A Machine Learning Approach for the Detection of Injection Attacks on ADS-B Messaging Systems

Joshua Price(1), Hadjar Ould Slimane(3), Khair Al Shamaileh(1), Vijay Devabhaktuni(2), Naima Kaabouch(3)

(1) Department of Electrical and Computer Engineering, Purdue University Northwest, Hammond, IN 46375, USA
(2) Department of Electrical and Computer Engineering, The University of Maine, Orono, ME 04469, USA
(3) School of Electrical Engineering and Computer Science, The University of North Dakota, Grand Forks, ND 58202, USA
E-mail: kalshama@pnw.edu

***Abstract*— This work proposes the use of machine learning (ML) as a candidate for the detection of various types of message injection attacks against automatic dependent surveillance-broadcast (ADS-B) messaging systems. Authentic ADS-B messages from a high-traffic area are collected from an open-source platform. These messages are combined with others imposing path modification, ghost aircraft injection, and velocity drift obtained from simulations. Then, ADS-B-related features are extracted from such messages and used to train different ML models for binary classification. For this purpose, authentic ADS-B data is considered as Class 1 (i.e., no attack), while the injection attacks are considered as Class 2 (i.e., presence of attack). The performance of the models is analyzed with metrics, including detection, misdetection, and false alarm rates, as well as validation accuracy, precision, recall, and F1-score. The resulting models enable identifying the presence of injection attacks with a detection rate of 99.05%, and false alarm and misdetection rates of 0.76% and 1.10%, respectively.**

***Index Terms*— Automatic dependent surveillance-broadcast (ADS-B), federal aviation administration (FAA), machine learning (ML), message injection, national airspace system (NAS).**

I. INTRODUCTION

ACCORDING to the federal aviation administration (FAA), air traffic organization services approximately 45,000 flights and 2.9 million passengers daily, making the United States national airspace system (NAS) one of the most complex worldwide [1]. To modernize NAS for accommodating such a high demand in traffic, the FAA has launched the next generation air transportation system, NextGen, which seeks to implement new technologies for improving aviation navigation and communication capabilities [2]. One such technology is the automatic dependent surveillance-broadcast (ADS-B) messaging system, which allows aircraft to broadcast their position to other aircraft and ground stations more frequently than radar systems [3, 4]. The FAA has mandated that as of January 2020, all aircraft operating in open airspace must be equipped with this system [5]. Australia, Canada, Europe, and Mexico, among other locations, also enforce mandates that require aircraft to be equipped with ADS-B capabilities to a lesser extent [6]. Considering that ADS-B messaging is utilized on such a scale, vulnerabilities to the associated cyberattacks pose a significant threat since ADS-B communications are broadcast openly and unencrypted by design [7]. Common vulnerabilities include jamming, eavesdropping, message modification, and message injection [8].

Message injection attacks are particularly hazardous, where attackers construct and transmit falsifying ADS-B messages to alter the flight path of the target aircraft, change its speed, or impose fake aircraft in proximity to the targeted aircraft. In addition, such attacks are relatively easy to launch, as the attack data must simply adhere to the ADS-B format to be picked up by receivers. Therefore, multiple ADS-B message injection detection approaches were investigated and proposed in literature, such as multilateration, Kalman filtering, and data fusion. Multilateration entails the deployment of sensors at the ADS-B receiver sites to approximate the adversary location by tracing an attack back to its point of transmission [9–11]. However, this approach requires sensor networks be installed at the various ADS-B receivers and is only reliable if the targeted aircraft can be located by at least four receivers [10]. Moreover, it may also be challenging to achieve optimum sensor placement in the intended deployment area. Kalman filtering can be used as a standalone method or in conjunction with other methods (e.g., multilateration) to detect message injection attacks, localize the attack source, track the position and velocity of a target aircraft, and even estimate the target's intended course [12, 13]. Nevertheless, the performance of Kalman filtering in attack detection greatly decreases in the attack scenarios that closely mirror authentic ADS-B data [11]. Data fusion uses data from both ADS-B and secondary surveillance radar (SSR) systems to approximate the location of an aircraft [14]. This approach, however, suffers from the differences in sampling rates and location accuracy between the two systems, leading to challenges in obtaining clear paths for the target aircraft in an attack event [11]. Machine learning (ML) offers solutions (i.e., trained models) that generally avoid many of these issues. Once such models are trained to detect certain types of attacks, constant access to live authentic data is no longer needed as opposed to data fusion. Furthermore, these models can be realized with minimal to no additional hardware implementations, unlike multilateration techniques.

In this work, ML models for binary classification are developed and evaluated. Authentic samples are considered as Class 1, and three types of message injection attacks; namely, path modification, ghost injection, and velocity drift are considered as Class 2. Binary classification is beneficial in the case of detecting ADS-B message injection attacks since a specific attack class does not need to be known to raise alarm.

```
Algorithm: Path Modification (PM) and Velocity Drift (VD) Sample Generation
Given: Dataset          -Dataset of ADS-B samples
       R_Earth          -Radius of Earth in meters
       Drift            -Drift angle for PM attack in degrees
       NbTotalAttack    -Number of attack samples to generate
       minVelocity      -Minimum velocity for VD attack
       maxVelocity      -Maximum velocity for VD attack

1: Procedure: Attack_Generation_VD_PM()
2: Loop: for each target address, do:
3:    prevOrigData =list()
4:    index          = 0
5:    nbAttack       = 0                // Initialize number of attacks generated
6:    attackRatio    = floor((num_samples * 4) / 10)
7:    midIndex       = floor(num_samples / 3)
8:    if(midIndex <= 1):                // Decide whether to start attack
9:       attackStart = 1
10:   else:
11:      attackStart = rand_int(1, midIndex)
12:   Loop: for each sample, do:
13:   if ((index >= attackStart) or (atkRatio < nbAttack) or (class == authentic)):
14:      nbAttack        += 1
15:      NbTotalAttack -= 1
16:      location1        = (prevLat, prevLon)
17:      location2        = (currenLat, currentLon)
18:      if(attack == velocityDrift)     // If performing VD attack
19:         time          = 10
20:         newVelocity = (2 * prevVelocity)%maxVelocity
21:         dist   = newVelocity * time   // Compute distance with new velocity
22:         label = 3
23:      else                            // If performing PM attack
24:         dist         = Haversine(location1, location2) //Find haversine distance
25:         driftAngle = (currentHeading + drift) % 360 // Change target's headng
26:         currentHeading = driftAngle           // Update aircraft heading
27:         label =1
28:     newLat = calcLat(prvAtkLat, driftAngle, R_Earth, dist)
29:     newLon = calcLon(prvAtkLon, prvAtkLat, currentLat, driftAngle, R_Earth, dist)
30:     attackSample = currentSample with updated attack parameters
31:     prevOrigData = currentSample.tolist()
32:     prevAttackSample = attackSample
33:     write attackSample to dataset
34:     index += 1
35:   end for
36: end for
```

Fig.1: Pseudocode for generating the ADS-B feature samples for the path modification and velocity drift cyberattacks.

The remainder of this paper is organized as follows: Section II discusses the collection of authentic ADS-B messages and the generation of the injection attack data. It also elaborates on the extracted signal features and suggests a qualitative analysis of the three injection attack types. Section III presents the results from the ML modeling. This includes the optimal hyperparameters and the performance evaluation of each model with respect to various metrics, such as the detection rate, recall and F1-score. Section IV summarizes these findings and concludes this work.

II. ADS-B MESSAGE COLLECTION AND FEATURE EXTRACTION

Authentic ADS-B messages are acquired from OpenSky, an open-source platform with records of ADS-B communications from sensor networks [15]. Collected messages are within a one-hour timeframe in a 50-km radius centered at the John F. Kennedy International Airport, with one aircraft identified as target. Python scripts are developed to generate the ADS-B messages that correspond to the aforementioned attack types (i.e., path modification, velocity drift, ghost injection). Each of these types is a form of a message injection cyberattack and seeks to disrupt the target aircraft in a different way. Path modification is concerned with altering the angle at which the target travels, thereby modifying the intended route. Velocity drift entails changing the position of the target by gradually changing its velocity. Ghost aircraft injection attack seeks to create multiple fake or 'ghost' aircraft in the vicinity of the target.

The developed Python scripts import the original dataset containing the authentic ADS-B data and defines parameters relative to the specific attack. For example, the generation of the path modification requires the drift angle with respect to true north that the attack intends to induce, while the generation of velocity drift requires the minimum and maximum velocities to be specified relative to the target. Figure 1 shows a pseudocode for generating the attack samples for both velocity drift and path modification cyberattacks. Path modification samples are generated after computing new latitude and longitude coordinates using the desired drift angle, target aircraft heading, and the original latitude and longitude values. Velocity drift attack samples are generated by computing new location coordinates using a range of velocity values specified for the attack. It is worth noting that the velocity is varied gradually as attack samples are generated so that the samples appear to be legitimate. The estimated distance traveled based on this velocity is used to determine the new location coordinates. In both attacks, sample generation continues using authentic data and previously generated attack samples to derive new attack messages. These are then appended to the dataset with the appropriate label. Finally, ghost aircraft injection requires a range of values for each ADS-B feature that will be used to generate samples that appear to retain proximity to the original aircraft. This range is defined by the selected attack radius and the features obtained from authentic ADS-B messages. Table 1 lists the relevant parameters for the ghost aircraft injection attack. The values of these parameters are selected such that a ghost aircraft is injected within the specified attack boundary (i.e., circle of 50-km in radius). Figure 2 represents a pseudocode for generating such an attack, which makes use of the parameters listed in Table I.

TABLE I
GHOST AIRCRAFT INJECTION ATTACK PARAMETERS

Parameter Pair	Description	Value	Unit
timeAttackStart/End	Determines the start and stop of the attack samples generation	–	–
nbTotalAttack	Total number of attack samples to generate	5,579	–
min/maxRange	Minimum and maximum distances from target in which attack will occur	6000 (Min) 10000 (Max)	Meters
min/maxDist	Minimum and Maximum distances a ghost aircraft can be from the target	500 (Min) 3000 (Max)	Meters
min/maxVel	Minimum and maximum velocity values	1 (Min) 300 (Max)	Meters/sec
min/MaxVert	Minimum and maximum vertical rate	–28 (Min) 37 (Max)	Meters/sec

```
Algorithm: Ghost Aircraft Injection (GA) Sample Generation
Given: Dataset          -Dataset of ADS-B samples
       R_Earth          -Radius of Earth in meters
       Table 1          -Contains list of parameters used in the attack

1: Procedure: Attack_Generation_GA()
2:  listAttackAircraft = list()
3: n                    = nbTotalAttack
4: rem                  = 199
5: Loop: for each target address, do:
6:   Loop: for each attack sample, do:
7:       time=data['time']              //Get time from authentic sample
8:       if (time >= timeAttackStart) and (time <= timeAttackEnd):
9:           nbAttackPerPoint = 20
10:          if(randFloat0to1( ) < 0.5) and not (rem == 0)
11:            nbAttackPerPoint += 1
12:            rem –= 1
13:          if(randFloat0to1( ) < 0.4) and not (rem == 0)
14:            nbAttackPerPoint += 1
15:            rem –= 2
16:          if(time == timeAttackEnd):
17:            nbAttackPerPoint = n
18:          if(n > 0):
19:            attackSampleTime = time
20:          for each attack per point:
21:            address = currentAddress
22:            dist = minRange + (minRange-maxRange)*randFloat0to1( )
23:            driftAngle = 360 * randFloat0to1( )
24:            if (randFloat0to1( ) < 0.5):
25:                drift = –drift
26:            driftAngle    =  (currentHeading+drift) % 360
27:            newLat        =  calcLat(curLat, driftAngle, R_Earth, dist)
28:            newLon        = calcLon(curLon, driftAngle, R_Earth, dist, newLat)
29:            newVelocity = minVel+(maxVel-minVel)*randFloat0to1()
30:            newVertrate =  minVert+(maxVert-minVert)*randFloat0to1()
31:            newHeading = driftAngle
32:            if(currentOnground == 1)
33:              newOnground = 1
34:            else
35:              newOnground = 0
36:            altDrift      = minDist+(maxDist-minDist)*randFloat0to1( )
37:            if(randFloat0to1() < 0.5) :
38:              altDrift      = –altDrift
39:            newBalt       = currentBalt+altDrift
40:            label          = 2
41:            attackSample = vector of the newly calculated features
42:            write attackSample to dataset
43:            listAttackAircraft.append(attackSample)
44:          end for
45:   end for
46: end for
```

Fig. 2: Pseudocode for generating the ADS-B feature samples for the ghost injection cyberattack.

Multiple points within the desired attack radius are selected, and a specified number of samples are generated for each point. The attack samples are written to the dataset, and sample generation continues until the desired number of attack samples are met or the defined attack timeframe has expired. Each of the three attack types differ further in complexity and severity, with complexity referring to the difficulty with which the attack is launched and the severity representing the overall impact on the target. Ghost aircraft injection is the simplest to launch, as the main requirement to be addressed is to transmit ADS-B messages of legitimate flight data. This attack confuses both pilots and ground stations, leading to potentially dangerous course corrections and altitude or speed adjustments [16].

TABLE II
MESSAGE INJECTION QUALITATIVE ANALYSIS: LAUNCH COMPLEXITY AND ATTACK SEVERITY

Complexity		**Severity**		
		1	**2**	**3**
	1	▨	▨	Ghost Injection
	2	Velocity Drift	Path Modification	▨

Both path modification and velocity drift attacks have a similar launch complexity and require precise timing to impact the target. Finally, velocity drift has the lowest severity since it alters only the velocity and leaves other information intact. Considering the relative ease with which a ghost aircraft attack is launched and the greater impact a successful attack has, it is more likely to be encountered by an ADS-B receiver. Table II summarizes these differences in complexity and severity, with scores of 1 and 3 being the lowest and highest, respectively.

Once authentic and unauthentic ADS-B messages are collected, features are extracted into a dataset. These features include the *latitude*, *longitude*, *barometric altitude, velocity*, *heading*, *vertical rate*, and *on-ground*. The *latitude* and *longitude* are in degrees, the *barometric altitude* is in meters, and the *velocity* is in meters per second. The *heading* presents the clockwise direction, in degrees, the aircraft is facing with respect to true north; whereas the *vertical rate* of the aircraft, in meters per second, represents its decline and recline rates. Finally, the *on-ground* indicates if the aircraft is grounded or not. Two extra features are obtained by calculating the received signal strength (RSS) and the Doppler shift as follows [18]:

$$Doppler = f_R\text{-}f_T \qquad (1.a)$$

$$RSS = \frac{P_T G_t G_r \lambda^2}{(4\pi)^2 d^2} \qquad (1.b)$$

where f_T and f_R in (1.*a*) are the transmitted and received signal frequencies, respectively. Here, $f_R = f_T \cdot (c+v_R)/(c+v_T)$ where c is the speed of light, v_R is the receiver velocity, and v_T is the transmitter velocity. The transmitted frequency is 1090 MHz (i.e., ADS-B standard) and the received frequency is in the range of 1090 MHz ± 1150 Hz. The parameters P_T, G_T, G_R, λ, and d in (1.*b*) represent the transmitted signal power, transmitter antenna gain, receiver antenna gain, the wavelength, and the transmitter-receiver separation distance, respectively. The transmitted signal power is set to 200 Watts, whereas the transmitter and receiver gains are set to10 dBi each. As a result, the final dataset contains a total of ten features: seven obtained from the ADS-B messages, the calculated RSS, the calculated Doppler shift, and the '*label*' of each sample with respect to its signal class (i.e., Class 1 for authentic transmission, Class 2 for unauthentic transmission).

The resulting dataset is preprocessed by eliminating the feature samples with a null value for any of their features and by removing the duplications, which are attributed to the multipath effect and the errors in the ADS-B transponders, causing them to transmit multiple instances of the same message [17]. In addition, the samples in all features are standardized to avoid divergence in the process of ML training.

TABLE III
PERFORMANCE METRICS FOR BINARY CLASSIFICATION

Classifier	VA(%)	DR(%)	Precision	Recall	F1-score	FAR(%)	MDR(%)	TT(ms)	PT(ms)
LR	86.8748±0.9291	86.3938	0.86	0.87	0.86	10.13	16.42	10252.17	0.998
KNN	92.7118±0.7628	92.4389	0.92	0.92	0.92	7.67	7.47	27.92	42.89
DT	97.7187±0.4338	97.8776	0.98	0.98	0.98	2.33	1.95	91.14	0.99
NB	86.5508±0.8233	86.3938	0.87	0.86	0.86	9.24	17.14	4.02	0.99
RFC	**99.001 ± 0.2832**	**99.0525**	**0.99**	**0.99**	**0.99**	**0.76**	**1.10**	**1601.01**	**57.85**
MLP	86.5104±2.6823	87.1707	0.89	0.87	0.87	4.41	19.64	2000.81	3.99

Finally, correlation analysis of features is carried out using the Pearson algorithm as demonstrated in Figure 3, which suggests that no dimensionality reduction is necessary. It is noteworthy to point out that a correlation of $|c| > 0.7$ is selected to determine if two feature pairs are correlated. The finalized dataset conveys 17,590 samples (i.e., 7,978 authentic samples, 9,612 attack samples). Of the attack samples, 1,911 belong to path modification, 5,579 are for ghost aircraft injection, and the remaining 2,122 are for velocity drift. The dataset is split into 70% and 30% in training and testing, respectively, and 10-fold cross validation is performed during model training/validation.

III. ML Modeling and Performance Evaluation

Six ML models are trained and validated for detecting the message injection attacks mentioned earlier. These models are logistic regression (LR), K-nearest neighbors (KNN), decision tree (DT), Gaussian naïve-Bayes (NB), random-forest classifier (RFC), and multi-layer perceptron (MLP). Grid search is performed with the use of the developed dataset to obtain the optimum hyperparameters for each model. Several metrics are used for performance evaluation, including detection rate (DR), precision, recall, F1-score, false alarm rate (FAR), and misdetection rate (MDR). These metrics are calculated as follows:

$$\text{DR}=\frac{TP+TN}{TP+FP+TN+FN} \tag{2.a}$$

$$\text{Precision}=\frac{TP}{TP+FP} \tag{2.b}$$

$$\text{Recall}=\frac{TP}{TP+FN} \tag{2.c}$$

$$\text{F1-Score}=\frac{2\times Precision\times Recall}{Precision+Recall} \tag{2.d}$$

$$\text{FAR}=\frac{FP}{FP+TN} \tag{2.e}$$

$$\text{MDR}=\frac{FN}{FN+TP} \tag{2.f}$$

where *TP*, *TN*, *FP*, and *FN* are the number of true positive predictions, true negative predictions, false positive predictions, and false negative predictions, respectively. The DR measures the percent of samples which are classified correctly in testing stage.

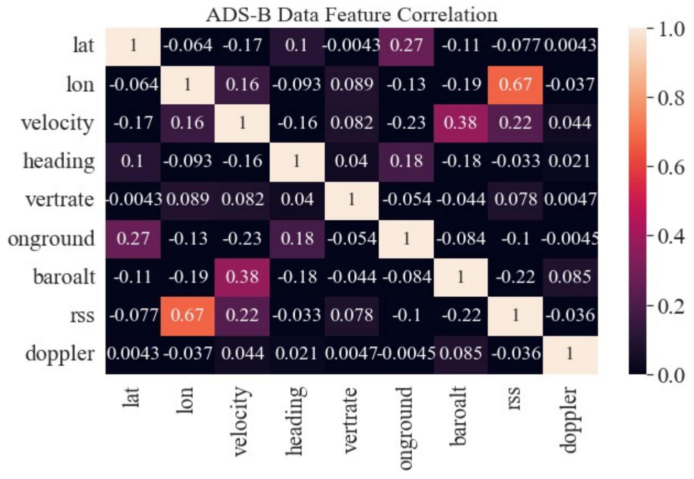

Fig.3: Correlation heatmap for the ADS-B features Dataset.

The precision determines the overall quality of the positive and negative predictions made by the model, while the recall determines the ability of the classifier to correctly identify positive samples. The F1-score metric helps to further analyze the incorrectly classified samples and is calculated as the harmonic mean of the precision and recall. The FAR calculates the percentage of samples that are incorrectly classified as positive, while the MDR calculates the percentage of positive samples which are incorrectly classified as negative.

Table III presents the performance of the models with respect to (2.*a*–*f*) as well as the resulting training time (TT) and prediction time (PT). A Windows 10 PC with an Intel i7-7700HQ CPU @ 2.80 GHz and 32 GB of DDR4-2400 MHz memory is used for training, validating, and testing the ML models. LR, NB, and MLP have comparable performance, with validation accuracy (VA) and DR close to 86%. KNN achieves better performance, characterized by VA and DR of around 92.5% each. In addition, its precision, recall, and F1-score are all found to be 0.92. On the other hand, it is found that the best performing models are the RFC and DT, with the former attaining the optimum VA and DR of approximately 99%. Furthermore, the resulting FAR and MDR for the RFC model are 0.76% and 1.10%, respectively, indicating low incidences of misdetection and false alarms. It is also worth noting that the TT and PT of the RFC model are 1,601 ms and 57.85 ms, respectively, enabling real-time detection of the investigated injection attacks. Finally, Table IV shows the optimal grid-search based hyperparameters for each model. The optimization of all these models is performed with Scikit-Learn ML library.

TABLE IV
OPTIMIZED HYPERPARAMETERS FOR EACH CLASSIFIER

Classifier	Optimal Hyperparameters
LR	Maximum number of iterations: 1000 Norm used for penalty: L2 Optimization algorithm: Coordinate Descent Regularization constant: 0.1
KNN	Distance metric: Manhattan Nearest neighbor algorithm: K-D Tree Number of neighbors: 5 Weight function: Distance
DT	Maximum tree depth: None Maximum number of leaf nodes: None Node split strategy: Best Quality of split criterion: Entropy
NB	Smoothing parameter for stability: 1E–8
RF	Number of trees: 100
MLP	Activation function: Rectified Linear Unit Hidden layer size: 100 Learning rate: Constant Solver for weight optimization: Adam Strength of L2 regularization term: 0.0001

IV. CONCLUSION

To conclude, ADS-B communications are particularly vulnerable to attacks as information is openly broadcast to ground stations and other aircraft. Message injection attacks allow attackers to broadcast false information, which can alter the flight path of aircraft or falsely suggest that other aircraft are present. A ML method is therefore proposed to predict the presence of three ADS-B message injection attacks: path modification, ghost aircraft injection, and velocity drift, through binary classification. Samples obtained from a dataset containing authentic ADS-B data and message injection attack data are used to train and validate six ML models. Evaluation metrics such as the VA, DR, FAR, and MDR are used to compare the performance of the developed models. This ML approach produces a RFC model capable of accurately predicting the presence of any of the three attacks with 99% confidence, associated with a FAR of 0.76% and MDR of 1.10%.

ACKNOWLEDGMENT

This research is funded by the National Science Foundation, Secure and Trustworthy Cyberspace under Award no. 2006662.

REFERENCES

[1] Air traffic by the numbers. [Online]. Available: https://www.faa.gov/air_traffic/by_the_numbers/

[2] Next Generation Air Transportation System (NextGen). [Online]. Available: https://www.faa.gov/nextgen

[3] Automatic Dependent Surveillance - Broadcast (ADS-B). [Online]. Available: https://www.faa.gov/about/office_org/headquarters_offices/avs/offices/afx/afs/afs400/afs410/ads-b

[4] Equip ADS-B: Ins and outs. [Online]. Available: https://www.faa.gov/air_traffic/technology/equipadsb/capabilities/ins_outs/#:~:text=ADS%2DB%20Out,can%20immediately%20receive%20this%20information.

[5] Frequently asked questions. [Online]. Available: https://www.faa.gov/air_traffic/technology/equipadsb/resources/faq/#:~:text=The%20rule%20dictates%20that%20after,meeting%20requirements%20defined%20in%2091.227. Where is ADS-B out required? [Online]. Available: https://www.aopa.org/go-fly/aircraft-and-ownership/ads-b/where-is-ads- b-out-required

[6] Z. Wu, T. Shang, and A. Guo, "Security issues in automatic dependent surveillance-broadcast (ADS-B): A survey," *IEEE Access*, vol. 8, pp. 122147–122167, 2020.

[7] A. Fatimah, A. Amal and H. Nermin, "Effective security techniques for automatic dependent surveillance-broadcast (ADS-B)," *International Journal of Computer Applications*, vol. 180. pp. 23-28.

[8] D. Ala, B. Evangelos, T. Brice, and P. Christina, "On ADS-B sensor placement for secure wide-area multilateration", *MDPI Proceedings*, vol. 59, no. 1, p. 3, 2020.

[9] M. Monteiro et al., "Detecting malicious ADS-B broadcasts using wide area multilateration," *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC),* 2015, pp. 4A3-1 - 4A3-12

[10] T. Li, and B. Wang, "Sequential collaborative detection strategy on ADS-B data attack," *International Journal of Critical Infrastructure Protection*, vol. 24, no. C, pp.78-99, 2019.

[11] M. Leonardi and G. Sirbu, "ADS-B crowd-sensor network and two-step Kalman filter for GNSS and ADS-B cyber-attack detection," *Sensors*, vol. 21, no. 15, p. 4492, 2021.

[12] K. Birendra, K. Samer and P. Boris, "Detecting GNSS spoofing of ADS-B equipped aircraft using INS," *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS),* 2020, pp. 548-554

[13] T. Yong, W. Honggang, X. Zhili, and H. Zhongtao, "ADS-B and SSR data fusion and application," *IEEE International Conference on Computer Science and Automation Engineering* (*CSAE*), 2012, pp. 255–258.

[14] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up OpenSky: A large-scale ADS-B sensor network for research," *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, 2014, pp. 83–94.

[15] M. Schäfer, V. Lenders, and I. Martinovic, "Experimental analysis of attacks on next generation air traffic communication," *Applied Cryptography and Network Security*. Springer, Berlin, Heidelberg: https://doi.org/10.1007/978-3-642-38980-1_16

[16] M. Strohmeier, M. Schäfer, V. Lenders, and I. Martinovic, "Realities and challenges of nextgen air traffic management: the case of ADS-B," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 111–118, 2014.

[17] S. Kurt and B. Tavli, "Path-loss modeling for wireless sensor networks: A review of models and comparative evaluations," *IEEE Antennas and Propagation Magazine*, vol. 59, no. 1, pp. 18–37, 2017.