Distributed and Parallel Databases (2021) 39:231–257
https://doi.org/10.1007/s10619-020-07302-1

GPS spoofed or not? Exploiting RSSI and TSS in crowdsourced air traffic control data

Gaoyang Liu[1] · Rui Zhang[2] · Yang Yang[3] · Chen Wang[1] · Ling Liu[4]

Published online: 24 June 2020

Abstract

GPS-dependent localization, tracking and navigation applications have a significant impact on the modern aviation industry. However, the lack of encryption and authentication makes GPS vulnerable for spoofing attacks with the purpose of hijacking aircrafts or threatening air safety. In this paper, we propose GPS-Probe, a GPS spoofing detection algorithm which leverages the air traffic control (ATC) messages periodically broadcasted by aircrafts. By exploiting the received signal strength indicator (RSSI) and the timestamps at server (TSS) of the ATC messages monitored by multiple ground sensors, GPS-Probe constructs a machine learning enabled framework which can estimate the real position of the target aircraft and then detect whether GPS is spoofed or not. Unlike existing techniques, GPS-Probe neither requires any updates of the GPS infrastructure nor of the GPS receivers. It also releases the requirement on the time synchronization of the ground sensors distributed around the world. We further present GPS-Probe-Plus by incorporating a flight height estimation module and a calibration method for RSSI and TSS values, which performs better on both target localization and spoofing detection than GPS-Probe. Using the real-world ATC data crowdsourced by OpenSky Network, our experiment results show that GPS-Probe (resp. GPS-Probe-Plus) can achieve an average detection accuracy and precision, of 81.7% (resp. 86.8%) and 85.3% (resp. 91.2%), respectively, significantly outperforming the state-of-the-arts.

Keywords GPS spoofing attack · Air traffic control data · OpenSky Network · Adaptive *k* nearest neighbor · XGBoost

✉ Chen Wang
chenwang@hust.edu.cn

Extended author information available on the last page of the article

1 Introduction

In recent years, the Global Localization System (GPS) has become a ubiquitous source of localization, tracking, and navigation for more than a billion devices [1–5]. Among the fields of GPS deployments, the aviation industry is one of the earliest civil areas, and its reliance on GPS increases rapidly due to the requirement of navigation and air traffic monitoring. With the dramatic growth of Unmanned Aerial Vehicles (UAVs), GPS is mission-critical not only for airplanes but also for UAVs, ranging from consumer-class drones to tactical and strategic UAVs [6].

Unfortunately, the civilian GPS signals sent by the satellites are neither authenticated nor encrypted [7–9]. A malicious transmitter can easily imitate the legitimate GPS signals and emit fake signals with a higher power or slightly different time delays. Aircrafts' GPS receivers are vulnerable to signal spoofing attack as the spoofed GPS signal arrives with a higher signal strength than the real ones. As a consequence, the aircraft's position calculated with the spoofed signals can be tampered by the attackers arbitrarily [10–13]. One best-known real-world case was happened in 2011, when a CIA stealth UAV (RQ-170) was captured by Iranians, who hijacked its GPS coordinates and safely brought it down.[1]

Driven by the increasing threats, a number of countermeasures have been developed, which generally fall into two class. The first class aims at authenticating signals from satellites by adding extra signals that are unpredictable to attackers [14–16]. However, these methods require a hardware upgrade of the GPS infrastructure and can be cracked easily by replay attacks. The second class focuses on the spoofing detection at signal-processing level, which utilize special-designed GPS receivers with adapted signal processing techniques and enhanced electronic circuits. With customized receivers, spoofing attacks can be detected from more relevant information, such as the angle-of-arrival of GPS signals [17], random antenna motion [18, 19], and automatic gain control on the radio frontend [20]. Nevertheless, most if not all existing countermeasures require far-reaching modifications of either the GPS infrastructure or the receiving devices, yielding them unlikely to be deployed in the near future.

In response to provide short-term practical solutions, the latest work dubbed Crowd-GPS-Sec proposes to leverage crowdsourcing to monitor the air traffic control (ATC) messages periodically broadcasted by aircrafts [21]. Crowd-GPS-Sec first estimates the position of the target aircraft based on the time difference of arrival (TDoA) of the ATC messages received by different ground sensors. By testing out the inconsistencies between the estimated position and the GPS-derived position contained in the ATC messages, GPS spoofing attacks can thus be detected without the need to update neither the GPS infrastructure nor the GPS receivers. One major challenge that hinders the application of Crowd-GPS-Sec, however, is its high reliance on precise time synchronization of the ground sensors, which is also the inherent limitation of TDoA-based localization technologies [22]. As the ATC

[1] https://en.wikipedia.org/wiki/Iran-U.S._RQ-170_incident.

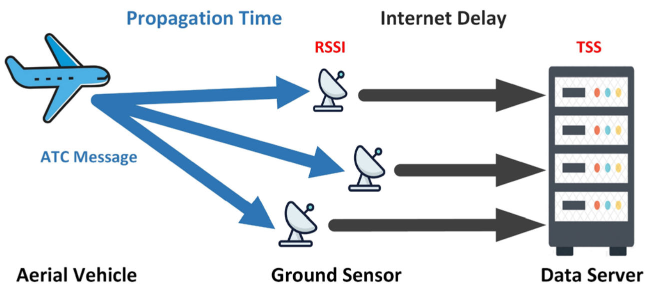


Fig. 1 Schematic of the crowdsourced ATC data transmission. ATC messages are periodically broadcasted by the aircraft, and can be received (along with the RSSI) by the ground sensors. TSS provides synchronized arrival times of one specific ATC message after being received and forwarded by different ground sensors. Both RSSI and TSS are leveraged as features for GPS spoofing detection

messages are spread approximately 300,000 km/h, a small time offset will lead to a large localization error, which in turn degrades the attack detection accuracy on the whole.

In this paper, we take a step forward and propose GPS-Probe, a GPS spoofing detection scheme which releases the requirement on the time synchronization of the ground sensors. The key insight is that, instead of using TDoA, GPS-Probe leverages the *received signal strength indicator (RSSI)* monitored by ground sensors and the *timestamps at server (TSS)* in ATC data as features (see Fig. 1), and relocates the target aircraft via a set of machine learning techniques. The estimated position is then utilized for GPS spoofing detection by analyzing its difference from the GPS-derived position.

We further develop GPS-Probe-Plus which can estimate the flight height of the target aircraft, and calibrate the RSSI and TSS values of the ATC messages received by OpenSky Network. With the calibrated RSSI and TSS, GPS-Probe-Plus can thus achieve better performance on both target localization and spoofing detection than GPS-Probe.

We summarize the major contributions as follows:

- We propose the first work to take advantage of RSSI and TSS features of crowdsourced ATC data for GPS spoofing detection, and design two algorithms leveraging machine learning techniques with these features. This leads to fundamental changes of traditional TDoA-based technique in the sense of releasing the time synchronization requirement.
- We propose an adaptive *k* nearest neighbor classifier named Ada-NN, aiming to suppress the noise in RSSI and to mitigate the errors of TSS-based localization, which further helps improve the attack detection performance.
- We evaluate our methods with real-world data from OpenSky Network[2] [23] which maintains a network of more than 800 ground sensors around the world

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

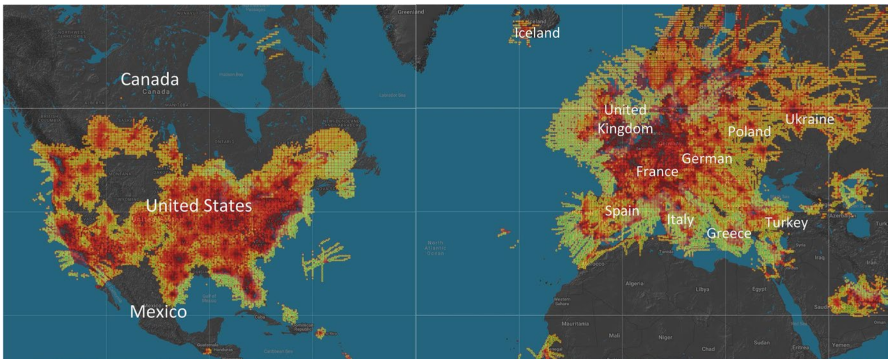


Fig. 2 Worldwide coverage of OpenSky Network

(see Fig. 2). The results show that the proposed techniques are able to detect GPS spoofing attacks globally in less than 5 s, and the detection accuracy can achieve 90% with attack range of 10 km.

The remainder of this paper is organized as follows. In Sect. 2, we provide some preliminary knowledge. In Sect. 3, we present more details on the design of GPS-Probe. In Sect. 4, we elaborate the improved version GPS-Probe-Plus in detail. Section 5 evaluates both GPS-Probe and GPS-Probe-Plus. Section 6 reviews related work and finally Sect. 7 concludes the paper.

2 Preliminary

2.1 GPS

GPS is a satellite-based navigation network of 32 satellites located in the medium Earth orbit, and provides geo-location, velocity and time information to GPS receivers anywhere on or near the Earth [2]. By measuring the time of arrival from unobstructed satellites, GPS receivers can calculate the distances to each satellite, and then multilateration of those distances derives the position and the local time of the receivers.

GPS is typically used by pilots or UAVs' controllers for self localization and the technique is also used for remote air-traffic surveillance and collision avoidance applications. For the safety of air traffic, the aircrafts need to periodically broadcast position, heading and velocity advertisements to notify neighboring aircraft and ground sensors by the Automatic Dependent Surveillance-Broadcast (ADS-B) system[3] or the Flarm[4] system.

[3] http://www.ads-b.com/.

[4] https://flarm.com/.

Although GPS is widely deployed in the civilian aviation, the lack of encryption and authentication makes it vulnerable for spoofing attacks [8, 24]. Attackers can easily mitigate the legitimate signals of GPS satellites, modify the data of GPS signals, and transmit these deliberately crafted signals to the target location with a higher power. Whenever a GPS receiver locks on to the spoofed signals, the position advertisements of aircrafts derived from GPS can be tampered by attackers arbitrarily [10–12].

2.2 Crowdsourced ATC data

ATC [25] is a service provided by controllers who direct aircrafts passing through controlled airspace on the ground. For the purpose of collision avoidance and air traffic organization, the aircrafts periodically broadcast the ADS-B or Flarm messages to declare their flight status. These status messages received by ATC ground stations construct the so-called ATC data. Note that as the ADS-B and Flarm communication protocols are open-source to public, anyone can monitor and collect these ATC data with state-of-art soft defined radio devices.

OpenSky Network [23] adopted in this paper is a crowdsourcing initiative to collect ATC data and making the data available to the public. The ground sensors of OpenSky are most installed and operated by aviation enthusiasts and volunteers. The volunteers continuously monitor the ATC data, which are then sent to the data center via Internet. As of this writing, it collects more than 200,000 messages per second at peak time from over 800 ground sensors distributed all over the world (see Fig. 2). The coverage in all regions is constantly growing, yet a single sensor can receive GPS signals up to a distance of 700 km, allowing to cover the whole world with a few thousand ground sensors [26].

2.3 RSSI

Available in mainstream wireless signal measurements, RSSI represents the power present in a received radio signal after the propagation attenuation, and is widely used in radio frequency (RF) system to provide necessary information to adjust receiver antenna gain. RSSI of received signals is typically calculated as follows:

$$RSSI_{RF} = 10\log(P_r/P_t) \tag{1}$$

where P_r (resp. P_t) is the power of transmitted (resp. received) signal. In the free-space model which is widely used in line-of-sight RF localization [27], the following relationship holds:

$$P_r \propto P_t \cdot \left(\frac{1}{d}\right)^2 \tag{2}$$

where d is the distance between the RF transmitter and receiver. From Eqs. (1) and (2), we can observe that RSSI is correlated with the propagation distance. The shorter the propagation distance is, the higher RSSI value will be.

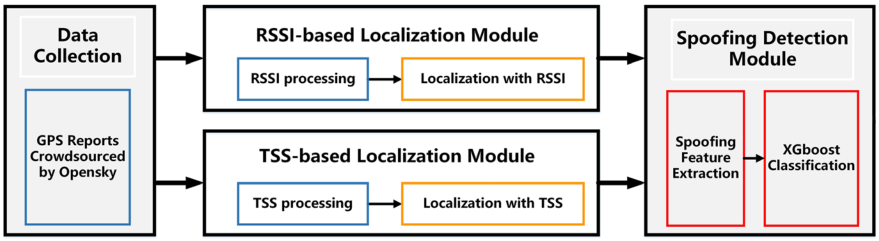

Data Collection
GPS Reports Crowdsourced by Opensky
RSSI-based Localization Module
RSSI processing
Localization with RSSI
TSS-based Localization Module
TSS processing
Localization with TSS
Spoofing Detection Module
Spoofing Feature Extraction
XGboost Classification

Fig. 3 System overview of GPS-Probe

As for OpenSky Network, the transmitters of the RF signals are the aircrafts which periodically broadcast the ATC messages, while the receivers are the ground sensors. The transmitting power of the ATC messages is determined by the electronic circuits on the aircrafts, and it is impossible to disturb the RF transmission modules by existing GPS spoofing attacks. After propagation, the ATC messages are received by the ground sensors of OpenSky Network, and the RSSI of the received signals are credible even the GPS information is modified by malicious attackers. As such, we can calculate the real geo-position of the target aircraft by leveraging the distance information hidden behind the RSSI.

3 GPS-probe design

The proposed GPS-Probe acts as an independent infrastructure on the ground which analyzes ATC messages and detects GPS spoofing attacks continuously. To this end, GPS-Probe mainly involves the following three modules (c.f. Fig. 3):

(1) *RSSI-based localization module* In this module, GPS-Probe leverages RSSI of ATC messages to estimate the position of the target aircraft. We first remove the outliers and noise of the RSSI measurements, and utilize the processed RSSI to construct a RSSI fingerprint map. We divide the surveillance area into $M \times N$ grids and construct the RSSI feature vectors corresponding to every grid by combining the original and processed RSSI. By training a model on the RSSI feature vectors and corresponding grid labels, GPS-Probe can determine which grid the target aircraft is in, and finally obtain the location estimation Loc_{RSSI} with the proposed Ada-NN classifier.

(2) *TSS-based localization module* In this module, we leverage TSS in ATC messages to estimate the position of the target aircraft. The key idea is to remove the Internet delays in TSS measurements, which is caused by the ground sensors transmitting ATC messages to the data center. Then we can construct TSS feature vectors corresponding to every grid in the surveillance area. After the preprocessing of TSS measurements, we take a similar approach to the RSSI-based localization module to obtain the position estimation Loc_{TSS}.

(3) *GPS spoofing detection module* Given the estimated positions Loc_{RSSI} and Loc_{TSS}, GPS-Probe leverages XGBoost [28] to train a binary detection model, with the

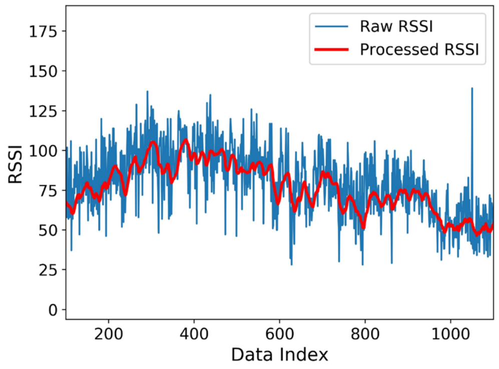


Fig. 4 RSSI preprocessing to remove the outliers and noise

deviation between the estimated positions and the GPS-derived position contained in the ATC messages as features. Consequently, for any aircraft, GPS-Probe monitors the RSSI and TSS in ATC messages in real time and detect whether or not the target aircraft is attacked by GPS spoofing.

3.1 RSSI-based localization

3.1.1 RSSI preprocessing

RSSI measurements of ATC messages obtained from ground sensors contain outliers and noises from various sources, such as multipath fading, transmission power adaptation at the transmitter side, and random amplitude errors, so we first remove the outliers and noise of the RSSI measurements. To remove the outliers, we simply utilize the Hampel identifer [29]. For denoising, we apply the classical yet simple sliding window averaging method, with a window width of 30 seconds on the raw RSSI series of one specific aircraft. The RSSI preprocessing enables us to remove the outliers and noise while keeping the effective information of the RSSI (c.f. Fig. 4).

3.1.2 Feature extraction

Given the processed RSSI measurements, and combining Eqs. (1) and (2), we can derive the following relationship:

$$10\exp(-RSSI/20) \propto d \tag{3}$$

where d is the distance from the aircraft to the ground sensor, and $RSSI$ is the signal strength indicator after preprocessing.

To improve the localization accuracy, we construct a new feature as follows:

$$RSSI_{linear} = 10\exp(-RSSI/20) \tag{4}$$

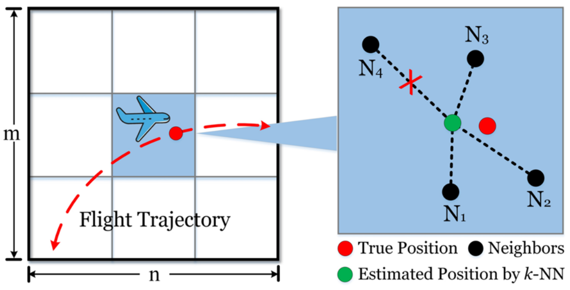


Fig. 5 A toy model of RSSI-based localization, which first determines the grid where the target vehicle is in (left), and then pinpoints the target with *k*-NN (right)

where $RSSI_{linear}$ is linearly dependent on the distance d according to Eq. (3). We emphasize here that, $RSSI_{linear}$ amplifies the $RSSI$ measurements exponentially, and provides more differences among adjacent locations which can help our machine learning models localize the target aircraft more accurately. For instance, if there are two received RSSI measures of neighboring aircrafts' GPS broadcasts which are 50 and 51 respectively, the corresponding $RSSI_{linear}$ will be 3162 and 3548, and thus the RSSI difference of 1 is amplified to 386. We combine $RSSI$ and $RSSI_{linear}$ together into a new feature vector $RSSV$ and leverage it to estimate the position of the target aircraft.

3.1.3 Localization with RSSI

We propose a grid-based *k*-NN algorithm for localization with RSSI, which has an offline training phase and an online localization phase. In the offline phase, we build $M \times N$ grids in the surveillance area as in [30], and construct $RSSV$ among the ground sensors for every position using $RSSI$ and $RSSI_{linear}$. By leveraging $RSSV$ and the corresponding grid labels, we train a standard *k*-NN classifier named $k\text{-NN}_{RSSI}$ that can determine the grid coordinates and the nearest neighbors of the location in the incoming ATC message.

In the online localization phase, the RSSI feature vector of an incoming ATC message is calculated from the original RSSI measurements at first. Using the classifier $k\text{-NN}_{RSSI}$, we obtain the closest grid from our training grid that matches the RSSI vector of an incoming ATC message as:

$$\begin{aligned} &\underset{(m,n)}{\arg\min}\ \frac{1}{I_{m,n}} \sum_{i=1}^{I_{m,n}} \left\| RSSV_{test} - RSSV^{i}_{(m,n)} \right\|_2 \\ &m \in [1, 2 \ldots M], n \in [1, 2 \ldots N], I_{m,n} \in \left[I_{1,1}, I_{1,2} \ldots I_{M,N}\right] \end{aligned} \tag{5}$$

where $RSSV_{test}$ is the feature vector of the target aircraft to be localized, and $RSSV^{i}_{(m,n)}$ is the i_{th} RSSI feature vector of the training data which locates in the (m, n) grid. $I_{(m,n)}$ is the number of training RSSI vectors which belong to the grid (m, n). Then we can obtain the predicted grid label of $RSSV_{test}$ (c.f. the left part of Fig. 5).

After getting the grid label (m, n), we next calculate the final location of the target aircraft. Averaging the locations of a certain number of nearest neighbors seems like a reasonable method. However, the accuracy of RSSI-based localization algorithm is

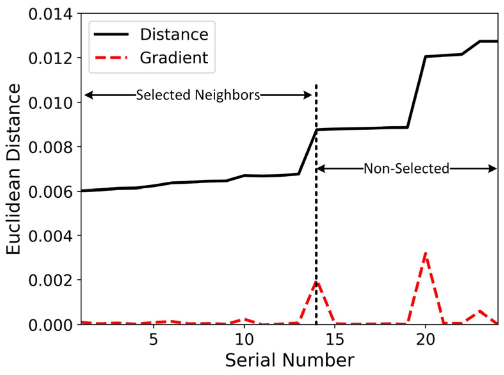


Fig. 6 Neighbor selection with *Distance* and *Gradient*

Table 1 Feature vectors, ML Algorithms and outputs of every modules

Module	Feature vector	ML Algorithm	Module output
RSSI localization	$RSSV = [RSSI,\ RSSI_{linear}]$	*k*NN + Ada-NN	Loc_{RSSI}
TSS localization	$TSSV = [TSS,\ TSS\_diff,\ SenMarker]$	*k*NN + Ada-NN	Loc_{TSS}
Spoofing detection	$DetV = [Loc_{RSSI}, Loc_{TSS}, Loc_{GPS}, Loc^{diff}_{RSSI}, Loc^{diff}_{TSS}, D^{geo}_{TSS}, D^{geo}_{RSSI}]$	XGBoost	Spoofed or not

influenced by the number of nearest neighbors severely, and it is difficult to choose a preset number which is just the best. As shown in the right part of Fig. 5, if we select 4 neighbors fixedly, the outlier N_4 will be included to obtain the estimated position and the localization result will deviate from the real position severely. On the contrary, estimating the position with averaging 2 neighbors still results in a relatively large error since the eligible point N_3 is excluded.

To tackle the above issue, we introduce an adaptive neighbor selection approach (denoted as Ada-NN) based on the gradient of Euclidean distance series between $RSSV_{test}$ and $RSSV_{(m,n)}$:

$$G_R = \nabla_i\, Sort\left(\left\| RSSV_{test} - RSSV_{(m,n)} \right\|_2\right) \tag{6}$$

where $\left\| RSSV_{test} - RSSV_{(m,n)} \right\|_2$ is the Euclidean distance series between $RSSV_{test}$ and the training vectors of the grid (m, n). *Sort* means sorting the distance series in ascending order, and then G_R is the derived gradient series corresponding to the distance series denoted.

To choose a proper number of nearest neighbors, we traverse the gradient series G_R to find the first local maximum [31]. In our adaptive neighbor selection, the serial number $K_{selected}$ corresponding to the local maximum serves as a demarcation point between the selected and non-selected neighbors. We take the training data whose serial numbers are less than $K_{selected}$ as the selected neighbors, and then average the positions of these neighbors to estimate the target's location. Figure 6 shows how Ada-NN works with the sorted distance series and corresponding gradient series G_R.

With Ada-NN, we can thus remove the outliers from the nearest neighbors while picking out all the appropriate neighbors from our testing RSSI feature vectors.

By now we have looked up sufficient nearest neighbors of $RSSV_{test}$, we can then average the geographic coordinates of the neighbors, finally yielding the position of the target aircraft denoted as Loc_{RSSI}, with its latitude Lat_{RSSI} and longitude Lon_{RSSI} (c.f. Table 1).

3.2 TSS-based localization

In the data obtained from OpenSky network, we find that nearly a quarter of RSSI measurements are missing. Thus the localization accuracy is restricted with only RSSI information. Fortunately, OpenSky also provides us TSS when the ATC messages arrive at the data server, and thus we can get the synchronized arrival times of one ATC message after being received and forwarded by different ground sensors. TSS is determined by the clock of data server, and therefore it cannot be modulated by spoofing attacks either. So we implement a supplementary localization module which leverages TSS to locate the aircrafts.

3.2.1 TSS preprocessing

TSS measurements provided by OpenSky Live API contain two parts of extra times to the transmission time (c.f. Fig. 1). One is the propagating time caused by GPS RF signals transmitting from the aircraft to the ground sensor. The other is the Internet delay introduced by the data transmission from the ground sensor to the OpenSky data server. Generally, the propagating time is tens or hundreds of microseconds, while the duration of Internet delay is tens of milliseconds. Thus the Internet delay overrides the fluctuation of propagation time caused by the distance changes between the aircraft and ground sensors.

For aerial localization with TSS measurements, we need to calculate the ATC message's propagation time through subtracting the transmitting time of the aircraft, and then remove the Internet delays. The transmitting time of the ATC message can be easily obtained by OpenSky Network. For Internet delay, we notice that its distribution approximately obeys the Weibull distribution [32]. So we remove the mean value of the Internet delay from TSS measurements, which makes TSS measurements relatively close to the transmitting time of ATC messages and thus can reflect more accurate distance between the signal source and the ground sensors.

3.2.2 Feature extraction

Even we remove the Internet delay in advance, however, it is still not easy to get the precise transmitting time. Hence, in our TSS-based localization module, we construct several new features to improve the localization performance.

At first, we make use of the difference between any two TSS measurements among different ground sensors as the additional features of different locations:

$$\begin{aligned} TSS_{diff} &= [T_{1,1}, T_{1,2}, \dots, T_{i,j}, \dots, T_{Q,Q}] \\ T_{i,j} &= T_i - T_j \;\; i,j \in [1,2,\dots Q] \;\; Q \in [1,2,\dots,P] \end{aligned} \tag{7}$$

where $T_{i,j}$ is the difference of TSS measurements obtained at *i*th and *j*th receiving ground sensors. *P* is the total number of the ground sensors and *Q* is the number of ground sensors receiving the ATC messages. TSS_{diff} implies the order in which one ATC message arrives at different ground sensors, and the order can represent the relative position of the aircraft to the receiving ground sensors.

In addition, we construct the feature vector *SenMarker*, a $P \times 1$ vector, which marks the receiving sensors of each ATC message. If an ATC message is received by p_{th} sensor, we will set the corresponding value in *SenMarker* to 1, while other values still maintain 0.

We combine the TSS measurements, TSS_{diff} and *SenMarker* together into a new feature vector *TSSV*, as the fingerprints of the aerial positions derived from the ATC messages, and then leverage *TSSV* to estimate the location of the target aircraft.

3.2.3 Localization with TSS

Similar as RSSI-based localization, TSS-based localization module also has two phases. In the offline training phase, we use the same $M \times N$ grids. By using *TSSV* of different locations, we train a *k*-NN classifier $k\text{-NN}_{TSS}$ that can determine the grid coordinates and the nearest neighbors in the training data set.

In the online localization phase, *TSSV* of an incoming ATC message is calculated from TSS obtained by the OpenSky Live API. Then we get the grid label (*m*, *n*) of the incoming message by $k\text{-NN}_{TSS}$:

$$\begin{aligned} &\underset{(m,n)}{\arg\min} \; \frac{1}{I_{m,n}} \sum_{i=1}^{I_{m,n}} \left\| TSSV_{test} - TSSV^{i}_{(m,n)} \right\|_2 \\ &m \in [1,2\dots M],\; n \in [1,2\dots N],\; I_{m,n} \in \left[I_{1,1}, I_{1,2} \dots I_{M,N}\right] \end{aligned} \tag{8}$$

where $TSSV_{test}$ is the feature vector of the incoming ATC message to be localized, and $TSSV^{i}_{(m,n)}$ is the i_{th} TSS feature vector of the training data which locates in the (*m*, *n*) grid. $I_{(m,n)}$ is the number of training RSSI vectors which belong to the grid (*m*, *n*). Consequently, we can obtain the predicted grid label of $TSSV_{test}$. Finally, with a similar process in Sect. 3.1.3, we can use Ada-NN to derive the estimate of the aircraft position Loc_{TSS}, with its latitude Lat_{TSS} and longitude Lon_{TSS} (c.f. Table 1).

3.3 GSP spoofing detection

After obtaining Loc_{TSS} and Loc_{RSSI}, we next focus on detecting the spoofing attack based on the deviation between the estimated positions and the GPS-derived position contained in the ATC message ($Loc_{GPS} = [Lat_{GPS}, Lon_{GPS}]$). Since the attackers have the ability to manipulate the GPS-derived positions of aircrafts arbitrarily, the position offset modes can thus vary a lot, which largely affect the detection accuracy of directly using the deviation. To address this issue, the proposed spoofing detection module is based on XGBoost [28], a reliable technique which has the capability to learn various position deviation modes and even to resist the impact of localization errors.

3.3.1 Spoofing feature extraction

In order to achieve a high detection accuracy, we reveal the latent differences between the spoofed location and the estimated location of the target aircraft by constructing elaborated input classification features.

The feature vector *DetV* contains the following components (c.f. Table 1):

(1) The coordinates of Loc_{RSSI}, Loc_{TSS} and Loc_{GPS}.
(2) The latitude/longitude difference between Loc_{GPS} and Loc_{RSSI} (resp. Loc_{TSS}), which can be directly computed as follows:

$$Loc_{est}^{diff} = [Lat_{est} - Lat_{GPS}, Lon_{est} - Lon_{GPS}] \tag{9}$$

from which Loc_{RSSI}^{diff} and Loc_{TSS}^{diff} can be obtained.
(3) The geographical distance between Loc_{GPS} and Loc_{RSSI} (resp. Loc_{TSS}), which measures their distance along the surface of the earth, and can be computed following [33]:

$$D_{est}^{geo} = 2(R + h)\arcsin(\sqrt{C_{est}}) \tag{10}$$

$$with \quad C_{est} = \sin^2\left(\frac{Lat_{est} - Lat_{GPS}}{2}\right) + \cos\left(Lat_{est}\right)\cos\left(Lat_{GPS}\right)\sin^2\left(\frac{Lon_{est} - Lon_{GPS}}{2}\right) \tag{11}$$

where R is the radius of earth, and h is height of the 2D grid constructed by GPS-Probe. With Eqs. (10) and (11), D_{RSSI}^{geo} and D_{TSS}^{geo} can thus be obtained.

3.3.2 Spoofing detection based on XGBoost

The goal of the detection is to infer whether an aircraft is spoofed or not, so we can turn the spoofing detection into a binary classification problem, and build the spoofing detection module using XGBoost, with *DetV* as input feature vector.

To achieve a high detection accuracy, we have to adjust a set of parameters of XGBoost such as *max-depth*, *num-trees*, *learning-rate*.[5] We leverage the function *GridSearchCV* of scikit-learn machine learning library[6] to find the proper parameter values. We use standard accuracy metric on training and testing dataset for evaluating different parameters. When the detection accuracy of the model in the training set does not increase, or the accuracy in the testing set starts to decline,

[5] Please refer to Appendix for the principle of XGBoost.
[6] https://scikit-learn.org/stable/modules/grid_search.html.

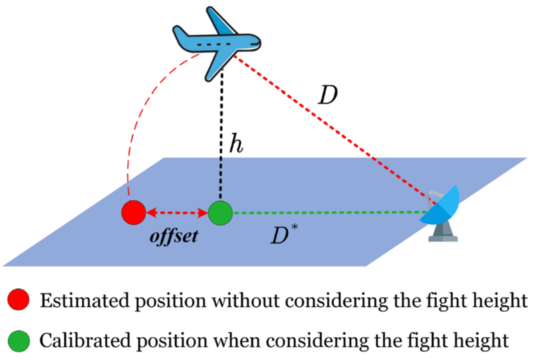

D
h
offset
D^*
Estimated position without considering the fight height
Calibrated position when considering the fight height

Fig. 7 A toy demo of the necessity to consider the flight height of the target aircraft, where *offset* = $D - D^*$

GridSearchCV stops searching and outputs the current parameters as the optimal selection.

After tuning the parameters, we obtain the binary model with two output classes, "Spoofed" and "Not". With the detection model, GPS-Probe can finally get the detection result of whether or not the target aircraft is spoofed.

4 GPS-Probe-Plus design

In the design of GPS-Probe, we assume that all flights cruise in a 2D plane with the same altitude. In practice, however, the target aircraft and its neighbors often fly at different altitudes for collision avoidance, which will introduce RSSI and TSS offsets and further affect the accuracy of the location estimation and spoofing detection. Let us take a toy demo to show the difference (c.f. Fig. 7). In GPS-Probe, we directly use the RSSI/TSS feature (corresponding to D) to obtain the estimated location of the target aircraft. However, as we actually compute the aircraft's location by averaging the geographic coordinates of the selected neighbors in the 2D plane, D^* projected in the 2D plane is therefore more precise than D for location estimation. When the flight height of the target aircraft is unknown, using D obviously yields coarse-grained location estimation. To address this issue, we propose GPS-Probe-Plus which can estimate the flight height of the target aircraft, so as to calibrate RSSI and TSS measurements by removing their offsets and finally improve the performance of location estimation and spoofing detection.

4.1 Flight height estimation

In order to estimate the flight height of the target aircraft, we first get the estimated position through GPS-Probe, and then project this position which we denote as Loc_P for clarity. Since the RSSI values of the ATC messages can represent the distance between target aircraft and the receiving ground sensor, and GPS-Probe can provide us with the estimated position of the target, we can simply calculate the flight height according to the Pythagorean theorem.

Though the basic idea of our height estimation sounds straightforward, it is non-trivial to realize this module of GPS-Probe-Plus due to several problems. First, as the most ground sensors of OpenSky Network are operated by voluntaries and amateurs, the RF hardware of GPS receivers varies a lot. The received RSSI values of different sensors differ a lot even the single source is the same aircraft at the same distance. What is more, the propagation of RF singles is effected by climate factors which can vary a lot in different positions. Therefore, the correspondence mode of received RSSI and distance D varies for different positions, and it is hard for us to model all the RSSI-distance modes. As a consequence, we cannot directly calculate the distance D from the RSSI value.

To address this issue, we leverage XGBoost regression to construct a machine learning model of height estimation for each ground sensor individually. We make use of RSSI value, the projection of estimated position Loc^P and the position of ground sensor Loc_{sen} to build our feature vector. We first transform RSSI to $RSSI_{linear}$ based on Eq. (4), and hence we can provide the height estimation model with a new feature which has the ability to represent D linearly. Then we project the estimated positions obtained from the RSSI and TSS based localization module, which we denote as Loc^P_{RSSI} and Loc^P_{RSSI} respectively. After that, we calculate the difference between Loc_{sen} and Loc^P_{RSSI} (resp. Loc^P_{TSS}), denoted as $Loc_{R_{diff}}$ (resp. $Loc_{T_{diff}}$). Therefore, this new feature Loc^P_{RSSI} and Loc^P_{TSS} includes latent information about the D^*. Finally we combine all the features together into a vector:

$$\begin{aligned} HgtV = [&RSSI, RSSI_{linear}, Loc^P_{RSSI}, Loc^P_{TSS},\\ &Loc_{sen}, Loc_{R_{diff}}, Loc_{T_{diff}}] \end{aligned} \tag{12}$$

where $HgtV$ is the height vector that we leverage to estimate the flight height h of a target aircraft.

4.2 RSSI and TSS calibration

By now we have constructed the height estimation model, we can easily get the target's flight height h. Next, we need to remove the offsets of RSSI and TSS measurements introduced by the flight height h. It is obvious that the relationship among h, D and D^* follows the Pythagorean theorem as:

$$D^* = \sqrt{D^2 - h^2} \tag{13}$$

Actually, RSSI and TSS measurements of the received ATC message correspond to the line-of-sight distance D from the target aircraft to the ground sensor, as shown in Fig. 7. When the target aircraft is projected to the 2D plane, the distance D^* matches the calibrated values of RSSI and TSS that are not affected by the flight height. Accordingly, the proportional relationship between D and D^* can be used to calibrate the RSSI and TSS measurements.

For calibrating RSSI, we first transform original value of RSSI to $RSSI_{linear}$ according to Eq. (4). Since $RSSI_{linear}$ is linearly dependent on the distance d, we can get the following relationship:

$$\frac{10\exp(-RSSI_{Plus}/20)}{10\exp(-RSSI/20)} \propto \frac{D^*}{D} \tag{14}$$

where $RSSI_{Plus}$ is the calibrated RSSI value.

For calibrating TSS, likewise, we leverage the linear relationship between the TSS and the distance D to remove the TSS offset introduced by the flight height:

$$\frac{TSS_{Plus}}{TSS} \propto \frac{D^*}{D} \tag{15}$$

where TSS_{Plus} is the calibrated TSS value.

4.3 Target relocation and spoofing detection

Now that we have the calibrated $RSSI_{Plus}$ and TSS_{Plus} of the ATC message that can represent the distance between the ground sensor and the projection location of the target aircraft in the 2D plane, then we can relocate the target aircraft and detect whether it is spoofed or not.

To relocate the target aircraft, we calibrate the RSSI and TSS values of all the received GPS messages, and construct the feature vector $RSSV$ and $TSSV$ with the calibrated values for each ATC message. Then we take the same localization procedure of GPS-Probe to obtain the estimated position of the target aircraft, denoted as Loc_{RSSI}^{Plus} and Loc_{TSS}^{Plus}. For GPS spoofing detection, we leverage Loc_{GSP}, Loc_{RSSI}^{Plus} and Loc_{TSS}^{Plus} to detect whether the target is spoofed or not by the same means as the detection process described in Sect. 3.

5 Performance evaluation

To evaluate the feasibility and effectiveness of GPS-Probe(-Plus), we build a simulation framework to generate impacts of spoofing attacks on GPS coordinates with real-world ATC data from OpenSky Network. We further compare GPS-Probe(-Plus) against Crowd-GPS-Sec [21] to show their superiority in terms of spoofing detection accuracy and detection delay.

5.1 Setup

The ATC data we used to evaluate GPS-Probe(-Plus) is from OpenSky Network, which contains 274,583 unique positions of 181 aircrafts over a period of 1 h. More specifically, this dataset has 94 millions of ADS-B messages broadcasted by the aircrafts, and 45% of these messages are used to report their aerial position. We divide the data set into the following three parts according to the proportions of 70%, 20% and 10%:

- Training set, which is used to build the RSSI/TSS localization fingerprints and the spoofing detection module.

- Development set, which is used to help us adjusting the module parameters to achieve a better detection accuracy.
- Testing set, which is leveraged to evaluate the performance of GPS-Probe(-Plus).

For evaluating the performance of our spoofing detection, we randomly select 10,000 position records from the testing set, and set the number of positive class equal to the number of negative class, in order to achieve the baseline accuracy which is expected to be 0.5.

5.1.1 Simulation framework

As the data crowdsourced by OpenSky Network are real GPS-derived position reports (without spoofed), we build a simulation framework to imitate the results of already spoofed coordinates of GPS-derived position. This allows us to analyze spoofing scenarios with a controlled spoofing deviation without having to spoof real aircrafts.

As we discussed before, even the same method of GPS spoofing attack may result in different localization deviations of both directions and distances. To imitate the spoofed positions, we add random noise to the latitude and the longitude of the selected ATC messages, the same way as in Crowd-GPS-Sec [21]. Subsequently, we put the spoofed data back while ensuring that the proportion of spoofed data is consistent in the training set, the development set and the testing set. Note that for each simulation, we performed 100 randomized runs to average out the randomness.

5.1.2 Grid design

Since the ATC data we used are generated by airliners and the civil aircrafts usually cruise at an altitude of 10,000 m except for take-off and landing phases, we construct a 2D grid over a typical flight altitude of 10,000 m with a size of 5.5 degrees longitude and 2.5 degrees of latitude, which covers an area of 610 km $\times$ 200 km = 122,000 km^2. Obviously, the computation time and localization accuracy also depend on the size of the monitoring area and the grid partition. 122,000 km^2 are typical for wide area ATC which includes a sufficient number of OpenSky ground sensors. The grids are evenly spaced in the monitoring area and the number of grids is a trade-off between computation time and performance. We separate the monitored aerial area into 1375 parts with an interval of 0.1 degree experientially. Then we leverage this 2D grid to construct the fingerprint map of GPS-Probe and localize the target aircraft. As for GPS-Probe-Plus, we construct another 2D grid with the same structure except for the grid altitude. Specifically, we build the new 2D grid on the earth surface with an altitude of 0 meters.

5.2 Performance of localization

We analyze the location accuracy by comparing our estimated position of the target aircraft to the authentic GPS-derived data and discuss the performance of

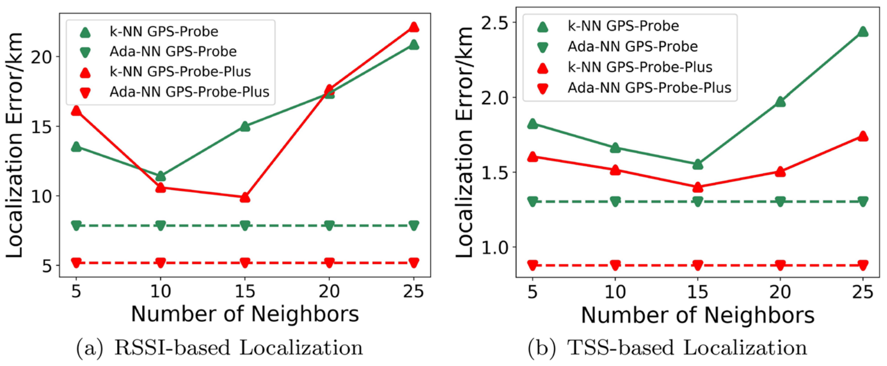


Fig. 8 Localization errors with original *k*-NN and Ada-NN

different localization approaches. We further compare Ada-NN against fixed number of neighbors, to show that our method could improve the localization accuracy efficiently.

5.2.1 Localization based on RSSI

We use a part of the whole flight dataset (comprising over 300,000 ATC messages), where each message has been received by at least 3 ground sensors, providing us with abundant RSSI measurements to estimate the position of the aircraft. All location messages are marked with pre-defined labels of grids, and are presented in terms of latitude and longitude. It is noticed that the nearest neighbors are selected from the dataset within one particular grid, thus grid classification has huge impacts on the localization accuracy. We make use of XGBoost to mark the labels of grids. To avoid over-fitting, we empirically set the number of decision trees to 150, and the tree depth to 8.

Subsequently, we analyze the accuracy of RSSI based localization with Ada-NN, and compare it with traditional *k*-NN based localization. The results show that Ada-NN performs 16.5% better than *k*-NN with 8 nearest neighbors on mean errors when we deploy Ada-NN to GPS-Probe. As for GPS-Probe-Plus, Ada-NN can assist our localization module reducing the mean localization error from 10.6 km obtain with original *k*-NN to 5.2 km. From Fig. 8a we find that for both GPS-Probe-Plus and GPS-Probe, Ada-NN works better than traditional *k*-NN on noisy RSSI measurements such as those we obtained from the real-world data.

5.2.2 Localization based on TSS

In line with the RSSI based localization, XGBoosts and TSS measurements in ATC messages are utilized to predicate the grid where the target aircraft locates in. In our experiments we find that the prediction accuracy increases dramatically with the number augment of decision trees, when the trees are less than 120. Further the accuracy gain goes to be flatten even the scale of decision trees are relatively

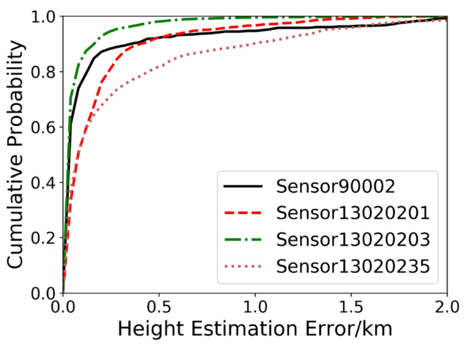


Fig. 9 Flight height estimation error of GPS-Probe-Plus

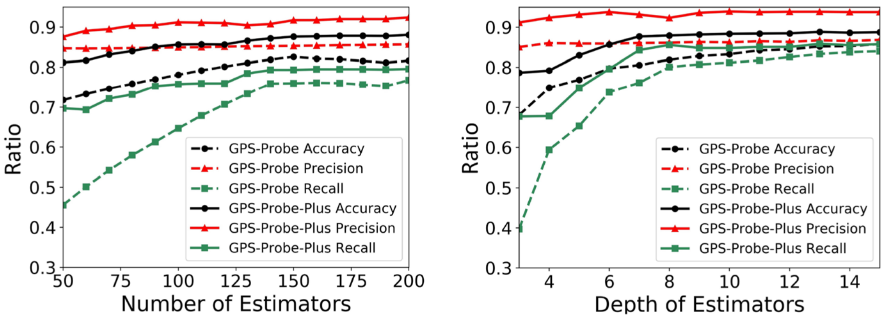
large. As for the selection of decision tree depth, we find that our XGBoost classifier achieves its highest accuracy with max depth of 6. To balance the accuracy and expense of module training, we empirically set the number of decision trees to 120 and tree depth to 6.

Then we compare the TSS localization accuracy between Ada-NN and traditional *k*-NN method. Experiments show that for the localization performance of GPS-Probe, Ada-NN does 11.7% better than *k*-NN with 8 nearest neighbors on mean errors. With regard to GPS-Probe-Plus, Ada-NN can help reduce the mean localization error from 1.4 km obtain with original *k*-NN to 0.9 km. From Fig. 8b we find that Ada-NN performs better than *k*-NN algorithm even the *k* changes. In general, regardless of the constant number of neighbors, an outlier could always induce a non-negligible localization bias to *k*-NN based localization algorithms. Ada-NN can filter out the outliers when selecting neighbors, thereby achieving a high accuracy with the real-world flight data.

5.3 Flight height estimation

Since GPS-Probe-Plus requires estimating the flight height of the target aircraft to calibrate the RSSI and TSS values of an incoming ATC message, the accuracy of the height estimation has a great impact on the localization and spoofing detection performance. We need to evaluate the height estimation accuracy of GPS-Probe-Plus to show that our estimation module performs well with the realistic ATC data.

In the height estimation module of GPS-Probe-Plus, we train a regression model for each ground sensor in OpenSky Network separately with XGBoost's regression package. Subsequently, we evaluate the performance of the height estimation module and the experiment results of a part of sensors are shown in Fig. 9. From the experiment results we can see that our estimation module can achieve a mean error of 0.168*km* over all ground sensors. The sensor 13020203 has the best performance of height estimation over all sensors, and achieves a mean error of 0.06 km. The sensor 13020235 performs worst and achieve an average error of 0.298 km. While the average estimation errors of the rest sensors are around 0.18 km. One point should be noted that the error of height estimation will introduce an offset to the calibration of TSS and RSSI values. The offset is relatively small and has a slight impact on the



(a) Impacts of estimators (b) Impacts of depths

Fig. 10 Detection performance of different parameters

performance of localization and spoofing detection of GPS-Probe-Plus. Therefore, the error range of our height estimation module is acceptable.

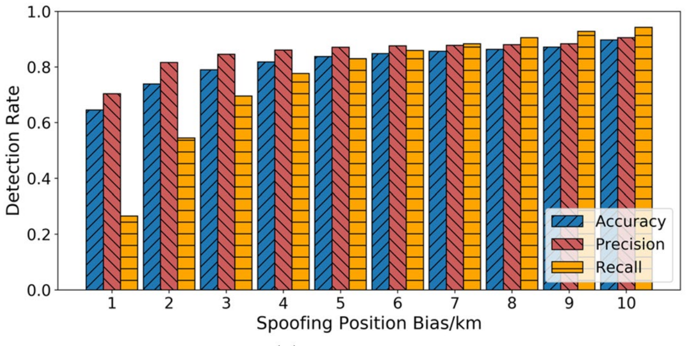
5.4 Performance of spoofing detection

To show how well GPS-Probe(-Plus) perform with the real-world flight data, we focus on evaluating the of spoofing detection accuracies, as well as the detection delays.

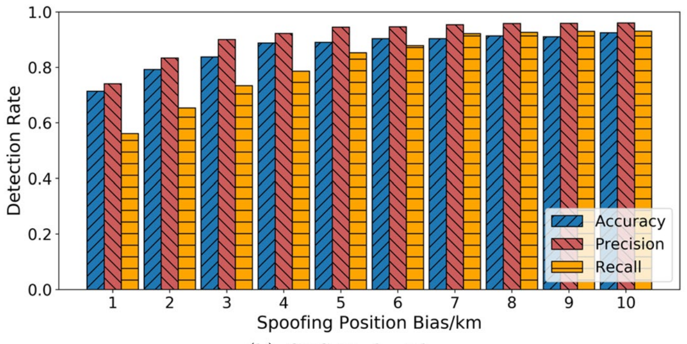
5.4.1 Detection accuracy

In this part, we evaluate the performance of our spoofing detection using the standard accuracy, precision, and recall metrics in the machine learning filed. The accuracy is the fraction of the correctly detected records for all testing records. The precision reflects the proportion of the testing records predicted as being spoofed which are indeed attacked by the GPS spoofers. The recall means the fraction of the spoofed records which we can correctly detect. All these metrics are in the form of proportions, and indicate that our detection performs well if they are close to 1. We firstly assess the impact of different parameters of XGBoost which we use to detect the GPS spoofing attacks. However, there are a lot of XGBoost parameters which have effects in classifier performance and the module adjustment is not the main part of our paper. We only discuss two significant parameters: the number and the depth of the decision trees, denoted as N_{tree} and D_{tree} respectively.

Concerning the optimal choice of N_{tree}, Fig. 10a demonstrates the accuracy gained with the increasing of decision trees of GPS-Probe(-Plus). For GPS-Probe, we can see a large accuracy improvement until $N_{tree} = 160$. Further decreases in mean accuracy are small and much less pronounced with more decision trees. For GPS-Probe-Plus, the detection accuracies barely increase when N_{tree} is greater than 150, therefore $N_{tree} = 150$ is a balanced choice between model's performance and over-fitting avoidance. In order to select the optimal choice of the depth of decision trees, we leverage the experiment results obtained with different tree



(a) GPS-Probe



(b) GPS-Probe-Plus

Fig. 11 Detection accuracy of different spoofing attack ranges

depth shown in Fig. 10b. GPS-Probe(-Plus) can obvious accuracy improvements until $D_{tree} = 9$ and 10 respectively. These experiments show that we can achieve a high detection accuracy with a set of proper parameters of XGBoost.

Subsequently, we evaluate the performance of GPS-Probe with different position deviations caused by GPS spoofing attacks. In order to simulate the spoofed GPS positions, we use the simulation framework described previously to add different localization deviations (from 1 to 10 km) on a set of randomly selected flight data, and analyze the detection performance of GPS-Probe under these conditions.

The experiment results of GPS-Probe show that the detection accuracy and precision is up to 64.5% and 7.04% respectively when the spoofed GPS position has a distance of 1 km to the authentic position. Figure 11a shows that the increasing of spoofed position deviation makes it easier to detect attacks in GPS-Probe. The detection accuracy, precision and recall all can reach around 90% with a mean GPS-derived position spoofed bias of 10 km. As for GPS-Probe-Plus, it can achieve the mean detection accuracy and precision of 71.3% and 74.1% respectively on the condition of 1 km spoofing position bias. As shown in Fig. 11b, the accuracy, precision and recall of GPS-Probe-Plus increase with the

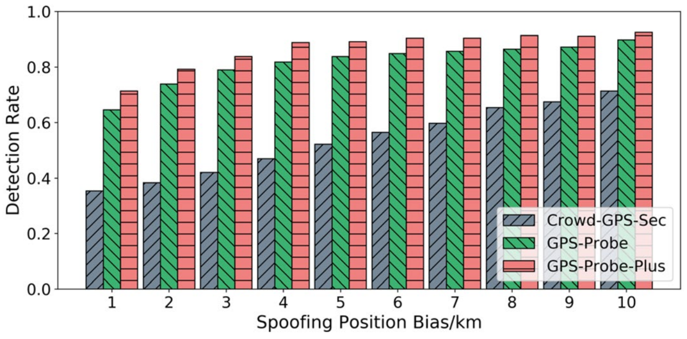
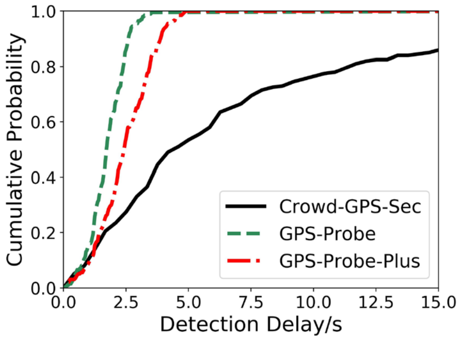


Fig. 12 Comparison of the detection accuracy

Fig. 13 Comparison of the detection delay



spoofing position bias increasing, and all can exceed 90% when the spoofing bias is greater than 6 km. The detection accuracy and precision GPS-Probe-Plus can reach to 92.5% and 96.1% respectively with a mean bias of 10 km.

From the results shown in Fig. 11, we can see that GPS-Probe-Plus performs better than GPS-Probe, especially when the spoofing position bias is not large. The main reason is that the localization module of GPS-Probe-Plus could achieve a more precise localization performance which has huge impact on the accuracy of spoofing detection.

5.4.2 Detection delay

We define the detection delay as the time cost between the time when the ATC message is received by the ground sensor until GPS-Probe detects the spoofing attack. This delay corresponds to the time induced by the computation of localization and spoofing detection. Since the ATC message is usually received by several different ground sensors, we denote the arriving time at the first receiving ground sensor as the reception time. Once our localization and detection models are constructed, GPS-Probe can achieve an average detection delay close to 1.7 s as show in Fig. 13. In other words, GPS-Probe can detect an attack as soon as a spoofed position report is received by ground sensors. As for GPS-Probe-Plus, since GPS-Probe-Plus needs to estimate the flight height and then re-locate the position of the target aircraft, GPS-Probe-Plus attains a mean detection delay of 2.3 s which is slight slower than GPS-Probe.

5.5 Comparison results

We compare the detection accuracies and delays of among GPS-Probe, GPS-Probe-Plus and Crowd-GPS-Sec in this part. Since Crowd-GPS-sec can not detect spoofing attacks with less than 3 ground sensors, we select the ATC data which are monitored by at least four ground sensors to conduct our comparisons. As for the parameters and settings of our comparison, GPS-Probe(-Plus) remains consistent with the above experiments, while Crowd-GPS-Sec leverages the OpenSky's default settings as described in [21].

Figure 12 shows the the mean detection accuracy of the three algorithms corresponding to different spoofing position biases. Our evaluation shows that GPS-Probe-Plus apparently outperforms Crowd-GPS-Sec, and GPS-Probe has a resemble detection performance with GPS-Probe-Plus. GPS-Probe-Plus achieves a mean detection rate of 86.8%, and GPS-Probe achieves a mean detection rate of 81.7% which is better than Crowd-GPS-Sec by about 28.2%. More specifically, GPS-Probe-Plus can achieve a detection accuracy of 71.3% with only 1 km spoofing position bias, and the GPS-Probe can achieve 64.5%. However, Crowd-GPS-Sec only can achieve a rate of 35.4% on the same condition. The detection rates of all methods increases with the spoofed biases, and GPS-Probe-Plus attains an accuracy higher than 90% in the first place when the spoofing bias increases. Even with $10km$ attacked position bias, the detection rate of GPS-Probe-Plus outperforms GPS-Probe and Crowd-GPS-Sec by 2.8% and 18.3% respectively. The results are not surprising since our detection model based on machine learning has the ability to "learn" different attack modes, while Crowd-GPS-Sec only can detect spoofing attacks with a constant threshold. Besides, the localization module of GPS-Probe-Plus performs better than that of GPS-Probe, and thus GPS-Probe-Plus could detect the spoofing attacks more precisely than our original method.

From Fig. 13, we can observe that the detection delay of Crowd-GPS-Sec is much longer than GPS-Probe(-Plus). The average delay of Crowd-GPS-Sec is 5.3 s due to long MLAT calculation time. In contrast, GPS-Probe can get the estimated position within 1 s once our localization model is constructed. Then GPS-Probe and GPS-Probe-Plus can achieve a mean spoofing detection delay of 1.7 s and 2.3 s, respectively, resulting from both localizing the target aircraft and detecting spoofing attacks.

6 Related work

Detection of GPS spoofing attacks has received considerable attentions both in the industry and academia. Existing detection methods can be coarsely categorized into three class. The first class leverages the characteristics of RF signals, such as signal strength, peaks, and phase. Akos [20] suggests to monitor the incoming GPS signal strength and the state of the automatic gain control. Another technique called SPREE relies on auxiliary peak tracking [34] to detect suspicious peaks from signals with weaker acquisition correlation peaks. Psiaki et al. [35] propose a detection scheme which uses an additional reference receiver to correlate the received signal with authentic signals assuming the inclusion of the encrypted military signal. A spoofed signal does not correlate with the reference node's received signal and the attack can be detected.

The second class of detection approaches relies on multiple receiving antennas or multiple receivers. Montgomery et al. [17] use a dual antenna receiver setup to determine the angle of arrival of incoming signals, and Psiaki et al. [36] extended this approach to include differential carrier phase measurements. Bhamidipati et al. [37] propose to leverage geographically distributed multiple directional antennas for GPS spoofing detection and develop a Belief-Propagation-based extended Kalman filter to estimate the timing errors. Multiple co-located GPS receivers are used in [24, 38–40] to detect attacks by comparing the GPS coordinates and times of these receivers. The use of receiving antenna arrays shows that signal diversity of different antennas is an effective indicator to detect spoofing attacks without knowledge of the target's location [41]. Although these detection approaches do not require upgrades of the existing GPS infrastructure, they need more sophisticated GPS receivers which would significantly increase the costs, complexity and power requirements.

Since the massive upgrades of GPS infrastructure are unlikely to happen in the near future, Jansen et al. [21] proposed Crowd-GPS-Sec, a novel method to detect spoofing attacks. This approaches leverages the ATC messages of aircrafts that aircraft periodically broadcast for ATC purposes. Crowd-GPS-Sec locates a target aircraft by an independent infrastructure on the ground which analyzes the contents and the times of arrival of these GPS broadcasts. Then it compares the location results with the GPS-derived position to identify whether the target is attacked. However, the performance of Crowd-GPS-Sec depends heavily on location accuracy of GPS, even with a low GPS error, the detection rate just reaches 75%.

7 Conclusion

In this paper we first present GPS-Probe, a synchronization-free GPS spoofing detection scheme with crowdsourced ATC data. GPS-Probe is the first work to take advantage of RSSI and TSS features and machine learning techniques for GPS spoofing detection with crowdsourced ATC data. It neither requires any updates of the GPS infrastructure nor of the GPS receivers. More importantly, it also releases the requirement on the time synchronization of the ground sensors distributed around the world. Then we improve our original work and propose GPS-Probe-Plus which performs much better on both target localization and spoofing detection. Evaluation results with real-world ATC data from OpenSky Network validate our spoofing detection methods effectiveness and efficiency. In the future, we plan to integrate the RSSI and TSS data to improve the accuracy of our localization and the performance of our spoofing detection.

Funding This work was supported in part by the National Key Technologies R&D Program of China under Grant No. 2016YFC0201101; the National Natural Science Foundation of China under Grant Nos. 61872416, 61702204, 61671216, and 51479159; by the Fundamental Research Funds for the Central Universities of China under Grant 2019kfyXJJS017; by the Natural Science Foundation of Hubei Province of China under Grant 2019CFB191; by the Open Research Project of Hubei Key Laboratory of Intelligent Geo-Information Processing under Grant No. KLIGIP-2018A03; and by the fund of Hubei Key Laboratory of Transportation Internet of Things under Grant 2018IOT004. Ling Liu's research is partially support by the National Science Foundation under NSF Grant Nos. 1547102, 1564097 and an IBM faculty award. An earlier version of this work appeared in *Proceedings of 20th IEEE MDM* [1].

Appendix: Extreme gradient boosting

Tree boosting is a highly effective and widely used machine learning technique. Due to the poor classification performance of single decision tree, the method of random forests [42] is proposed to achieve better prediction precision by assembling multiple decision trees. Random forest trees are built independent of each other, and therefore the trained models are often unstable, and do not perform well on "small" data.

Extreme Gradient Boosting (XGBoost) algorithm was proposed in 2016 [28] and can address the aforementioned issues of random forests. XGBoost builds a new tree according to the already built ones, and the new tree focuses on how to correctly classify the misclassified data samples.

The training process of XGBoost is as follows. For a given dataset: $\{(x_i, y_i) : i = 1 \dots n, x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$. The result given by an ensemble represented by the generated model is:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{16}$$

where f_k is a single decision tree and $f_k(x_i)$ represents the score given by the kth tree to the ith observation in data. The goal of XGBoost is to minimize the following regularized objective function in order to choose the structure of decision tree f_k:

$$\mathcal{L} = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{17}$$

where l is the loss function and Ω is the regularization. Specifically, the penalty term Ω is shown as follows:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{18}$$

where γ and λ are parameters controlling the number of leaf nodes and magnitude of leaf weights, respectively.

XGBoost leverages an iterative method to minimize the objective function (17). In jth iteration, XGBoost adds a new tree f_j and minimizes the modified objective function as:

$$\mathcal{L}^j = \sum_i l((y_i, \hat{y}_i^{j-1}) + f_j(x_i)) + \sum_k \Omega(f_k). \tag{19}$$

Then adopting Taylor expansion, XGBoost can simplify this function and derive the loss function after the tree split from given node. By comparing the loss of tree's nodes, XGBoost can find the best split at a given node. It is easy to see that XGBoost can construct a series of trees by gradually iteration and every new tree correctly predicts the misclassified data obtained from the already built ones.

When using XGBoost in practice, we need to adjust the parameters of the XGBoost model to control the model structure and achieve better performance. There are two important parameters which manipulate the model structure locally and globally. One is "max_depth" which controls the depth of the decision tree f_j, and the other is "n_estimators" which controls the number of decision trees, i.e. the upper limit of j in Eq. (19). Increasing the value of "max_depth" will make the XGBoost model more complex and more likely to overfit, and the default max_depth is set to 6. XGBoost adds a new tree to the existing model in each iteration, and thus "n_estimators" also control the maximum number of iterations. Since every added tree is trained on the misclassified data by previous trees, increasing the value of "n_estimators" will also make the model overfitting on some outlier data and losing model's generalization, while decreasing "n_estimators" will degrade the accuracies of XGBoost models. What is more, increasing "max_depth" or "n_estimators" both will result in the requirements of longer training time and more calculation resources. So the key point is to choose an appropriate set of model parameters.

Besides those improvements in terms of the algorithm, XGBoost also performs better than other tree boosting methods. It supports an approximate split finding, which improves the process of the building trees and scales very well with the number of CPU cores (detailed in its github page.[7])

References

1. Liu, G., Zhang, R., Wang, C., Liu, L.: Synchronization-free GPS spoofing detection with crowdsourced air traffic control data. In: Proceedings of 20th IEEE MDM, pp. 260–268 (2019)
2. Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J.: Global Positioning System: Theory and Practice. Springer, New York (2012)
3. Liu, Z., Shi, X., He, L., Yu, D., Jin, H., Yu, C., Dai, H., Feng, Z.: A parameter-level parallel optimization algorithm for large-scale spatio-temporal data mining. In: Agrawal, D., Mokbel, M. (eds.) Distributed and Parallel Databases, pp. 1–27. Springer, Cham (2020). https://doi.org/10.1007/s10619-020-07287-x
4. Wang, C., Lin, H., Jiang, H.: CANS: towards congestion-adaptive and small stretch emergency navigation with wireless sensor networks. IEEE Trans. Mob. Comput. **15**(5), 1077–1089 (2016)
5. Zhang, W., Li, M., Tandon, R., Li, H.: Online location trace privacy: an information theoretic approach. IEEE Trans. Inf. Forensics Secur. **14**(1), 235–250 (2019)

[7] https://github.com/dmlc/XGBoost.

6. Tang, J., Chen, G., Coon, J.P.: Secrecy performance analysis of wireless communications in the presence of UAV jammer and randomly located UAV eavesdroppers. IEEE Trans. Inf. Forensics Secur. **14**(11), 3026–3041 (2019)
7. Psiaki, M.L., Humphreys, T.E.: GNSS spoofing and detection. Proc. IEEE **104**(6), 1258–1270 (2016)
8. Moser, D., Leu, P., Lenders, V., Ranganathan, A., Ricciato, F., Capkun, S.: Investigation of multi-device location spoofing attacks on air traffic control and possible countermeasures. In: Proceedings of ACM MobiCom, pp. 375–386 (2016)
9. Schmidt, E., Ruble, Z., Akopian, D., Pack, D.J.: Software-defined radio GNSS instrumentation for spoofing mitigation: a review and a case study. IEEE Trans. Instrum. Meas. **68**(8), 2768–2784 (2019)
10. Humphreys, T.E., Ledvina, B.M., Psiaki, M.L., O'Hanlon, B.W., Kintner, P.M.: Assessing the spoofing threat: development of a portable GPS civilian spoofer. In: Proceedings on Radionavigation Laboratory Conference (2008)
11. Kerns, A.J., Shepard, D.P., Bhatti, J.A., Humphreys, T.E.: Unmanned aircraft capture and control via GPS spoofing. J. Field Robot. **31**(4), 617–636 (2014)
12. Psiaki, M.L., Humphreys, T.E., Stauffer, B.: Attackers can spoof navigation signals without our knowledge. Here's how to fight back GPS lies. IEEE Spectr. **53**(8), 26–53 (2016)
13. Zhao, P., Li, J., Zeng, F., Xiao, F., Wang, C., Jiang, H.: ILLIA: enabling k-anonymity-based privacy preserving against location injection attacks in continuous LBS queries. IEEE Internet Things J. **5**(2), 1033–1042 (2018)
14. Wesson, K.D., Gross, J.N., Humphreys, T.E., Evans, B.L.: GNSS signal authentication via power and distortion monitoring. IEEE Trans. Aerosp. Electron. Syst. **54**(2), 739–754 (2018)
15. Heng, L., Work, D.B., Gao, G.X.: GPS signal authentication from cooperative peers. IEEE Trans. Intell. Transp. Syst. **16**(4), 1794–1805 (2015)
16. Wesson, K., Rothlisberger, M., Humphreys, T.: Practical cryptographic civil GPS signal authentication. Navigation **59**(3), 177–193 (2012)
17. Montgomery, P.Y.: Receiver-autonomous spoofing detection: experimental results of a multi-antenna receiver defense against a portable civil GPS spoofer. In: Proceedings of Radionavigation Laboratory Conference (2011)
18. Nielsen, J., Broumandan, A., Lachapelle, G.: GNSS spoofing detection for single antenna handheld receivers. Navigation **58**(4), 335–344 (2011)
19. Psiaki, M.L., Powell, S.P., O'hanlon, B.W.: GNSS spoofing detection using high-frequency antenna motion and carrier-phase data. In: Proceedings of the ION GNSS Meeting, pp. 2949–2991 (2013)
20. Akos, D.M.: Who's afraid of the spoofer? GPS/GNSS spoofing detection via automatic gain control (AGC). Navigation **59**(4), 281–290 (2012)
21. Jansen, K., Schäfer, M., Moser, D., Lenders, V., Pöpper, C., Schmitt, J.: Crowd-GPS-Sec: leveraging crowdsourcing to detect and localize GPS spoofing attacks. In: Proceedings of IEEE S&P, pp. 1018–1031 (2018)
22. Xu, B., Sun, G., Yu, R., Yang, Z.: High-accuracy TDOA-based localization without time synchronization. IEEE Trans. Parallel Distrib. Syst. **24**(8), 1567–1576 (2013)
23. Schäfer, M., Strohmeier, M., Lenders, V., Martinovic, I., Wilhelm, M.: Bringing up OpenSky: a large-scale ADS-B sensor network for research. In: Proceedings of ACM/IEEE IPSN, pp. 83–94 (2014)
24. Tippenhauer, N.O., Pöpper, C., Rasmussen, K.B., Čapkun, S.: On the requirements for successful GPS spoofing attacks. In: Proceedings of ACM CCS, pp. 75–85 (2011)
25. Nolan, M.: Fundamentals of Air Traffic Control. Cengage Learning, Boston (2010)
26. Trüb, R., Moser, D., Schäfer, M., Pinheiro, R., Lenders, V.: Monitoring meteorological parameters with crowdsourced air traffic control data. In: Proceedings of ACM/IEEE IPSN, pp. 25–36 (2018)
27. Zheng, Y., Liu, Y., Zhou, Z.: From RSSI to CSI: indoor localization via channel response. ACM Comput. Surv. **46**(2), 1–32 (2013)
28. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of ACM SIGKDD, pp. 785–794 (2016)
29. Liu, X., Cao, J., Tang, S., Wen, J., Guo, P.: Contactless respiration monitoring via off-the-shelf WiFi devices. IEEE Trans. Mob. Comput. **15**(10), 2466–2479 (2016)
30. Strohmeier, M., Lenders, V., Martinovic, I.: A localization approach for crowdsourced air traffic communication networks. arXiv preprint arXiv:1610.06754 (2016)

31. Wang, C., Liu, G., Huang, H., Feng, W., Peng, K., Wang, L.: MIASec: enabling data indistinguishability against membership inference attacks in MLaaS. IEEE Trans. Sustain. Comput. **1**, 1–12 (2020). https://doi.org/10.1109/TSUSC.2019.2930526
32. Hernández, J.A., Phillips, I.W.: Weibull mixture model to characterise end-to-end internet delay at coarse time-scales. IEE Proc. Commun. **153**(2), 295–304 (2006)
33. Van Brummelen, G.: Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry. Princeton University Press, Princeton (2012)
34. Ranganathan, A., Ólafsdóttir, H., Capkun, S.: Spree: a spoofing resistant GPS receiver. In: Proceedings of ACM MobiCom, pp. 348–360 (2016)
35. Psiaki, M.L., O'Hanlon, B.W., Bhatti, J.A., Shepard, D.P., Humphreys, T.E.: GPS spoofing detection via dual-receiver correlation of military signals. IEEE Trans. Aerosp. Electron. Syst. **49**(4), 2250–2267 (2013)
36. Psiaki, M.L., O'hanlon, B.W., Powell, S.P., Bhatti, J.A., Wesson, K.D., Humphreys, T.E.: GNSS spoofing detection using two-antenna differential carrier phase. In: Proceedings of Radionavigation Laboratory Conference (2014)
37. Bhamidipati, S., Kim, K.J., Sun, H., Orlik, P.V.: GPS spoofing detection and mitigation in pmus using distributed multiple directional antennas. In: Proceedings of IEEE ICC, pp. 1–7 (2019)
38. Jansen, K., Tippenhauer, N.O., Pöpper, C.: Multi-receiver GPS spoofing detection: error models and realization. In: Proceedings of ACM ACSAC, pp. 237–250 (2016)
39. Wang, Q., Lu, Z., Gao, M., Qu, G.: Edge computing based gps spoofing detection methods. In: Proceedings of IEEE DSP, pp. 1–5 (2018)
40. Jiang, C., Chen, S., Chen, Y., Bo, Y., Xia, Q., Zhang, B.: Analysis of the baseline data based GPS spoofing detection algorithm. In: Proceedings of IEEE/ION PLANS, pp. 397–403 (2018)
41. Magiera, J., Katulski, R.: Detection and mitigation of GPS spoofing based on antenna array processing. J. Appl. Res. Technol. **13**(1), 45–57 (2015)
42. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

Affiliations

Gaoyang Liu[1] · Rui Zhang[2] · Yang Yang[3] · Chen Wang[1] · Ling Liu[4]

Gaoyang Liu
liugaoyang@hust.edu.cn

Rui Zhang
zhangrui@whut.edu.cn

Yang Yang
yangyang@hubu.edu.cn

Ling Liu
lingliu@cc.gatech.edu

[1] School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China

[2] Hubei Key Laboratory of Transportation Internet of Things, School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, People's Republic of China

[3] School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, People's Republic of China

[4] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0765, USA