users might – intentionally or not – send faulty or fake data to the tracker and in this way threaten the integrity of the information provided by the network. In this paper, we provide unique insights into the data integrity challenges we faced during 6 years of operating the OpenSky Network. We analyze the different types of integrity breaches and discuss methods to detect and filter faulty data.**

I. Introduction

Crowdsourcing is a well-established paradigm in the commercial air traffic tracking domain. Volunteers around the world (the "crowd") set up and operate large numbers of receivers for transponder signals and send the live tracking data to a central server via the Internet. This approach has been used by several flight trackers such as Flightradar24, FlightAware, and the OpenSky Network to achieve worldwide coverage within just about a decade. This scale is considerable given that not even aviation authorities have managed to build a globally connected surveillance network. Backed by a large community of aviation enthusiasts from all parts of the world, crowdsourcing has helped to overcome financial, organizational, and language barriers within a relatively short amount of time.

The crowdsourced air traffic control (ATC) data is used for a plethora of applications. Most (if not all) of these applications have in common that they implicitly rely on the integrity of the data. For example, commercial applications such as automated delay compensation claims or media reports about incidents need reliable information as they have financial and legal implications. Also non-commercial use cases such as incident investigation by authorities or scientific studies like our previous reports [1], [2] [1] draw conclusions from the data with real-world implications. By nature, each of these applications is practically rendered unusable if the underlying data fails to provide a sufficient level of information integrity.

[1]See https://opensky-network.org/community/publications for a more exhaustive list of studies

Data integrity poses a major challenge in the crowdsourcing context. The main reason for this is that the control over the on-site infrastructure rests not with the network operator but with the crowd. It is often impossible for the central network operator to access or control the software and hardware which produces the data sent to the network. Hence, the maintenance of the receiving infrastructure (e.g., installing updates to fix software bugs) depends on many independent individuals and is often neglected. In addition, crowdsourced networks have to be as open as possible to foster growth. This means that they usually allow anybody to feed any data to the network, including faulty, fake, or intentionally misleading data.

To make the benefits of crowdsourcing available to integrity-sensitive applications nevertheless, adequate server-side methods to detect and filter the different kinds of integrity breaches are needed. In this paper, we provide unique insights into the data integrity challenges we encountered during the 6 years of operation of the OpenSky Network. We use a large set of surveillance data gathered by the network to analyze and quantify integrity aspects of the crowdsourced data. We furthermore provide an overview of the methods used by OpenSky to clear its data of the different kinds of noise.

This paper makes the following contributions:

- We provide a novel taxonomy of integrity breaches occurring in crowdsourced air traffic control networks. In this, we cover both intentional and unintentional breaches that affect the quality of the crowdsourced data.
- We analyze several case studies of such breaches occurring in the real world. The provided examples stem from the OpenSky Network, which has been continuously operated for six years.

The remainder of this paper is organized as follows. Section II describes the current state of the OpenSky Network. Section III outlines our definition of integrity, while Section IV provides a taxonomy of breaches occurring in crowdsourced ATC networks. Section V presents some real-world cases of integrity breaches while Section VI suggests possible avenues how to detect and handle them. Section VII discusses our experiences and finally Section VIII concludes this work.

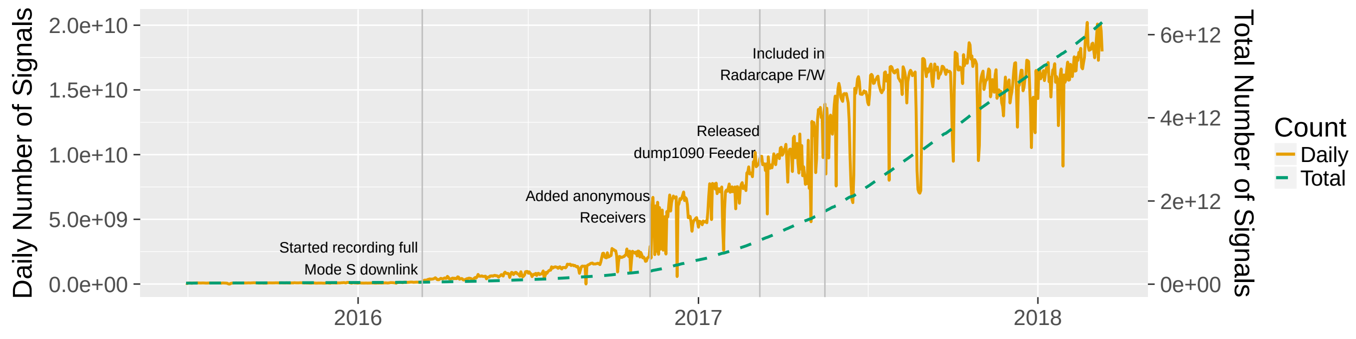

Figure 1: The growth of OpenSky's dataset over time from June 2013 to March 2018

II. The OpenSky Network

The OpenSky Network is a crowdsourced sensor network collecting air traffic control (ATC) data. Its objective is to make real-world ATC data accessible to the public and to support the development and improvement of ATC technologies and processes. Since 2012, it continuously collects air traffic surveillance data. Unlike commercial flight tracking networks (e.g., Flightradar24 or FlightAware), the OpenSky Network keeps the raw Mode S replies as they are received by the sensors in a large historical database which can be accessed by researchers and analysts from different areas.

The network started with eight sensors in Switzerland and Germany and has grown to more than 1000 active receivers in over 70 countries. As of this writing, OpenSky's dataset contains five years of ATC communication data. While the network initially focused on ADS-B only, it extended its data range to the full Mode S downlink channel in March 2017. The dataset currently contains more than 8 trillion Mode S replies. The growth of the dataset is depicted in Figure 1. Besides the payload of each Mode S downlink transmission, OpenSky stores additional metadata. Depending on the receiver hardware, this metadata includes precise timestamps (suitable for multilateration), receiver location, and signal strength. For more information on OpenSky's history, architecture and use cases refer to [3], [4] or visit http://opensky-network.org.

III. Data Integrity

At first, it is imperative to provide our definition of data integrity in the context of crowdsourced ATC receiver networks. The definition given by the Oxford English Dictionary specifies the 'internal consistency or lack of corruption in electronic data'. In information security, the focus is on 'maintaining and assuring the accuracy and completeness of data over its entire lifecycle' [5], where we consider the lifecycle to comprise everything from the creation of a message in an aircraft transponder over the reception by a crowdsourced receiver until its storage in OpenSky's systems.

There are two distinct types of data that we collect at OpenSky: the content of a message and its metadata.

- With respect to the content of the SSR Mode S downlink communication (including 1090ES ADS-B), we consider a datum being corrupted, if the bit sequence making up the message arriving at OpenSky was never actually transmitted by a legitimate transponder.

- With respect to the metadata (e.g., timestamps, signal strength, and other physical layer information), a datum is corrupted if it is inaccurate to a degree unusual for the system's normal operation. Note that the bounds of normal operation can be very broad for some metadata, for example the received signal strength of a message.

Furthermore, we explicitly consider the absence of data also as a potential integrity problem, referenced in the definition's allusion to completeness. In the literature, this problem is often covered separately as 'availability' [6]. However, in the application domain of aircraft tracking, the selective absence of low-level message signals can have an impact on higher-level abstractions, for example the interpolation of aircraft tracks by the system.

Considering our definition, we have to deal with the fundamental problem that it is impossible to obtain absolute ground truth on the number, types and metadata of messages that should be received by a sensor or the system as a whole. This means it is not possible to exactly quantify the extent of the problem, instead, we rely only on the indicators available to us to detect and assess the extent of the problem in our work.

IV. Taxonomy of Integrity Breaches

Fundamentally, we differentiate between two different types of integrity breaches, unintentional and intentional. Whereas the former happens without malicious purpose, the latter is conducted by actors who want to attack and deteriorate the data quality of the system. Fig. 2 illustrates our taxonomy.

A. Unintentional Breaches

We classify seven different types of breaches that are unintentional but negatively impact the integrity of the data processed by the crowdsourced system. These range from various soft- and hardware problems on both the sender and receiver side to environmental reasons.

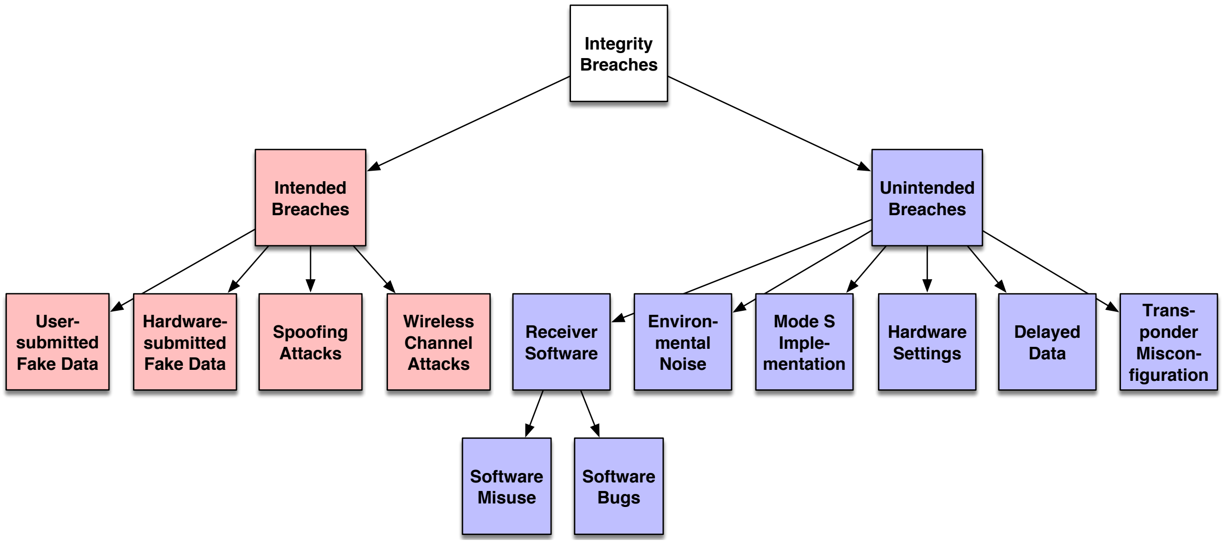

Figure 2: Taxonomy of integrity breaches in crowdsourced ATC sensor networks.

In a similar vein, some ADS-B enthusiasts operate their own multilateration (MLAT) network, effectively using the data of several distributed receivers to localize aircraft independently from their position broadcasts by exploiting the time distances of arrival of the message signals. The intentions behind this approach are laudable, as it can increase for example the knowledge about aircraft without ADS-B transponders. However, feeding such data to OpenSky is not desired by the network as it is not possible with the current feeding protocols to distinguish between real message transmissions and data artificially produced from MLAT results. In addition, if the network wants to operate its own MLAT implementation, it is not possible with this combined data stream as it cannot separate the data from the different receivers. Hence, the optimal solution would be to send the data from the receivers via separate feeds rather than through a combined feed.

1) Receiver software bugs: In the large and varied ecosystem of software available for the decoding of Mode S messages and feeding to crowdsourced networks, there are invariably bugs that affect the data integrity. The previously mentioned re-encoding of received data is error prone and introduces new potential for problematic bugs. Thus, we have seen many receivers sending us scrambled data which is likely caused by bugs in the encoding software component.

2) Environmental noise: As every wireless technology, Mode S and ADS-B suffer from typical artefacts caused by the propagation over the wireless channel. This includes for example multipath propagation, an effect familiar for all radar technologies, which has the same signal arriving at the receiver through two (or more) different paths, leading to the duplicate reception and processing of the message. Other physical effects, which lead to interferences with the message signal, can lead to similar outcomes.

3) Mode S implementation: Physical layer interference may also result in bit errors which cannot be detected. Although Mode S employs a cyclic redundancy check (CRC) to be able to detect bit errors, the actual implementation of this CRC limits the protection, especially as a passive bystander. The reason for that is that in many reply types, the CRC is XORed with the transmitter or the receiver ID. If both are not known a priori, the CRC is basically unusable (see [7] for a more detailed explanation). For example, in all-call replies, the CRC at the end of the reply is XORed with the original interrogator ID [1]). There are 80 valid interrogator IDs and if the uplink data stream is unknown, there is no way to detect bit errors in the last few bits. Many receivers (e.g., Radarcape) have problems with this artefact, causing them to produce a non-negligible amount of duplicates and noisy data.

4) Hardware settings: Another effect related to the detection of seemingly valid replies from environmental noise is due to the settings used by the widely heterogeneous receiver hardware. Many receivers aim to provide maximum sensitivity to produce the highest possible reception rate rather than the highest integrity level, which is typically a direct trade-off. High message reception rates are usually achieved by using extremely low transmission detection thresholds. If the threshold is sufficiently low, a receiver can pick up a high number of random (but superficially valid) Mode S replies from environmental noise.

5) Delayed data: Some receiver setups experience an unusually high delay when forwarding received data to the OpenSky Network for reasons unknown to us. Besides the usual Internet propagation delay, we have observed extreme cases where data is seemingly buffered for 30-60 minutes and then sent to our servers with a large delay. This can negatively impact the performance and the tracking of the system as such data needs to be properly sorted or filtered out.

6) Transponder misconfiguration: Finally, not only receivers can produce integrity breaches, we have also observed many aircraft, which transmit erroneous data. Examples of misbehaving transponders deployed on aircraft range from misconfigured ICAO 24-bit identification to the broadcast of wrong position advertisements (potentially based on false navigational data, e.g. when based on dead reckoning, see [8] for more information).

B. Intentional Breaches

In contrast to the previously listed integrity breaches, the following are based on actively malicious behaviour, either by a feeder or by an outsider who manipulates the data previous to its reception by a feeding user (i.e., on the sending side or even while on the wireless channel).

1) User-submitted fake data: Considering the fact that there are a) no cryptographic integrity checks in Mode S, and b) no authentication options in the current set of feeding protocols used by any of OpenSky's supported sensors, anyone can sign up and send incorrect or outright fabricated data to a crowdsourced ATC network. Motivations for such actions could range from the personal, e.g., to obtain free access to services offered by the network (some of which require to become a feeder) to ulterior motives held by competing networks, which seek harm the reputation and operations of another network by compromising the integrity of their data. Where crowdsourced ATC data is used for other purposes such as a backup to ATC surveillance or to provide weather data (as in [7]), the aim of an attacker could also be to subvert the processing and results of these services.

2) Hardware-submitted fake data: It is further conceivable that hardware sets provided to users by other commercial tracking networks might produce data that is watermarked or otherwise modified in a particular way only known to the original manufacturer. This could be done to compromise the feed integrity and prevent the hardware from being used by competitors. Similar data poisoning incidents have happened before in other domains, e.g., in peer to peer file distribution [9] and streaming networks [10].

3) Spoofing attacks: Finally, it has been well known for several years that it is trivial to spoof Mode S and ADS-B messages due to the lack of cryptographic security in these technologies [11], [12]. An attacker can manipulate every part of a message at will using software-defined radio transceivers and send it over the wireless channel (using the 1090 MHz frequency or 978 MHz in case of the UAT datalink), where it can be picked up by receivers, which feed to crowdsourced networks. In OpenSky we have seen users experimenting with such attacks using their own sensors, which could easily be filtered as the spoofed aircraft used appropriate callsigns (e.g., 'TEST1234'). However, filtering a more sophisticated attack would require equally advanced defenses.

4) Wireless channel attacks: Besides spoofing, i.e., the creation of messages from scratch, it is also possible to conduct several attacks, which affect existing legitimate signals on the 978 or 1090 MHz frequencies. It has been shown that it is possible to modify messages sent by aircraft, if an attacker is in the right position compared to a receiver and the target airplane [12]. Finally, it is also possible to outright jam the signals at a receiver, causing a denial of service. This can be done either through broadly jamming the frequency, affecting all messages or by selectively attacking only specific messages, for example of a particular aircraft [13]. The absence of messages can also affect data integrity in line with our definition.

V. Case Studies

We look at several case studies of unintentional interference with the data of OpenSky and discuss their possible reasons.

A. Data Set

The data set considered in this work is a snapshot of the unmodified data ("raw data") that came into OpenSky between 6am and 7am UTC on the 27th of June 2018. During this period, 922 sensors from 70 countries reported 1.1 billion Mode S signal receptions to the network. We decoded the Mode S replies using the latest version of our open-source decoding framework libadsb[2] and applied several preparation algorithms to it. Most notably, we applied spacial binning to the data, that is, we assigned a bin ID to each ADS-B position report. Each of these bins has a horizontal size of 10x10 km and a height of 3000 ft. The horizontal bin was determined using Albers equal-area conic projection. This binning allows us, for instance, to calculate the horizontal coverage in the enroute airspace by multilying the number of distinct horizontal bins by 100 square kilometers.

In the end, we generated two different views on the raw data for our analyses:

1) Statistics: we generated a view containing different statistics for each sensor and aircraft pair. These statistics include the number of Mode S replies, the number of ADS-B messages separated by format type code, and the number of messages which matched the tracking algorithm described in [7]. It furthermore includes the type of the receiver as well as the country associated with the ID of the transponder (according to [14]). In total, 222,755 of these statistics were generated from the raw data.
2) Coverage: the coverage data set contains the results of the binning algorithm mentioned above. Each data point in this view represents a sensor-transponder-bin relationship. In other words, the view contains the information at which point in time a certain receiver saw a certain aircraft in a certain bin. This allows for cross-correlating spaciotemporal information over sensors and aircraft. In total, 3,393,258 of these data points were generated from the raw data.

[2]https://github.com/openskynetwork/java-adsb

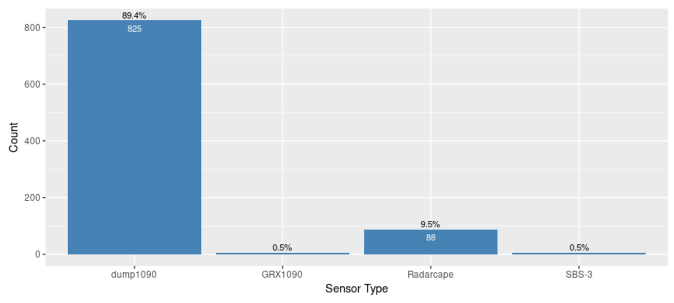

Figure 3: Distribution of the different sensor types in the examined data set.

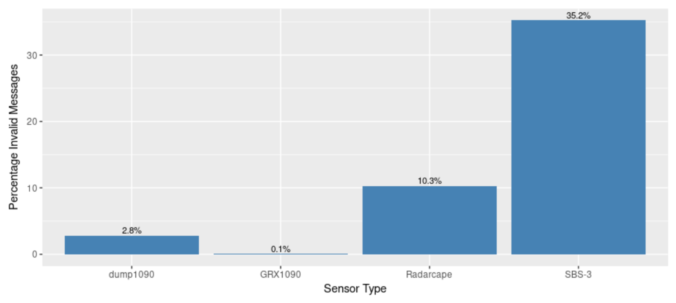

Figure 5: Average percentage of invalid messages in the data reported by each receiver type.

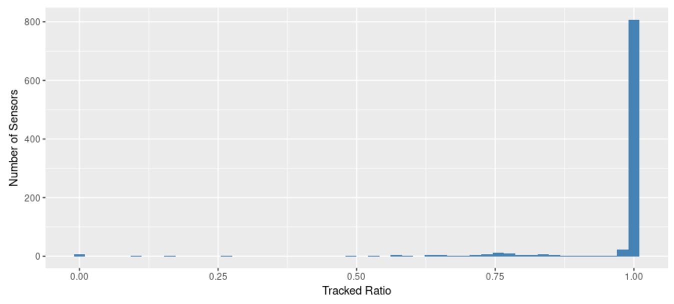

Figure 4: Distribution of $\mathrm{R}_{tracked}$ over all sensors.

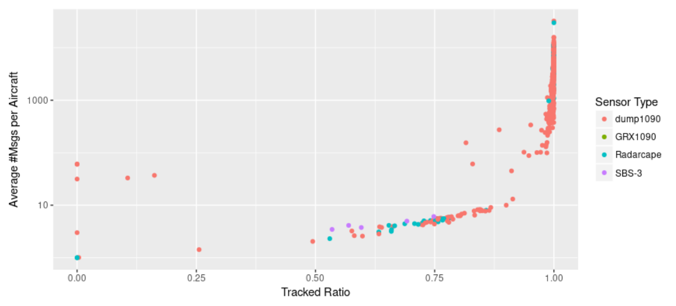

Figure 6: Distribution of tracked ratio against the average number of messages per aircraft for different sensors in the examined data set (logarithmic y-axis scale).

OpenSky supports four main sensor types: dump1090, Radarcape, GRX1090 and SBS-3. Figure 3 shows the distribution of the sensors in our examined dataset. We can see that a large majority of the 923 sensors are using a version of the dump1090 receiver software (825 or 89.4%). Radarcapes make up 88 devices or 9.5% of the share, while legacy SBS-3 receivers, which were the original basis of OpenSky, now only make up 5 devices or 0.5% of all sensors, just as the newly supported GRX1090.

B. *Environmental Noise*

Depending on the receiver setup, there are some receivers with a very low detection threshold for signals. As a result, they produce quite a high level of environmental noise in terms of random (but seemingly valid) frames picked up from the noise. Picking the right threshold in a conservative receiver is a tradeoff between sensitivity and false positive detection rate. Although this is certainly not a malicious integrity breach, it poses many problems and produces ambiguities when processing data from a large network such as OpenSky.

As an indicator for noise, we calculated the ratio $\mathrm{R}_{tracked}$ of messages which passed the tracking algorithm described in [7] and the total number of messages that were detected for a certain aircraft. This ratio is a good indicator for noise since the tracking algorithms are specifically made for eliminating messages with erroneous transponder IDs created from the environmental noise. The distribution of every sensor's $\mathrm{R}_{tracked}$ is shown in Fig. 4.

The vast majority of sensors work as intended, that is, they have a $\mathrm{R}_{tracked}$ close to 1. Although we are able to detect and filter this kind of noise, there are still reasons to measure this indicator and exclude sensors with a low tracked ratio from the data collection. For example, if the fraction of useful data coming from a sensor is extremely small, the network operator might not want to waste resources (storage, CPU time, network bandwidth) on these sensors' data. In addition, a low tracked ratio might indicate software bugs or setup problems which could be solved by the users.

Thus, it may be sensible to chose a threshold $\hat{\mathrm{R}}_{tracked}$ to minimize such effects. If we want to allow sensors to have at most 10% of 'random' frames in their data, or, having a $\hat{\mathrm{R}}_{tracked} = 0.9$. The number of sensors that do not satisfy this requirement in our data set was 85 or 9.2%.

It is worth noting that there were notable differences between the four sensor types (see Fig. 5). Whereas the dump1090 sensors making up the bulk of OpenSky's feeders had an average noise of 2.8% (calculated as $1 - \mathrm{R}_{tracked}$), and the GRX1090 sensors even the lowest noise at 0.1%, the other types exhibit quite significant differences. Radarcapes came in at 10.3% while for the older SBS-3 boxes more than a third of all messages (35.2%) could only be attributed to noise. This indicates a systematic problem for this sensor type. As for the other types, the $\mathrm{R}_{tracked}$ variance seems to depend on several possible factors such as antenna, SDR version, or software version.

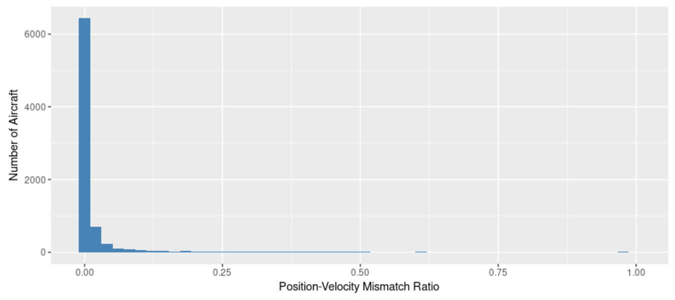

Figure 7: Distribution of position-velocity mismatch ratios over all aircraft.

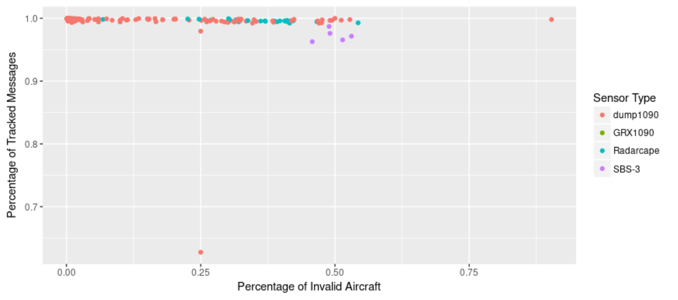

Figure 8: Percentage of invalid ICAO 24-bit transponder IDs reported by each of the receivers.

Finally, we look at how the $\mathrm{R}_{tracked}$ correlates with the average number of messages per aircraft transponder ID seen by a sensor. Fig. 6 shows that the sensors which have seen a particularly low number of messages per aircraft are those with the highest noise level. On average, they have only received very few messages per transponder ID which indicates that the different observed transponder IDs were merely a result of bit errors and invalid replies.

If we turn our focus on aircraft rather than sensors, there are many random aircraft in the data set that are merely an artefact of the random data picked up from the RF channel's noise. Indeed, the large majority of observed transponder IDs received from decoded messages have no (or almost no) valid data. This is explained by the fact that the vast majority of transponder IDs were seen in Mode S replies that did not pass the tracking filter. If we remove the IDs with $\mathrm{R}_{tracked} = 0$, the number of remaining IDs in the dataset is 34782, or 0.23% of the originally 15 million observed IDs. Thus, a huge portion (more than 99%) were products of decoding noise. If we filter these data out, 6 of the sensors even produce no valid data at all.

C. *Velocity vs. Position Change*

While the previous case study targeted decoding noise, we now analyze implausible data that was either really sent by aircraft or artificially generated by a receiver. As long as ADS-B transponders and airborne equipment work as expected, the reported positions should change according to the reported velocity. Based on our experience with ADS-B data, some transponders do not update their positions at the same rate as they broadcast it. As a result, we need to apply some averaging here since, for example, the same position might be reported several times although standing still is literally impossible for airborne airliners.

Fig. 7 shows the distribution per aircraft regarding the ratio of position-velocity mismatches compared to the total number of position messages seen from an aircraft. Notably, there is a significant number of aircraft with high mismatches. Whenever these aircraft were seen by many sensors, we can safely say that they were real aircraft transmitting real ADS-B messages. However, if the position-velocity mismatch ratio is still very high, there might be an issue with the ADS-B equipment. We checked some examples manually and found that, for instance, some transponders have bugs and transmit a constant longitude of 0 (prime meridian), resulting in implausible positions.

Interestingly, 9 out of the worst 10 transponders (according to the position-velocity mismatch metric) belong to Aeroflot. In fact, the vast majority (45) of the 50 worst transponders belong to Aeroflot or other Russian carriers. This suggests that their aircraft have some issue with the fitted transponders.

D. *Invalid Transponder IDs*

In the third case study, we examine the phenomenon of ICAO 24-bit transponder IDs showing up in the sensor-provided data, which are not assigned to any real aircraft. This could happen due to either intentional or unintentional interference with the data, before or after reception.

The range of ICAO 24-bit transponder addresses is split into blocks which are assigned to countries (see [14], [15] for the full allocation). There are some unused blocks and we have seen receivers sending us suspiciously many unassigned transponder IDs. In this way, sensor operators could possibly include some sort of 'watermark' into their data without actually destroying the good data.

Most sensors have a percentage of invalid ICAO 24-bit transponder addresses of at most 50%. While this sounds much, the reader should keep in mind here that this metric is very prone to decoding noise since it produces many random transponder IDs.

Fig. 8 plots the percentage of invalid ICAO identifiers for each sensor against the percentage of tracked messages. These results emphasize again the fact that decoding noise is again an issue since all five SBS-3 stations are having a percentage close to 50%. This is further illustrated in Fig. 9, where again the SBS-3 sticks out with a high percentage of invalid data; across all sensors of this type, almost 50% of all seen ICAO transponder IDs were invalid. The similarity to Fig. 5 confirms that the main factor producing these high numbers of invalid transponder IDs is still decoding noise and not any malicious behaviour.

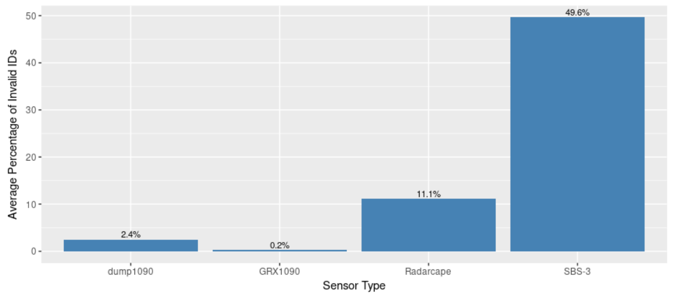

Figure 9: Average percentage of invalid ICAO 24-bit transponder IDs reported by each receiver type.

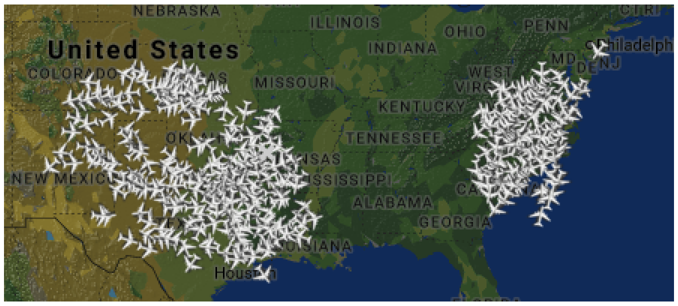

Figure 10: Example case of output from a 'multiple streams' sensor, where data from more than one sensor is being fed into Opensky Network as a single sensor.

The outlier in the top-right corner of Fig. 8 has an exceptionally high percentage of invalid ICAO IDs. More specifically, 28 of the 31 aircraft seen by this sensor have invalid transponder IDs. By looking closer at the reported transponder IDs, we found that the transponder IDs reported by this sensor followed a pattern. A deeper analysis of the raw data revealed that this sensor is located next to a Traffic Information Service-Broadcast (TIS-B) transmitter. In TIS-B, 24-bit transponder addresses are replaced by 12-bit Mode A addresses plus a track ID if a certain flag is set in the payload. This was the case for all seemingly invalid transponder IDs observed by this sensor. We therefore conclude that this sensor is not malicious, neither did it report false information. There is just a TIS-B transmitter nearby broadcasting these incompatible addresses. We also searched specifically for TIS-B messages and found more sensors seeing TIS-B messages from systematical transponder IDs. However, the other IDs were mostly mapped on valid ID ranges (Italy) which is why they did not appear in this analysis. Notably, all sensor which received TIS-B messages are located right next to a major US airport. For our analysis, however, we can ignore them as they do not pose an integrity breach.

E. Malformed Sensor Coverage

In this case, we present two examples of non-malicious integrity issues: randomized data and multiple sensors as one stream. Both present deliberate efforts to tamper with the data fed into Opensky, but neither show signs of being malicious in the sense of attempting to inject false but realistic data.

1) Data Randomization: Sensor x reports a typical number of tracked messages compared to nearby sensors, but has a coverage area close to three times larger than the sensor with the next largest coverage area. Naturally, this implies that the data reported by the sensor is problematic. On closer inspection, messages collected have callsigns replaced with random strings, yet does not modify the ICAO address. It also randomizes velocities and positions which then results in the unusually large coverage area of the sensor.

We can infer that this is deliberate tampering since individual messages are modified and the checksums are recalculated—something which should not be due to a software issue. Furthermore, the changes affect a specific set of data fields which is consistent across different tampered messages. This kind of message modification is problematic for the sensor network, especially since the ICAO address is the same. When fed into the network and merged with data from other sensors, it distorts the network-level view of ADS-B messages. Even though the message tampering is clearly random rather than deliberately modifying with realistic values, it causes wider integrity issues.

2) Multiple Sensors as a Stream: A different form of the same problem is where a sensor operator merges data collection from multiple sensors into a single feed, then passes this onto Opensky. An example of this can be seen in Fig. 10, where two US-based sensors are being fed into the network as a single sensor. We can identify this by both the coverage range and disparate centers of ADS-B message clusters. More extreme examples exist; one sensor feeds multiple sensors located across Europe and the US as a single sensor.

Whilst this approach to feeding a network provides a lot of data and is not malicious, it creates a number of integrity issues. Firstly, it removes the ability to verify messages based on coverage range, location and message rates, since there is no single location of the sensor to do this with. Furthermore, it reduces the ability to verify at the message level—the network operators can no longer rely on the reported timestamps especially in the case of multiple sensors spanning timezones.

VI. Data Handling and Countermeasures

Ideally, we wish to to filter as much problematic data as possible without loosing too much 'good' data. In order to do this we must ensure that any filtering techniques used are accurate. We now discuss how Opensky can handle the integrity issues described above, using different indicators for integrity breaches and derive adequate filter thresholds where possible.

A. Data Handling

In order to give the most accurate picture of the airspace, Opensky needs to filter out as many of the integrity issues as possible. However, the network provides a number of abstraction levels for different applications. Raw data, decoded data, state vectors and flight data are the provided layers in order of increasing abstraction. Importantly, not all of these need to be filtered for integrity, with the divide falling between the raw data and decoded data levels.

1) Raw Data: This should not be filtered for integrity at all, since it provides the most fundamental view of the received messages. Indeed, the examples given in Sec. V can be identified by looking at the raw data. On top of this, it is important to leave this type of data untouched so that research on other topics including receiver performance or crowdsourced networks can take place.

It is also worth noting that integrity countermeasures at this level are difficult without having direct control of the receiving hardware and software. As discussed by Strohmeier et al. in [16], using cryptographic integrity protection is also a significant challenge as it would require onboard avionics to be changed.

2) Higher Layers: Above raw data, layers are focussing on the information content of the messages. As such, these layers need to provide integrity where possible, but to do so we need logical countermeasures relevant to that layer. We discuss integrity approaches for these layers below.

B. Countermeasures

We can further divide countermeasures for integrity issues into *technical* and *organizational* measures.

1) Technical: We first outline the following three technical integrity countermeasures:

- **Network redundancy** allows cross-checking of message reception at a given sensor with other sensors. Within Opensky's total coverage area, 71% is covered by two or more sensors. This enables the network to check messages collected by a sensor against nearby sensors, identifying differences in content as well as whether or not the message was received.
- **Plausibility checks** on the data can significantly reduce the incidence rate of integrity problems by checking message content is within bounds with respect to the sensor. For example, if a sensor is observing messages originating from outside the typical coverage area, we can deduce that this sensor has lower integrity.
- **Sensor scoring** can be used to quantify the integrity of the data fed to Opensky by continuously running plausibility and integrity checks on said data. Of course, sensors which have lower scores will be able to increase it by fixing problems causing integrity issues.

2) Organizational: Naturally, these technical measures feed into to some organizational approaches which are enacted at the network operator level:

- **Trust modelling** based on the technical countermeasures, adjusted based on the tolerance for integrity issues by the network. Such a modelling approach would use both network redundancy and sensor scoring, and build on work in [17].
- **Sensor operator interaction** involves contacting sensor owners who are feeding the network with problematic data. This can be assessed based on on sensor scoring and trust modelling, with the primary aim being to gain information on their sensor setup and fix it.

Although the technical measures will need to be implemented before organizational, they can be done gradually to assess the impact of each.

VII. Discussion

As demonstrated in this paper, integrity issues are observed on ADS-B and Mode S data fed into the Opensky Network; it is highly likely that similar issues exist for all crowdsourced flight tracking platforms. So far, unintentional integrity breaches are under control on the Opensky Network. However, malicious integrity problems are likely to be a long-term arms race made more difficult by operating on a noisy channel. To defeat this we will then need some way of trusting receivers or inbuilt encryption.

One of the most promising ways to defend against external attackers trying to compromise integrity is through adding more receivers with overlapping coverage. This has seen success in other wireless localization scenarios such as GPS [18]. Both increasing sensor coverage redundancy and adding positional variety to the network makes it considerably more difficult for malicious actors to inject tampered data into the network. Such an approach will also help defend against individual malicious sensors trying to directly inject tampered data into the network.

Arguably the most significant threat to these participatory sensor networks is a coordinated insider attack, known as a Sybil attack [19], [20]. This category of attack involves attackers controlling a number of sensors on the network which initially operate benignly to build reputation. After some period of time, they collude in order to deceive the network. They do this in such a way to only outvote sensors which they overlap with, so as not to make such an attack obvious.

Under this kind of attack, data integrity is very hard to establish. One way to defend against this to individually verify each sensor joining the network (and those already added). Such an approach would increase the time and effort required to place enough sensors on the network to attack it. However, it would also adds significant workload to operating the network which only gets worse as the network scales up.

VIII. Conclusion

Clearly, there are a range of challenges to integrity for Mode S and ADS-B collection by crowdsourced sensor networks. Some of these cannot be easily controlled, such as environmental noise, whereas others come from misconfiguration. Regardless of cause, low integrity data is problematic for the network operator as it can cause significant inaccuracies in the view of the airspace.

Since deploying encryption on the Mode S link is unlikely to happen in the near-term future, we propose a range of technical and organization approaches to identify and defend against these integrity problems. Each of these approaches leverage the nature of crowdsourced networks, namely redundancy and ability to distributed, unpredictable positioning.

We do note, however, that so far there is no evidence of efforts by a malicious party to reduce data integrity—though with no cryptographic security in place, this is likely to be a matter of time.

REFERENCES

[1] M. Schäfer, M. Strohmeier, M. Smith, M. Fuchs, R. Pinheiro, V. Lenders, and I. Martinovic, “OpenSky Report 2016: Facts and Figures on SSR Mode S and ADS-B Usage,” in *IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sep. 2016.

[2] M. Schäfer, M. Strohmeier, M. Smith, M. Fuchs, V. Lenders, M. Liechti, and I. Martinovic, “OpenSky Report 2017: Mode S and ADS-B Usage of Military and other State Aircraft,” in *IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, Sep. 2017.

[3] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, “Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research,” in *Proceedings of the 13th IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN)*, Apr. 2014.

[4] M. Strohmeier, M. Schäfer, M. Fuchs, V. Lenders, and I. Martinovic, “Opensky: A swiss army knife for air traffic security research,” in *Proceedings of the 34th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, Sep. 2015.

[5] J. E. Boritz, “Is practitioners’ views on core concepts of information integrity,” *International Journal of Accounting Information Systems*, vol. 6, no. 4, pp. 260–279, 2005.

[6] R. Von Solms and J. Van Niekerk, “From information security to cyber security,” *computers & security*, vol. 38, pp. 97–102, 2013.

[7] R. Trüb, D. Moser, M. Schäfer, R. Pinheiro, and V. Lenders, “Monitoring meteorological parameters with crowdsourced air traffic control data,” in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, ser. IPSN, apr 2018. [Online]. Available: http://www.lenders.ch/publications/conferences/IPSN18_3.pdf

[8] B. Syd Ali, W. Schuster, W. Ochieng, and A. Majumdar, “Analysis of anomalies in ads-b and its gps data,” *GPS Solutions*, vol. 20, no. 3, Jul 2016.

[9] N. Christin, A. S. Weigend, and J. Chuang, “Content availability, pollution and poisoning in file sharing peer-to-peer networks,” in *Proceedings of the 6th ACM conference on Electronic commerce*. ACM, 2005, pp. 68–77.

[10] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena, “The pollution attack in p2p live video streaming: measurement results and defenses,” in *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*. ACM, 2007, pp. 323–328.

[11] A. Costin and A. Francillon, “Ghost is in the Air(traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices,” in *Black Hat USA*, Jul. 2012, pp. 1–12.

[12] M. Schäfer, V. Lenders, and I. Martinovic, “Experimental analysis of attacks on next generation air traffic communication,” in *International Conference on Applied Cryptography and Network Security (ACNS)*. Springer, 2013, pp. 253–271.

[13] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, “Short paper: reactive jamming in wireless networks: how realistic is the threat?” in *Proceedings of the fourth ACM conference on Wireless network security (WiSec)*. ACM, 2011, pp. 47–52.

[14] *International Standards and Recommended Practices, Annex 10 to the Convention on International Civil Aviation: Aeronautical Telecommunications*, 2nd ed., International Civil Aviation Organization (ICAO), 2007, Volume III: Communication Systems.

[15] International Civil Aviation Organization (ICAO), “Registration of Aircraft Addresses with Mode S Transponders,” Punta Cana, Dominican Republic, Tech. Rep. NACC/DCA/3, WP/05, Sep 2008.

[16] M. Strohmeier, V. Lenders, and I. Martinovic, “On the security of the automatic dependent surveillance-broadcast protocol,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1066 – 1087, 2015. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6940209

[17] M. Strohmeier, M. Smith, M. Schäfer, V. Lenders, and I. Martinovic, “Assessing the impact of aviation security on cyber power,” in *Proceedings of the 8th International Conference on Cyber Conflict (CYCON)*. IEEE, 2016.

[18] K. Jansen, M. Schäfer, D. Moser, V. Lenders, C. Pöpper, and J. Schmitt, “Crowd-gps-sec: Leveraging crowdsourcing to detect and localize gps spoofing attacks,” in *2018 IEEE Symposium on Security and Privacy (SP)*, vol. 00, pp. 189–202. [Online]. Available: doi.ieeecomputersociety.org/10.1109/SP.2018.00012

[19] J. R. Douceur, “The sybil attack,” in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[20] J. Newsome, E. Shi, D. Song, and A. Perrig, “The sybil attack in sensor networks: Analysis & defenses,” in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, ser. IPSN ’04. New York, NY, USA: ACM, 2004, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/984622.984660