Context-Based Detection of GNSS Position Spoofing for Smartphones

Francesco Formaggio, Silvia Ceccato, Nicola Laurenti, and Stefano Tomasin

In this chapter, following the trace of [1], we delve into three solutions to let smartphones detect GNSS spoofing attacks. The first, and most simple, foresees the analysis of the visible satellites and the corresponding navigation message, checking both its integrity and its correctness. The second approach, first introduced in [2], makes use of the network connectivity of the smartphone, which provides alternative location information, in order to verify the consistency of the global navigation satellite system (GNSS) measurements. Finally, with the third approach we process the smartphone inertial measurements unit (IMU) data with a Kalman filter (KF) to derive a suitable spoofing detection mechanism.

1 Literature Background

The *spoofing* attack is a well known threat, where a malicious entity forges fake GNSS signals in order to trick a victim receiver into computing the desired false position and/or time. An extensive literature has been produced on spoofing detection for various use cases [3–17].

Spoofing detection in vehicular applications is investigated in [6], where a mobile device is used for comparing the absolute value of linear and angular acceleration with those obtained from GNSS. This approach avoids the calibration of inertial measurements units (IMU) and is invariant to manipulations of the device initial orientation. The automotive scenario is also the target application of [7], wherein the

F. Formaggio · S. Ceccato · N. Laurenti · S. Tomasin (✉)
Department of Information Engineering, University of Padova, Padova, Italy
e-mail: francesco.formaggio@dei.unipd.it; silvia.ceccato@dei.unipd.it; nicola.laurenti@dei.unipd.it; stefano.tomasin@dei.unipd.it

S. Paiva (ed.), *Precision Positioning with Commercial Smartphones in Urban Environments*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-71288-4_1

proposed solution integrates data from GNSS, IMU, and the odometer. Differently from [10] and [6], the comparison metric is position rather than acceleration, and the detection statistics are obtained as the norm of the difference between position vectors (from GNSS and from IMU or odometer). The novelty in this approach is the idea of performing GNSS-based sensor calibration at fixed time intervals only when the GNSS signal is considered authentic.

Note that it is generally believed to be impractical for an attacker to spoof a vehicle position without the end user detecting the inconsistency with the surrounding environment. However, in [11] the adversary aims at luring a victim receiver to a specific location, while maintaining the consistency between outside environment and Google Maps' trajectory. The spoofing optimization algorithm searches for areas of the map that are topologically similar to the road shape in the user real location. The spoofing signal then induces a jump to the user location, eventually driving it to the selected fake position through a path that mimics on the map the user actual trajectory, making it hard for the receiver to notice the attack.

In aviation, the authors of [10] identify high frequency acceleration components as a suitable source of randomness for authentication purposes, similarly to [8]. The work targets an attacker with imperfect information on the precise aircraft acceleration and develops a spoofing detection algorithm based on decoupling IMU and GNSS positioning, providing a direct comparison of the acceleration for an unlimited time window.

Other anti-spoofing techniques include cryptographic mechanisms applied to the navigation message [12, 13], spreading code encryption [14], signal quality monitoring techniques [15, 16], and physical-layer authentication schemes [17, 18].

Only in recent years the interest in anti-spoofing techniques has been extended to mobile devices applications. As location-based services (LBS) are now deeply integrated in billions of people everyday life, the security of positioning in mobile phones has become a concern. The most popular application of LBS is navigation: traffic monitoring, vehicle management, and road information are just some of the services that exploit positioning information in smartphones, offering guidance in unfamiliar environments and improving the overall traveling experience. LBS have also spurred the development of taxi sharing platforms, (e.g., Uber and Lyft) that have become a competitive alternative to other services, thanks to features such as real time monitoring for both providers and customers. Several other fields now benefit from LBS, such as emergency and disaster management, insurance and financial applications, and production process support. Home banking, financial transactions, mobile based transportation, goods delivery, and access control based on location proximity are just some examples of services readily accessible from our mobile device that could be compromised by spoofing attacks. Indeed, in [1] evidence is provided that even modern smartphones are vulnerable to such security threats. Table 1 (from [1]) shows the time needed to obtain a fix for various smartphones models, under three spoofing attacks:

Exp. 1 Spoofing from the correct position (Padova, Italy) to the fake position (New York, USA).

Table 1 Navigation data spoofing: Experiment results, from [1]

	Time to fix			
Smartphone	Exp. 1	Exp. 2	Exp. 3	Exp. 4
Apple iPhone 5	< 30 s	< 60 s	No fix	No fix
Apple iPhone 6s	< 120 s	< 30 s	No fix	No fix
Apple iPhone SE	< 60 s	< 60 s	No fix	No fix
Asus Nexus 7	< 120 s	< 30 s	No fix	No fix
Asus Zefone 2	< 30 s	< 180 s	< 30 s	No fix
Google Pixel	< 30 s	< 60 s	< 30 s	No fix
HTC one M9	< 60 s	No fix	No fix	No fix
Huawei Honor 8	< 30 s	< 30 s	< 30 s	< 30 s
Huawei Honor 9	< 30 s	< 30 s	< 120 s	< 120 s
Huawei p8 lite	< 60 s	< 180 s	No fix	No fix
Huawei p10 lite	< 30 s	< 120 s	No fix	No fix
LG Nexus 5	< 30 s	< 30 s	No fix	No fix
LG Nexus 5x	< 30 s	< 60 s	No fix	No fix
LG G6	< 30 s	< 60 s	< 60 s	No fix
LG G3	< 120 s	< 120 s	No fix	No fix
Motorola Moto G	< 30 s	< 30 s	No fix	No fix
OnePlus 2	< 60 s	< 120 s	No fix	No fix
OnePlus 5	< 60 s	< 120 s	No fix	No fix
Samsung S6 Edge	< 30 s	< 30 s	< 30 s	< 30 s
Samsung S6	< 30 s	< 60 s	< 60 s	< 240 s
Samsung S7 Edge	< 60 s	< 120 s	< 60 s	< 30 s
Xiaomi Mi 4c	< 60 s	< 120 s	No fix	No fix
Xiaomi Mi5	< 30 s	< 30 s	No fix	No fix
Xiaomi Mi6	< 120 s	No fix	No fix	No fix
Xiaomi Redmi Note 4x	< 60 s	< 120 s	< 30 s	No fix

Exp. 2 Spoofing from the Empire State Building to Canberra, Australia. In the spoofed signal navigation data were erased, except for the telemetry word, the handover word, and the time indicators. Old navigation data was used by the smartphone.

Exp. 3 Spoofing from the Empire State Building to the New York airport. In the spoofed signal all navigation data erased. Old navigation data was used by the smartphone.

Exp. 4 Spoofing from the Empire State Building to Canberra, Australia. In the spoofed signal all navigation data erased and all stored data in smartphone was erased.

See [2] for more details on the experiments.

2 Visible Satellites and Navigation Message

In this Section we present a client-server (CS) architecture designed to detect specific spoofing attacks. This CS architecture was also implemented in the form of an Android application (APP). In the context of GNSS spoofing, the client, or user, is also the victim receiver, subject to the attacker's GNSS signals. The server instead is an external application, which we assume authentic and capable of reliable communication with the user.

2.1 Attack Model

We consider spoofing attacks, wherein an attacker generates false GNSS signals and sends them to the victim receiver with the intention to induce a position estimation that does not correspond to the true user's position.

The attacker chooses a fake position and the set of satellites, visible from the fake position, that he intends to spoof. We then distinguish two attacks.

1. The attacker does not *null* the legitimate signals received by the user, while instead he transmits a higher power spoofing signal. Note that nulling is theoretically feasible, since for every time and every position it is possible to predict the exact shape of any legitimate GNSS signal; therefore, the attacker could send these signals with inverse polarity and cancel them at the victim receiver. However, nulling requires the exact computation of the carrier phase at each time instant, which is a demanding task, corroborating the significance of this first attack model.
2. In this case, the attacker tampers also with the navigation message, modifying its content or even completely deleting all information. This might not seem a clever attack, since predicting the navigation message is not as difficult as the carrier phase estimation problem for nulling. However, in [1] it is shown that nowadays smartphones obtain positioning information even from GNSS signals with incorrect or missing navigation message. Moreover, the attacker might want to simply disrupt or destructively interfere with the GNSS services, making the navigation message manipulation its very objective.

2.2 Client-Server Architecture

The proposed CS architecture is shown in Fig. 1, where, as previously mentioned, the client represents the victim receiver, subject to the two attacks described in Sect. 2.1.

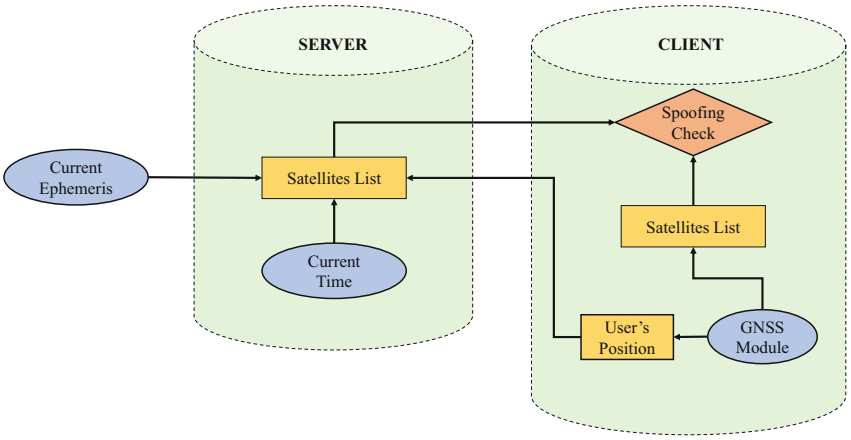


Fig. 1 CS architecture of the proposed spoofing detection strategy. The client queries the server sending the computed position, and the server answers with a list of visible satellites, for which the spoofing check is performed

The spoofing detection procedure starts with the GNSS module at the client side, which provides an estimation of the user's position. The client then queries the server by sending the estimated position itself.

The server keeps an updated version of ephemeris data, which are typically stored in institutional websites, such as [19] for GPS. Indeed, in Fig. 1 the current ephemeris block is placed outside the server, because it represents already existing information and the server needs only to retrieve it. With current ephemeris, current time (also available at the server), and user's position, the server builds a list of satellites that are visible from the client's position and sends it back to the client itself. The list contains the satellites' identifiers (IDs), which uniquely identify each satellite by its spreading code number, and the satellite's positions (azimuth and elevation). Note that the server does not use any GNSS module to build the list, therefore it is not subject to GNSS spoofing and the information sent to the user is assumed authentic.

At the client's side, another satellite list is produced, equivalent to the one at the server, but this time the IDs and positions of satellites are computed directly by the GNSS module, thus being prone to forgery. Indeed, the spoofing check consists in comparing the satellite list produced by the client and the one coming from the server, as shown in Fig. 1.

2.3 Spoofing Check

The spoofing detection at the client includes two checks against the two attacks of Sect. 2.1.

The first check needs actually only the satellites' IDs retrieved by the client through the GNSS module. Retrieving this information does not require the decoding of the navigation message, since spreading codes are known, and the visible IDs are available right after acquisition.

Let S_u and S_s be the set of IDs retrieved by the user and the server, respectively. Then the first spoofing check output is the Boolean results of the following expression

$$S_u \subseteq S_s. \tag{1}$$

The spoofing check reports no spoofing when the user sees only satellites that are supposed to be in view from his position. Otherwise, a flag is raised, meaning that there might be an ongoing spoofing attack. Note that we use the subset operator rather than equality, because even in nominal conditions some satellites may not be in view due to physical obstacles, e.g., buildings or trees.

The spoofing check (1) is designed to tackle attack 1 of Sect. 2.1. If the attacker does not null the legitimate signal (as discussed in the previous section), the user may be able to acquire also the legitimate satellites, even if tracking and position estimation are performed based on the more powerful spoofed satellites. Therefore, this spoofing check detects attacks inducing a fake position for which the visible satellites are different from those in view from the user true positions. If the true and fake positions are close enough to yield the same set of visible satellites, the attack goes undetected.

However, time also plays an important role, the satellites in view changes over time, therefore the acquired satellites may be inconsistent with the time/position spoofed by the attacker. Indeed, *time spoofing* is also an important attack to smartphones, since system level applications may rely on the GNSS time output and temporal inconsistencies can cause software failures (see [1]). Moreover, tampering with the navigation message (attack 2) can also result in such timing difference, triggering spoofing detection.

As previously mentioned, (1) can be evaluated without looking at the navigation message, and the attack 2 of Sect. 2.1 can easily go undetected if induced position and time satisfy the satellite visibility constraints. Therefore, to address navigation message tampering we compare the azimuth and elevation angles of the two satellites' lists.

The second check procedure works as follows. For every ID in S_u, retrieve its corresponding element in the server's list. If such element does not exist, the first spoofing check detects spoofing. Otherwise, compare the satellite's coordinates in the two lists. If the difference is above a suitable threshold, raise a spoofing warning, otherwise declare that no spoofing attack is occurring. Note that when comparing the satellites' coordinates, the threshold is needed whenever (as often occurs) communications and data processing of the CS architecture introduce a random delay. Indeed, even in nominal conditions, azimuth and elevation computed by client and server will not be the same due to the different computation instants. A timestamp sent by the user would solve the problem, but, at the same time, would pave the way for time spoofing attacks.

Unlike satellites' IDs, both azimuth and elevation can only be computed after decoding of the navigation message, therefore, attack 2 is immediately revealed when the navigation message is absent or corrupted in the specific fields used to compute the satellite position.

2.3.1 Stand-Alone Navigation Message Checks

Following the trace of the second proposed spoofing check, other integrity inspections aimed at verifying the compliance of the navigation message fields can be considered.

Some compliance check do not require the CS architecture, and can be performed offline. For example, navigation message specifications, are publicly available (e.g., see [20]), and some verification functions can be hard coded in the smartphone's software. Examples of compliance checks are:

- the *week number* filed must be equal for all satellites currently in view;
- the *issue of data clock* (IODC) and the *issue of data ephemeris* (IODE) are always between 0 and 1023.

2.3.2 Remark

The proposed spoofing checks are specific against the considered attacks, while more sophisticated attacks, e.g., a combination of GNSS spoofing and server hijacking, may go undetected. However, when a user can rely on such CS architecture and on the stand-alone techniques, the attacker's degrees of freedom are significantly reduced. Also, this approach assumes that the user has access to the server via a network connection, which is quite reasonable, given the smartphone-oriented use case.

2.4 Android Implementation

The architecture of Fig. 1 has been implemented, whereas the client is an Android APP running on the user smartphone, and the server is a desktop application.

Figure 2 shows two screenshots of the Android APP. In Fig. 2a the client has just acquired data from the GNSS module and the APP displays the list of visible satellites together with information on the decoded navigation message. At this stage, the APP shows the client's satellite list of Fig. 1.

Then, by tapping on the `Analyze` button, the client queries the server, which promptly answers with its own satellite list, and the spoofing checks is performed. This corresponds to Fig. 2b, where in the first half of the screen the two satellite lists are shown. The experiment has been carried out in nominal conditions, i.e., without

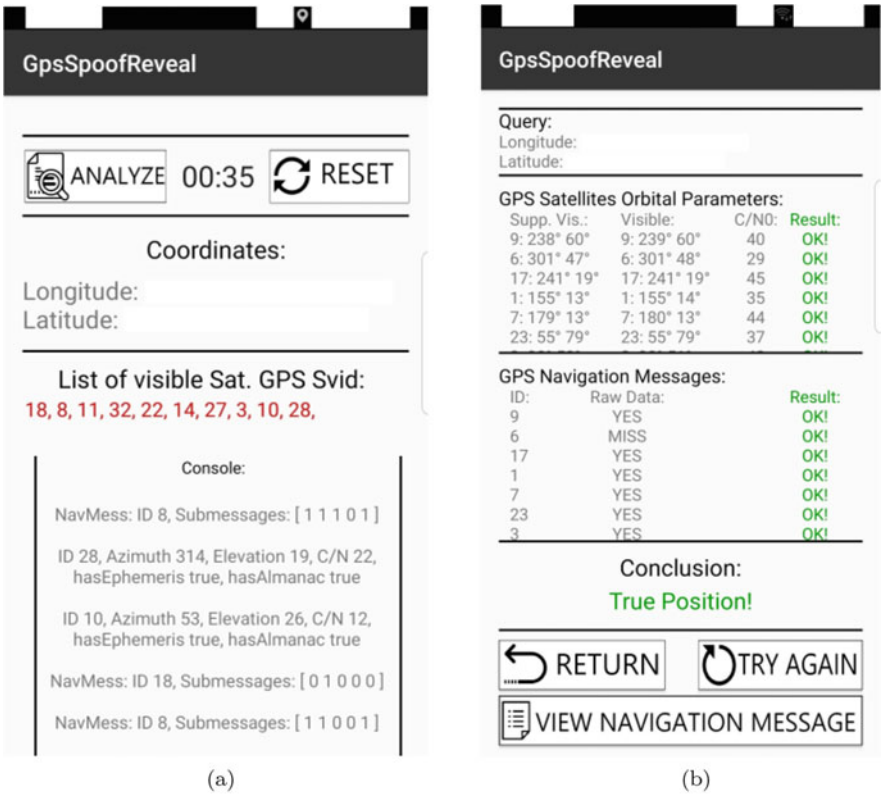


Fig. 2 Screenshots of the developed APP: data acquisition and spoofing checks with the data received from the server. (**a**) Acquisition of satellites at the client. (**b**) Spoofing checks on the visible satellites

spoofing, and the results show that all azimuth and elevation values differ at most by one degree, providing a design direction for the spoofing check threshold.

3 Network Connectivity

In this Section we report a novel technique that checks the consistency between the position estimates obtained by GNSS and the cellular network, summarizing our work [2]. For the latter estimate we propose two distinct solutions: one is based on the position of the base stations (BSs) (see Sect. 3.1) and the other directly on the smartphone position estimated by the cellular network (see Sect. 3.2). Both these solutions assume that the user has access to the network, and rely also on its security.

We have also implemented the proposed techniques in an Android APP and tested its effectiveness in detecting spoofing attacks.

3.1 BS-Position-Based Solution

A first simple solution to detect a spoofing attack is checking the consistency between the GNSS position and the region covered by the serving cell, as identified by the BS connected to the smartphone. With this technique we need to know: the position reported by the GNSS, the position of the serving BS, and its coverage area. We observe that the BS position is available since the second generation of cellular networks, i.e., with the global system for mobile communications (GSM); still, here we focus on a forth generation network, and Android application programming interface (API). By using the `TelephonyManager` class we obtain the cell ID and the location area code (LAC), that provide a unique identification of the BS in the network.

About the BS position, it can be easily retrieved by open databases available on Internet, such as OpenCellId [21] or Mozilla Location Service [22]. Note that a secure implementation of this solution would require secure communication with these servers and the assurance that the stored data are correct. As a proof of concept, the OpenCellId has been used in [2].

A third needed information is the coverage area. This is the most problematic data, which is not provided by available databases with good precision. For example, CellMapper [23] provides an approximated estimate of the coverage area, as shown in Fig. 3. Due the absence of the accurate coverage area, only qualitative information is obtained. A second option would be to model cells as Voronoi regions around each BS. Thus, by using the information on the BS position provided by OpenCellId, we can obtain the coverage information. Still, this procedure is based on a relevant assumption, and again it is hard to determine the resulting false-alarm and misdetection probabilities of spoofing detection.

Although this approach is very qualitative, it can protect against strong attacks to which current smartphones are subject, where the fake position is set thousands of kilometers from the true position (see [1, experiment 1]).

3.2 Network-Provided Position

A second, more accurate spoofing detection procedure checks the consistency between the position provided by the GNSS device and that provided by the cellular network. Indeed, from the 3rd generation of cellular systems, the network estimates the user position by exploiting directly the cellular signals, without resorting to the user GNSS device. In particular, we use the Android class `LocationManager`, which reports the position as given by the network provider, denoted network position (NP). By default, the NP is obtained by processing signals of WiFi access points nearby to the user. At the time of writing this Chapter, there is no clear distinction between the NP obtained only from cellular network signals and that obtained also from WiFi signals, in the Android documentation. However,

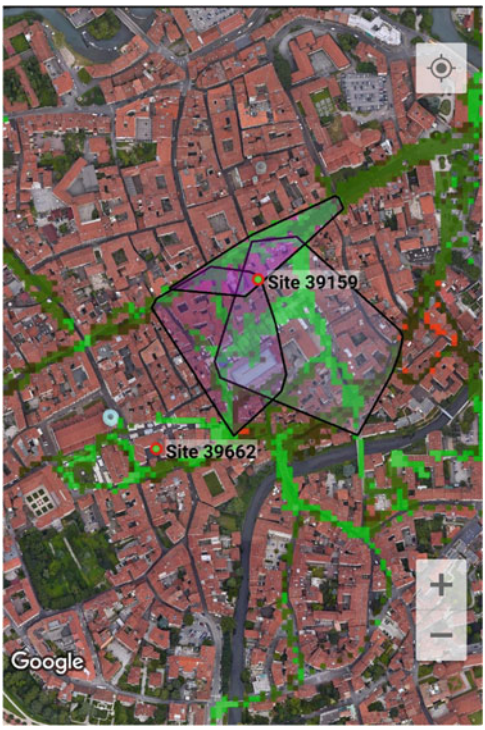


Fig. 3 Base stations and estimated coverage area. Screenshot from [23]

it is always possible to force the NP to rely only on the cellular network, by switching off the WiFi module. If in the future it will be possible to distinguish position estimations obtained from different signals (cellular network, WiFi, ...) the technique presented in this section can be easily extended to these richer scenarios.

In [2] an analysis of this spoofing detection technique has been carried out, exploiting the accuracy provided by the Android APIs on both GNSS position (GP) and NP estimates. In particular, starting from the two hypotheses

$$\begin{cases} \mathcal{H}_0 & \text{the GNSS module works correctly,} \\ \mathcal{H}_1 & \text{the GNSS module is under spoofing,} \end{cases} \tag{2}$$

the generalized likelihood ratio test (GLRT) has been derived, assuming a Gaussian distribution of the positions under hypothesis $\mathcal{H}_0$, i, l.

$$\hat{\mathcal{H}} = \begin{cases} \mathcal{H}_0 & \text{if } ||\hat{\boldsymbol{g}} - \hat{\boldsymbol{n}}|| < \gamma \\ \mathcal{H}_1 & \text{if } ||\hat{\boldsymbol{g}} - \hat{\boldsymbol{n}}|| \geq \gamma, \end{cases} \tag{3}$$

where γ is a threshold to be set for the desired false alarm probability, and $\hat{\boldsymbol{g}}$ and $\hat{\boldsymbol{n}}$ are the GP and NP, respectively.

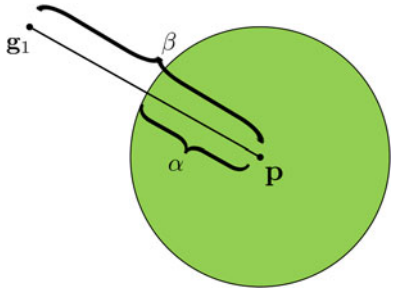


Fig. 4 α attack scenario, under hypothesis $\mathcal{H}_1$

When assuming specific attacks, the likelihood ratio test can be performed, which provides a more effective detection. In [2] two kinds of attacks have been considered: the α attack and the border spoofing attacks.

The α attack induces a fake position that is at least at distance α from the true one. In this scenario the measurement model under the two hypotheses is then

$$\mathcal{H}_0: \quad \hat{\boldsymbol{g}} = \boldsymbol{g}_0 + \xi_G e^{j\theta_G}, \quad \hat{\boldsymbol{n}} = \boldsymbol{p} + \xi_N e^{j\theta_N}, \tag{4a}$$

$$\mathcal{H}_1: \quad \hat{\boldsymbol{g}} = \boldsymbol{g}_1 + \xi_G e^{j\theta_G}, \quad \hat{\boldsymbol{n}} = \boldsymbol{p} + \xi_N e^{j\theta_N}, \tag{4b}$$

where $\boldsymbol{g}_0$ and $\boldsymbol{g}_1$ are the spoofed positions under $\mathcal{H}_0$ and $\mathcal{H}_1$, respectively, $\boldsymbol{p}$ is the true position, and the other parameters model the estimation errors. Note that it can also be $\boldsymbol{g}_0 = \boldsymbol{p}$ in the absence of spoofing. Due to the constraint on the distance of the spoofed position with respect to the correct one we have

$$||\boldsymbol{p} - \boldsymbol{g}_1|| \geq \alpha. \tag{5}$$

Figure 4 shows the α attack scenario under hypothesis $\mathcal{H}_1$, where β is the attack distance such that $\beta \geq \alpha$.

With the border spoofing attack the attacker aims at inducing a position that is beyond a known border, or inside a specific region in space. For example, this attack is meaningful against location-based services, where a service is granted only if the user is in a specific area, and the attacker wants to induce a position inside the specific area, in order to get access to the service. In [2] the specific area border is approximated as piece-wise linear, and the line closer to the correct position has known angular coefficient m and intercept q. The resulting scenario is shown in Fig. 5, where d_1 and d_2 are the distances from the border of $\boldsymbol{p}$ and $\boldsymbol{g}$, respectively. In this case, the measurement model is

$$\mathcal{H}_0: \quad \hat{\boldsymbol{g}} = \boldsymbol{p} + \xi_G e^{j\theta_G}, \quad \hat{\boldsymbol{n}} = \boldsymbol{p} + \xi_N e^{j\theta_N}, \tag{6a}$$

$$\mathcal{H}_1: \quad \hat{\boldsymbol{g}} = \boldsymbol{g} + \xi_G e^{j\theta_G}, \quad \hat{\boldsymbol{n}} = \boldsymbol{p} + \xi_N e^{j\theta_N}, \tag{6b}$$

where $\boldsymbol{g}$ is the spoofed position under $\mathcal{H}_1$.

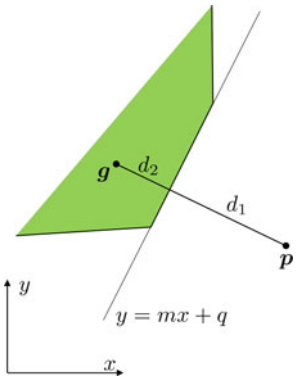


Fig. 5 Border spoofing scenario

While for a generic spoofing attack the GLRT uses the simple expression reported in (3), for both the α and the border spoofing attack we can provide specific defences requiring the following additional steps. First, the unknowns quantities $(\boldsymbol{p}, \boldsymbol{g}_0, \boldsymbol{g}_1)$ of (4), and $(\boldsymbol{p}, \boldsymbol{g})$ of (6), are estimated via a maximum likelihood approach, using the NP and GP measurements, as detailed in [2]. Then, the likelihood function can be computed as

$$\mathcal{L}(\hat{\boldsymbol{g}}, \hat{\boldsymbol{n}}) = \frac{p(\hat{\boldsymbol{g}}|\mathcal{H}_0)\, p(\hat{\boldsymbol{n}}|\mathcal{H}_0)}{p(\hat{\boldsymbol{g}}|\mathcal{H}_1)\, p(\hat{\boldsymbol{n}}|\mathcal{H}_1)}, \tag{7}$$

where the conditional probability density function $p(\cdot|\cdot)$ are derived from the measurement models (4) and (6), and computed using the estimated parameters. Finally, the detection test for both the α and the border spoofing attack is

$$\hat{\mathcal{H}} = \begin{cases} \mathcal{H}_0 & \text{if } \mathcal{L}(\hat{\boldsymbol{g}}, \hat{\boldsymbol{n}}) \geq \gamma', \\ \mathcal{H}_1 & \text{if } \mathcal{L}(\hat{\boldsymbol{g}}, \hat{\boldsymbol{n}}) < \gamma', \end{cases} \tag{8}$$

where for each value of the threshold γ' we obtain different performance in terms of false alarm and misdetection probability, i.e., the detection error tradeoff (DET).

3.3 Numerical Results

The idea of comparing the network-provided position with the GNSS position has been tested through an Android APP. An example of screenshot of the APP, under a spoofing attack is shown in Fig. 6: the two positions are shown, and a traffic light (showing red in this case) indicates that a spoofing attack has been detected. We recall that the two positions in Fig. 6 are the NP and the GP described in Sect. 3.2, and the underlying spoofing detection strategy follows the GLRT in (3), where the

Fig. 6 Screenshot taken from our APP, under a spoofing attack

threshold γ can be set by the user or it can be designed and hard coded, upon offline testing, in order to meet a certain false-alarm probability.

The false position in Fig. 6 is set to Lat. 40.7484404°, Long. −73.9878441°, i.e., the Empire State Building in New York City, USA. Note that this is the same setting of [1, Exp. 1]. Clearly our solution is able to detect the spoofing attacks where the fake position is far away from the correct one, a situation that remains undetected in nowadays smartphones, as shown in Table 1.

By collecting data from a measurement campaign, the APP has been tested and the DET curve of Fig. 7 shows the misdetection probability as a function of the false-alarm probability. The attack in this case is simulated by adding an offset δ to the GNSS position measurements. Both the GLRT test for a generic attack (continuous lines) and the test for α attack with $\alpha = 100$ m (dashed lines) have been performed. The continuous lines correspond to test (3) and, again, bigger δ means better detection performance. The detection test used for the α attack is, instead, given by (8). Note that in this case performance obtained with the two tests is almost identical.

Simulations have also been conducted to test specific α and border attack. For both attacks the detection test is given by (8), where however $\tilde{\boldsymbol{\theta}}$ changes with the attack, as seen in Figs. 4 and 5. Figure 8 shows the DET for α attack, $\alpha = 100$ m, $\boldsymbol{g}_0 = \boldsymbol{p}$, and different values of β. We observe that as the distance of the fake position from the correct position increases, the spoofing detection mechanism is

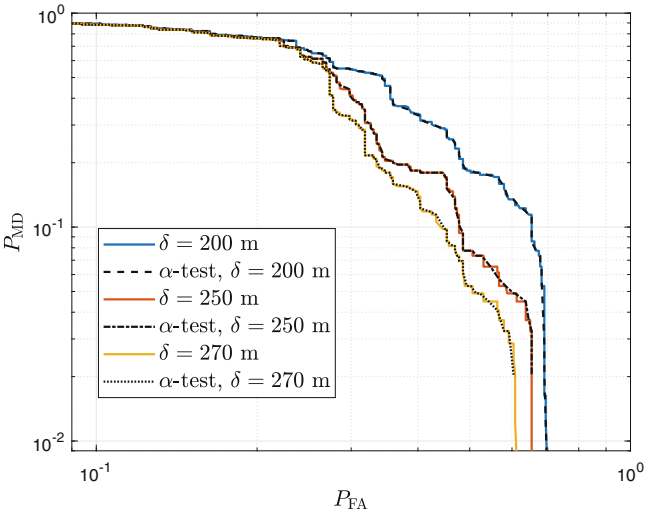


Fig. 7 DET on data from the measurement campaign, from [2]

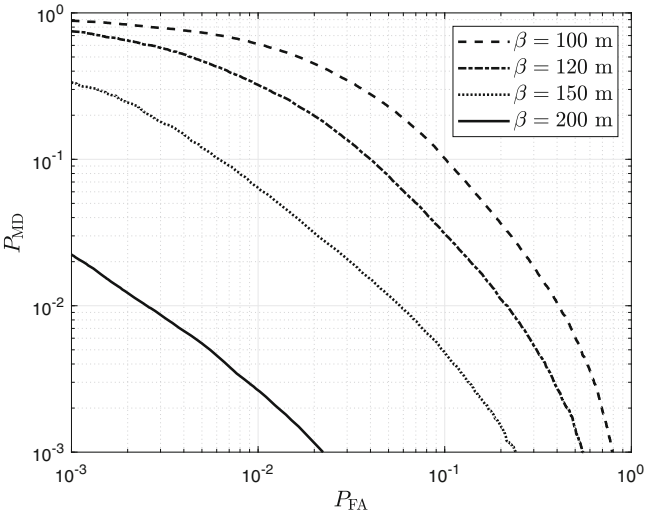


Fig. 8 DET for α attack and different values of β

more effective. For the border attack, Fig. 9 shows the DET for $d_2 = 0$ m and different values of d_1. Also in this case we observe that performance improves for attacks of more remote positions.

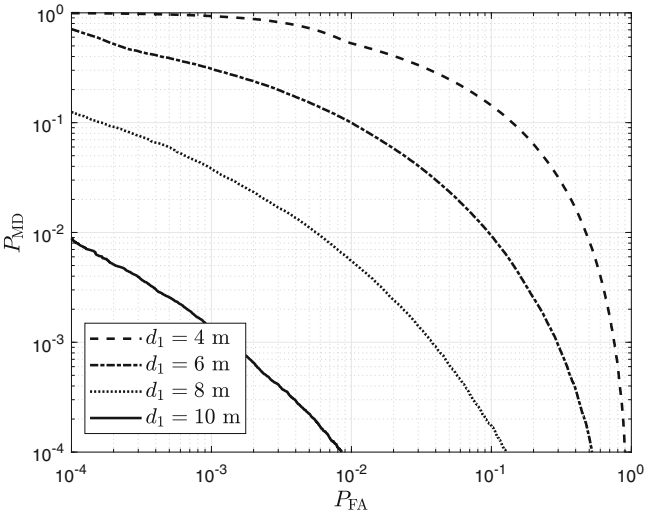


Fig. 9 DET for border attack, $d_2 = 0$ m and different d_1

4 IMU Data

This section tackles the use of opportunity data collected from integrated IMU for anti-spoofing purposes. This mechanism is well known in safety critical applications such as aviation, where several works have investigated the security improvement provided by high precision sensors, [8, 9]. Since raw GNSS measurements are now available to Android smartphones applications [24], several works are focusing on integrating GNSS and IMU measurements in mobile devices, not only for performance improvement, but also for security enhancement. In [25] a comparison of the performance improvement in the position, velocity, and time (PVT) between loosely and tightly coupled GNSS/IMU is carried out and compared to a system solely relying on GNSS. The result highlights how the tightly-coupled implementation boosts performance. Some relevant implementation issues are pointed out, such as the critical time synchronization between inertial-sensor measurements and GNSS chip set, the problem of data latency, and the management of different sampling frequencies. The authors of [26] and [10] perform quality assessment of measurements taken with IMU and GNSS chip sets in different mobile phone models. They point out how even measurements from low-cost IMUs of mobiles provide useful data for navigation integration. The performance of low-cost accelerometers for anti-spoofing in aviation is reviewed in [10], where high-frequency acceleration components are identified as a suitable source of randomness for authentication purposes, similarly to [8]. The work targets an attacker with imperfect information on the precise aircraft acceleration and develops a spoofing detection algorithm based on the decoupling of IMU and GNSS acceleration, allowing a direct comparison of acceleration for an unlimited

time window. The proposed detection algorithm is reviewed in different application scenarios such as railways and automotive, wherein is found to be less effective due to the lower intensity of high frequency components in the acceleration process. Spoofing detection in vehicular applications is investigated in [6], where a mobile device is used for comparing the absolute value of linear and angular acceleration with those obtained from the GNSS solution. This approach avoids the calibration of IMU and is invariant to manipulations to the device initial orientation. The automotive scenario is also the target application of [7], wherein the proposed solution integrates data from GNSS, IMU, and odometer. Differently from [10] and [6], the comparison domain is position and not acceleration, and the detection statistics are obtained as the norm of the difference between position vectors (from GNSS and from IMU/odometer). The novelty in this approach is the idea of performing GNSS based sensor calibration at fixed time intervals only if the spoofing detection algorithm confirms that the GNSS solution is authentic.

Several works in the literature have investigated the problem of anti-spoofing via integration and comparison with IMU data. The abundance of recent works on this idea confirms the interest of the community towards this topic.

4.1 Inertial Measurement Units

An IMU is usually composed by a combination of multiple accelerometers and gyroscopes that measure the force acting on them (and thus the resulting linear acceleration) and it angular velocity, respectively [27]. Some IMUs also integrate a magnetometer that measures the Earth magnetic field.

The general problem of navigation through sensor fusion, i.e., the integration of multiple positioning sources to obtain a more accurate and robust PVT solution, deals with data transformation between different coordinate frames. Following the approach of [28]:

Body frame, b: is the reference frame for IMU outputs. Its origin is located in the center of the accelerometer triad and the axes are generally aligned to the IMU case.

Navigation frame, n: is the target local geographical frame, where we want to measure the device PVT. In order to integrate the inertial IMU measurements we need to know the position and orientation of the b-frame with respect to the n-frame.

Inertial frame, i: is stationary with respect to the Earth. It has the origin in its center and the axis aligned with the stars;

Earth frame, e: rotates with the Earth (origin in the Earth center and axis fixed with respect to the Earth).

As reported in [28] the IMU outputs acceleration and angular velocity of the body frame relative to the inertial frame, with measurements that are expressed in the body frame (i.e., with the body frame as reference basis).

4.1.1 Gyroscope Measurements

The gyroscope measures the angular velocity of the body frame relative to the inertial frame, $\boldsymbol{\omega}_{\text{ib}}^{(\text{b})}$, where superscript (b) indicates that the vector is expressed through coordinates in the b-frame. For navigation purposes we are interested in the angular velocity of the sensor relative to the navigation frame expressed in the b-frame

$$\boldsymbol{\omega}_{\text{nb}}^{(\text{b})}(t) = \boldsymbol{\omega}_{\text{ib}}^{(\text{b})}(t) - R^{(\text{bn})}(t)(\boldsymbol{\omega}_{\text{en}}^{(\text{n})}(t) + \boldsymbol{\omega}_{\text{ie}}^{(\text{n})}), \tag{9}$$

where $R^{(\text{bn})}$ is the rotation matrix from the n-frame to the b-frame, $\boldsymbol{\omega}_{(\text{ie})}^{(\text{n})}$ (in rad/s) is the angular velocity of the earth frame relative to the inertial frame, and $\boldsymbol{\omega}_{\text{en}}^{(\text{n})}$ is the angular velocity of the n-frame relative to the e-frame.

In low grade applications it is customary to make the simplifying assumption that the n-frame is stationary with respect to the Earth, thus $\boldsymbol{\omega}_{\text{en}}^{(\text{n})} = 0$. Moreover, since $|\boldsymbol{\omega}_{\text{ie}}^{(\text{n})}| \approx 7.29 \cdot 10^{-5}$ rad/s, this contribution can also be assumed negligible. For ease of notation, from (9), we define the time varying vector $\boldsymbol{\omega}(t)$ as

$$\boldsymbol{\omega}(t) \triangleq \boldsymbol{\omega}_{\text{ib}}^{(\text{b})} \approx \boldsymbol{\omega}_{\text{nb}}^{(\text{b})}. \tag{10}$$

4.1.2 Accelerometer Measurements

The accelerometer measures the force acting on the sensor and computes the *specific* force, that is

$$\boldsymbol{a}_{\text{i}}^{(\text{b})}(t) = R^{(\text{bn})}(t)(\boldsymbol{a}_{\text{i}}^{(\text{n})}(t) - \boldsymbol{g}^{(\text{n})}), \tag{11}$$

where $\boldsymbol{g}^{(\text{n})}$ is the gravitational acceleration and $\boldsymbol{a}_{\text{i}}^{(\text{n})}(t)$ is the acceleration of the device relative to the i-frame. For navigation purposes we are interested in $\boldsymbol{a}_{\text{n}}^{(\text{n})}$, the acceleration of the device relative to the n-frame. The relationship between $\boldsymbol{a}_{\text{n}}^{(\text{n})}$ and $\boldsymbol{a}_{\text{i}}^{(\text{n})}$ is [28]

$$\boldsymbol{a}_{\text{i}}^{(\text{n})}(t) = \boldsymbol{a}_{\text{n}}^{(\text{n})}(t) + 2\boldsymbol{\omega}_{\text{ei}}^{(\text{n})} \times \boldsymbol{v}(t) + \boldsymbol{\omega}_{\text{ei}}^{(\text{n})} \times \boldsymbol{\omega}_{\text{ei}}^{(\text{n})} \times \boldsymbol{p}(t), \tag{12}$$

where $\boldsymbol{p}$ and $\boldsymbol{v}$ are the position and velocity of the device relative to the navigation frame.

In (12) the angular velocity of the Earth is assumed constant and the navigation frame is fixed to the Earth frame, which is reasonable when the travelled distance is negligible with respect to the Earth radius. This formulation is derived by using the relation between rotating coordinate frames. The last term of the sum represents the centrifugal acceleration, while the second is the Coriolis acceleration. The former is typically absorbed in the gravity vector and has a magnitude of around $3.39 \cdot$

$10^{-2}\mathrm{m}/s^2$ while the latter depends on the velocity of the object on which the IMU is mounted, and has a magnitude in the order of 10^{-3} for a speed of 120 km/h.

In order to simplify the model we assume these two terms to be negligible and we define the time varying vector

$$\boldsymbol{a}(t) \triangleq \boldsymbol{a}_{\mathrm{n}}^{(\mathrm{n})}(t) \approx \boldsymbol{a}_{\mathrm{i}}^{(\mathrm{n})}(t). \tag{13}$$

The physical relationships between $\boldsymbol{a}$, $\boldsymbol{p}$, and $\boldsymbol{v}$ are

$$\boldsymbol{v}(t) = \frac{\partial \boldsymbol{p}(t)}{\partial t}, \quad \boldsymbol{a}(t) = \frac{\partial \boldsymbol{v}(t)}{\partial t}. \tag{14}$$

4.1.3 Representation of Orientation

The acceleration $a(t)$ is not directly measured by the IMU. Indeed, the measured acceleration $a_b(t)$ is expressed with respect to the body frame rather than the navigation frame. In order to express $a(t)$ as a function of $a_b(t)$, a rotation of the former has to be performed and the entity and direction of this rotation is given by the orientation of the navigation frame with respect to the body frame.

Orientation can be parametrized in different ways. We consider unit quaternions, one of the most widely used orientation parametrization in estimation problems [28]. Unit quaternions are a 4-dimensional representation of orientation:

$$\mathbf{q} = (q_0, q_1, q_2, q_3)^T = \begin{pmatrix} q_0 \\ \mathbf{q_v} \end{pmatrix}, \ \mathbf{q_v} \in \mathbb{R}^3, \ ||\mathbf{q}||_2 = 1. \tag{15}$$

A rotation of a vector in $\mathbb{R}^3$ is a change of its direction while its length remains constant. The rotation of $\mathbf{x_a}$ into $\mathbf{x_b}$ can be expressed with unit quaternions as:

$$\mathbf{x_b} = \mathbf{q^{ba}} \odot \mathbf{x_a} \odot (\mathbf{q^{ba}})^c, \tag{16}$$

where the $\odot$ represents quaternion multiplication, that can be expressed in matrix form (see [28] for the derivation).

Rotations in $\mathbb{R}^3$ form the special orthogonal group, $SO(3)$, that is a matrix *Lie group*. As reported in [28], this allows to represent an orientation deviation with an exponential map over rotation vectors. An orientation with respect to the navigation frame, $\mathbf{q_t^{nb}}$ is thus represented in terms of a linearization point ($\tilde{\mathbf{q}}_\mathbf{t}^{\mathbf{nb}}$) and an orientation deviation parametrized by a rotation vector, $\boldsymbol{\eta_t}$, expressed in the body frame as

$$\mathbf{q_t^{nb}} = \exp\left(\frac{\bar{\boldsymbol{\eta}}_t}{2}\right) \odot \tilde{\mathbf{q}}_\mathbf{t}^{\mathbf{nb}}, \tag{17}$$

where $\bar{\boldsymbol{\eta}}_t = (0, \boldsymbol{\eta}_t{}^T)^T$, and $\boldsymbol{\eta_t} = \mathbf{n}\boldsymbol{\alpha}$ is a rotation vector, parametrized by a unit vector, $\mathbf{n}$ and rotation Euler angles $\boldsymbol{\alpha}$. The exponential operation is defined as

$$\exp(\bar{\eta}) = \cos||\eta||_2 + \frac{\bar{\eta}}{||\bar{\eta}||}\sin||\eta||_2. \tag{18}$$

In our case we are interested in expressing the orientation in time as a function of the angular velocity and the initial orientation. Therefore, (17) will be used, where the reference orientation $\tilde{\mathbf{q}}_{\mathbf{t}}^{\mathbf{nb}}$ is the initial orientation at time t_0, and the rotation vector is represented by the angle displacement, i.e., the time integral of the angular speed

$$\mathbf{q}^{\mathbf{nb}}(t) = \exp\left(\frac{\bar{\eta}(t)}{2}\right) \odot \tilde{\mathbf{q}}^{\mathbf{nb}}(t_0), \tag{19}$$

with

$$\eta(t) = \int_{t_0}^{t} \boldsymbol{\omega}(t)dt. \tag{20}$$

The acceleration in the navigation frame can thus be expressed in the following way with respect to the acceleration in the body frame and the relative orientation of the two coordinate frames

$$\mathbf{a}^{\mathbf{n}}(t) = \mathbf{q}^{\mathbf{nb}}(t) \odot \mathbf{a}^{\mathbf{b}}(t) \odot (\mathbf{q}^{\mathbf{nb}}(t))^{c}. \tag{21}$$

4.2 Measurement Error Models

Data from gyroscopes and accelerometers are corrupted by measurement noise. By collecting data from a stationary IMU standing on a flat surface the gyroscope is expected to measure only the earth rotation, while the accelerometers should measure the resulting acceleration that accounts for gravity and the centrifugal force. Over some tens of seconds the data seem to fit well a Gaussian distribution with non-zero mean.

4.2.1 Gyroscope

In general the noise can be divided into two distinct contributions: a slowly time varying bias $\boldsymbol{\delta}_{\omega,t}$ and a white noise component $\boldsymbol{e}_{\omega,t} \sim \mathcal{N}(0, \Sigma_\omega)$, with Σ_ω a 3×3 diagonal matrix. The subscript t denotes discrete time samples. Therefore, the measures can be written as

$$\boldsymbol{y}_{\omega,t} = \boldsymbol{\omega}(nT_\omega) + \delta_{\omega,t} + e_{\omega,t}, \tag{22}$$

where T_ω is the sampling period of the gyroscope.

There are two main approaches to model the bias, which is considered either constant or slowly time-varying in the time interval of the measurements. In the latter case, the bias is modeled as a random walk

$$\boldsymbol{\delta}_{\omega,t+1} = \alpha \boldsymbol{\delta}_{\omega,t} + \boldsymbol{e}_{\delta_{\omega,t},t}, \tag{23}$$

with $\alpha \in (0, 1)$, $\boldsymbol{e}_{\delta_{\omega,t},t} \sim \mathcal{N}\left(0, \Sigma_{\delta_{\omega,t}}\right)$ and $\Sigma_{\delta_{\omega,t}}$ 3×3 diagonal.

This model fits well the experimental data and can be verified by the means of the Allan variance.

4.2.2 Accelerometer

For the accelerometer the same observations of the gyroscope hold, and the noise has two contributions: a bias that is slowly time varying and a white noise. The measurement model for the accelerometer is:

$$\boldsymbol{y}_{a,t} = \boldsymbol{a}_{\mathrm{i}}^{(\mathrm{b})}(t) + \boldsymbol{\delta}_{a,t} + \boldsymbol{e}_{a,t}, \tag{24}$$

where T_a is the sampling period of the accelerometer, and $\boldsymbol{e}_{a,t} \sim \mathcal{N}\left(0, \Sigma_{\delta_{a,t}}\right)$, with again $\Sigma_{\delta_{a,t}}$ a 3×3 diagonal matrix.

4.2.3 GNSS Module

The lower level output of the GNSS module are the pseudorange measurements $\rho_t^{(s)}$ and carrier phase measurements $\phi_t^{(s)}$, where the superscript s denotes the number of satellites in view. From the raw measurements and the ephemeris data we can compute PVT.

GNSS measurements are corrupted by additive noise, i.e.,

$$\boldsymbol{y}_{p,t} = \boldsymbol{p}(nT_p) + \boldsymbol{e}_{p,t}, \tag{25}$$

where T_p is the sampling period of the GNSS module. A reasonable model for the additive error process is a Gauss-Markov process defined as follows

$$\boldsymbol{e}_{p,t+1} = \exp(-\beta T_p)\boldsymbol{e}_{p,t} + \nu_t, \tag{26}$$

where β is a parameter describing the correlation between successive samples, $\nu_t \sim \mathcal{N}\left(0, \Sigma_p\right)$. Parameters β, T_p, and Σ_p are tabulated in [29, 30].

4.3 Extended Kalman Filter

The KF is an efficient approach to evaluate the state of a complex system, governed by known laws and described by noisy measurements. In our problem, the system under exam is a moving object and the laws that describe its state (position, velocity, and time) are the physics laws of motion. The available measurements can be used for state evaluation according to a known measurement model (described in Sect. 4.2). The KF approach is based on the assumption that the noise corrupting both the measurements and the state estimate is Gaussian and that the equations describing the evolution of both the state and the measurements are linear. As the motion of an accelerating object cannot be represented by linear equations (since the object position is part of the state variables), it is customary to adopt the extended Kalman filter (EKF) instead. The EKF exploits a linearization of the non-linear equations that describe the state and measurement evolution through the Jacobian of non-linear functions. While the KF provides an optimal estimate, the EKF uses approximation and therefore it is not optimal. The EKF can be used to estimate the PVT of a moving object through the iterative repetition of two steps:

time update: the motion model is used to "predict" the state of the next time step;
measurement update: the predicted state estimate is updated according to the current measurement and the measurement model.

One of the most common implementations of EKF integrating GNSS and IMU uses the measurements from the former in the measurement update step, while the measurements from the latter are used in the time update step to predict the next state value. A complete derivation of the EKF equations and matrices can be found in [28].

4.4 Innovation Testing

Innovation testing is a spoofing detection approach that exploits the EKF designed for sensor fusion and navigation. Typically, such EKF have position, velocity, and orientation as state, and IMU and GNSS as measurements.

The innovation step (which is part of the measurement update step), in any linear KF (a similar expression holds for EKF), is

$$\boldsymbol{i}_k = \boldsymbol{z}_k - H\hat{\boldsymbol{x}}_{k|k-1}, \tag{27}$$

where $\hat{\boldsymbol{x}}_{k|k-1}$ is a prediction of the current state. The covariance matrix of $\boldsymbol{i}$ is known [31] and denoted by P_k. Then, by normalizing $\boldsymbol{i}_k$ by its covariance matrix, we obtain the test statistic

$$\beta_k = \boldsymbol{i}_k^t P_k \boldsymbol{i}_k, \tag{28}$$

which can be shown to be Chi-squared distributed with as many degree of freedom as the dimension of the measurement vector $\boldsymbol{z}$ [31].

Innovation testing for anti-spoofing leverages the fact that under spoofing, the state prediction derived from the IMU measurements are likely to disagree with the GNSS measurements (arbitrarily forged by the attacker). Therefore under attack $\boldsymbol{x}_{k|\hat{k}-1}$ is no longer a good prediction of the state, causing the absolute value of the innovation to increase. Then, under spoofing β_k is no longer Chi-squared distributed and the spoofing detection is performed, in the framework of binary hypothesis testing (see also (2)), according to

$$\hat{\mathcal{H}} = \begin{cases} \mathcal{H}_0 & \beta_k \sim \chi^2 \\ \mathcal{H}_1 & \beta_k \not\sim \chi^2. \end{cases} \tag{29}$$

Innovation testing is common in the literature, especially in aviation scenarios [4, 9].

In order to evaluate the results of innovation testing on a simple, analytic scenario, let us simulate a spoofing attack by designing a legitimate trajectory (LT), i.e., the trajectory that the user physically follows, and a spoofing trajectory (ST), i.e., the trajectory that the spoofer induces to the user. The corresponding IMU and GNSS measurements have been generated according to the measurement model described in Sect. 4.2. Let us recall that under $\mathcal{H}_0$ both GNSS and IMU follow the LT, while under $\mathcal{H}_1$ the GNSS module follows the ST.

The ST initially matches the LT, but then diverges by an angle θ. This is done by fixing 3 waypoints for the ST, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, and $\boldsymbol{w}_3$, such that

$$\boldsymbol{w}_1 = [0, 0, 0]^t, \tag{30}$$

$$\boldsymbol{w}_2 = [10, 3, 0]^t, \tag{31}$$

$$\boldsymbol{w}_3 = \boldsymbol{w}_2 + R_\theta \boldsymbol{w}_2, \tag{32}$$

where R_θ is a rotation matrix that rotates any vector by an angle θ in the (x, y) plane. We then create a temporal axis by fixing a time of arrival at each waypoint and specifying a trajectory sampling time. A cubic interpolation generates the intermediate points between waypoints using the time axis as interpolation query. Velocity and acceleration profiles are computed by numerical derivation of position vector.

Using the same time axis, we specify also an orientation profile, i.e., a quaternion for each time instant describing the orientation of the body frame with respect to the navigation frame. From the orientation profile we compute the angular velocity in the body frame at each time instant, such that

$$\omega_t^{\mathrm{b}} = \left(\boldsymbol{q}_t^{(\mathrm{nb})}\right)^{\mathrm{c}} \odot \boldsymbol{q}_{t+1}^{(\mathrm{nb})}. \tag{33}$$

From angular velocity, acceleration and position profiles we can generate GNSS and IMU measurements.

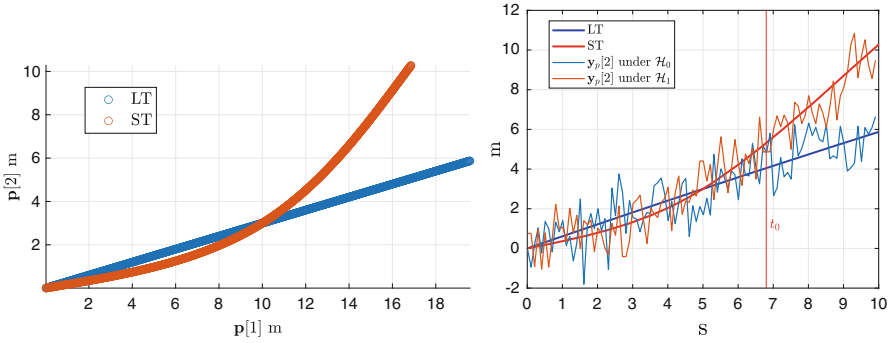


Fig. 10 Simulation scenario. (**a**) Scatter plot of $\boldsymbol{p}[1]$ and $\boldsymbol{p}[2]$ of the spoofed and authentic trajectory. $\theta = \pi/6$. (**b**) $\boldsymbol{p}[2]$ of the spoofed and authentic trajectory, versus time. Thinner lines are the corresponding measurements

Figure 10 shows the LT and the ST. Figure 10b shows a 2D scatter plot of $\boldsymbol{p}$, where divergence between LT and ST starts at point $(10, 3)$ and $\theta = \pi/6$. Before diverging, the two trajectories are not exactly the same because of the cubic interpolation that avoids singularities in later numerical derivations. Figure 10b shows the same trajectories (only the second component of each 3D position vector) as function of time together with the corresponding GNSS measurements, where the velocity absolute value is constant.

4.4.1 Analytical Results

We expect that the test statistic β(coinciding with the normalized innovation) is Chi-squared distributed with 3 degrees of freedom under $\mathcal{H}_0$, since the measurement vector of the EKF is 3-dimensional. Hence we have

$$\beta \sim \chi^2_3, \quad p(\beta|\mathcal{H}_0) = \frac{1}{2^{3/2}\Gamma(3/2)}\beta^{3/2-1}e^{-\beta/2}, \tag{34}$$

where $\Gamma(\cdot)$ is the well-known gamma function. Figure 11a shows $\hat{p}(\beta_k|\mathcal{H}_0)$ together with $p(\beta|\mathcal{H}_0)$ and we can see how the two distributions match, as expected. Then we apply the EKF on the ST and Fig. 11b shows how in this case the innovation test is not chi-squared distributed and hence spoofing detection can be performed.

The measurement frequency was set to 100 Hz for the IMU and 10 Hz for GNSS. A window of 1 s worth of innovation values was used for detection purposes, with varying the test instant t_0 and the trajectory angle θ. The DET curves were derived through Monte Carlo simulations, by collecting the statistics of the normalized innovation at different time instants, both in the authentic and spoofing case. The

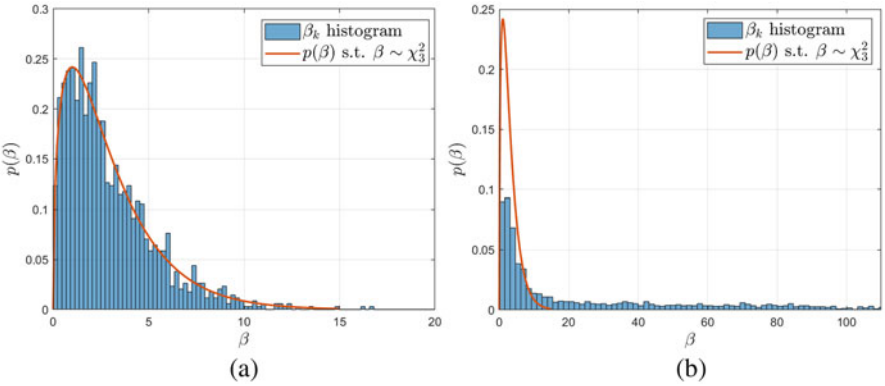


Fig. 11 Histogram plots of β_k under a LT and ST, i.e., under $\mathcal{H}_0$ and $\mathcal{H}_1$. (**a**) Histogram plot of $\beta_k|\mathcal{H}_0$. (**b**) Histogram plot of $\beta_k|\mathcal{H}_1$

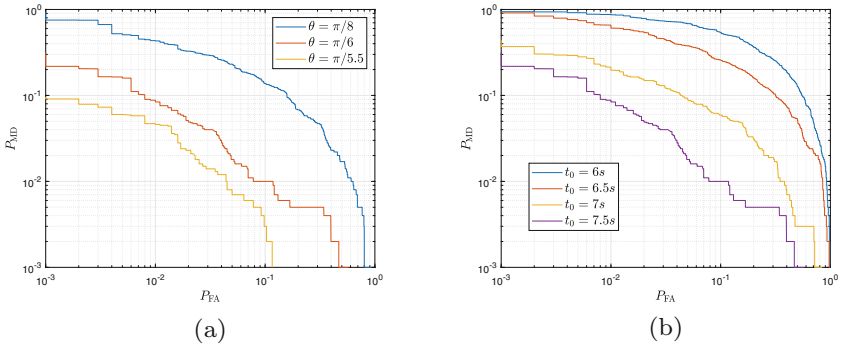


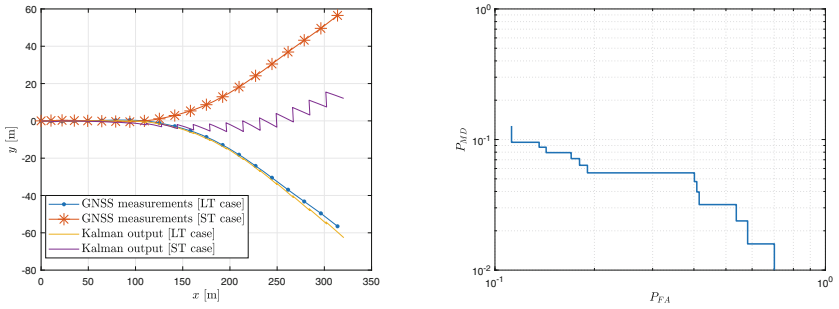
Fig. 12 DET curves for spoofing detection with the EKF. (**a**) DET curves for different θ ($t_0 = 7.5$s). (**b**) DET curves for different t_0 ($\theta = \pi/6$)

results are reported in Fig. 12. It is noticeable that, as expected, the more the two trajectories diverge, the more effective is the spoofing detection.

4.4.2 Testing on Real World Data

In the following some results are presented from the processing of measurements gathered from a Novatel sensor [32].

Figure 13a shows part of the LT and ST used for this experiment. The LT is taken directly from the available GNSS measurements, while the ST was obtained from the LT, such that the two trajectories diverge symmetrically. In both scenarios the IMU measurements are the same.



(a) (b)

Fig. 13 Experimental scenario and innovation testing performance. (**a**) LT and ST example with data from [32]. In both cases it is shown the position as estimated by the KF. (**b**) DET resulting from the application of innovation testing the scenario of Fig. 13a

The spoofing detection experiment was performed by feeding the EKF with the two trajectories and the IMU measurements. Figure 13a shows the position estimates of the EKF in both cases, making it possible to see the effect of measurement inconsistencies in the spoofing case. Figure 13b shows, instead, the DET resulting from innovation testing. False alarm and misdetection probabilities are estimated from 10 different portions of trajectory that are similar to the LT in Fig. 13a, for a total of 4×10^4 IMU samples and 210 GNSS samples.

4.4.3 Testing on a Software Receiver

In the following we evaluate the performance of innovation testing by exploiting the GNSS software receiver built by the University of Padova. The innovation testing module has as input the position computed by the PVT module and generates acceleration and gyroscope noisy measurements, according to the model in Sect. 4.2. The GNSS signal that is fed as input to the software receiver is generated with the c++ signal generator built by the University of Padova.

For both the nominal and the spoofing scenarios, 150 s worth of GNSS signal were generated. This time in the nominal scenario the receiver is stationary, therefore the IMU records only Gaussian noise. In the spoofing scenario the attacker is assumed to fake a stationary position, while the receiver is actually moving with constant acceleration. Indeed, the GNSS module computes a stationary position in the spoofing scenario, while the IMU measures a constant acceleration of 3 m/s^2 in magnitude.

Performance in terms of DET is shown in Fig. 14. The results are in the same order of magnitude of those in Fig. 13b.

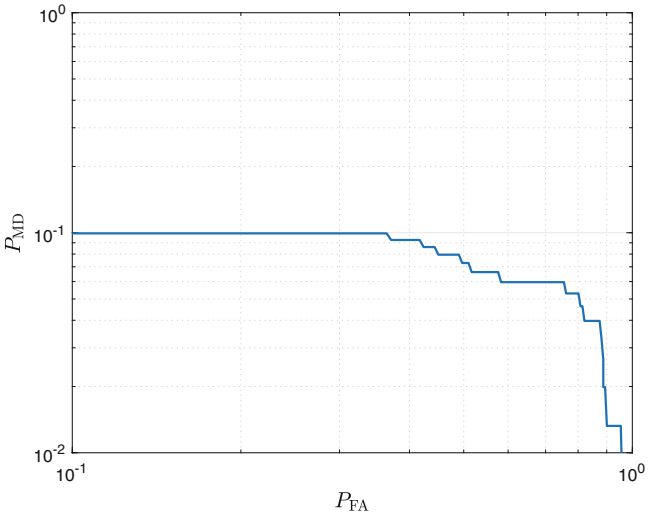


Fig. 14 DET resulting from innovation spoofing applied to the scenario described in Sect. 4.4.3

5 Conclusions

We have proposed various spoofing detection techniques to be applied on smartphones, using context information coming from other components, such as the cellular network or the IMU, or through consistency-checks of the received signals. All these techniques have been tested in APPs developed in Android and effects of the detection parameters have been studied in order to achieve a desired trade-off between false-alarm and misdetection probabilities. We have also shown the effectiveness of these defence strategies against various attacks previously reported in the literature.

Acknowledgments The authors gratefully acknowledge the contributions of Amedeo Pachera and Giovanni Carollo for developing the apps, and Marco Ceccato for his support with some simulations.

References

1. Ceccato, S., Formaggio, F., Caparra, G., Laurenti, N., Tomasin, S.: Exploiting side-information for resilient GNSS positioning in mobile phones. In: Proc. 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), pp. 1515–1524 (Jun 2018)
2. Formaggio, F., Ceccato, S., Basana, F., Laurenti, N., Tomasin, S.: GNSS spoofing detection techniques by cellular network cross-check in smartphones. In: Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019), pp. 3904–3916, Miami, Florida (September 2019)

3. Qiao, Y., Zhang, Y., Du, X.: A vision-based gps-spoofing detection method for small uavs. In: 2017 13th International Conference on Computational Intelligence and Security (CIS), pp. 312–316 (Dec 2017)
4. Kerns, A.J., Shepard, D.P., Bhatti, J.A., Humphreys, T.E.: Unmanned aircraft capture and control via gps spoofing. J. Field Rob. **31**, 617–636 (2014)
5. Kwon, C., Liu, W., Hwang, I.: Analysis and design of stealthy cyber attacks on unmanned aerial systems. J. Aerosp. Inf. Syst. **11**, 525–539 (2014)
6. Curran, J.T., Broumendan, A.: On the use of low-cost IMUs for GNSS spoofing detection in vehicular applications. In: International Technical Symposium on Navigation and Timing (ITSNT) 2017, Toulouse, France (2017)
7. Broumandan, A.. Lachapelle, G.: Spoofing detection using gnss/ins/odometer coupling for vehicular navigation. Sensors (Apr 2018)
8. Tanil, C., Khanafseh, S., Pervan, B.: Impact of wind gusts on detectability of GPS spoofing attacks using RAIM with INS coupling. In: Proc. ION Conference, pp. 674–686 (April 2015)
9. Tanil, C., Khanafseh, S., Pervan, B.: An INS monitor against GNSS spoofing attacks during GBAS and SBAS-assisted aircraft landing approaches. In: Proc. 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016), pp. 2981–2990, Portland, Oregon (Sept. 2016)
10. Lo, S., Chen, Y.H., Reid, T., Perkins, A., Walter, T., Enge, P.: Keynote: The benefits of low cost accelerometers for gnss anti-spoofing. In: ION 2017 Pacific PNT Meeting (2016)
11. Zeng (Curtis), K., Liu, S., Shu, Y., Wang, D., Li, H., Dou, Y., Wang, G., Yang, Y.: All your GPS are belong to us: Towards stealthy manipulation of road navigation systems. In: 27th USENIX Security Symposium (USENIX Security 18), pp. 1527–1544. USENIX Association, Baltimore, MD (2018)
12. Fernández-Hernández, I., Rijmen, V., Seco-Granados, G., Simon, J., Rodríguez, I., David Calle, J.: A navigation message authentication proposal for the galileo open service. Navig. J. Inst. Navig. **63**(1), 85–102 (2016)
13. Caparra, G., Sturaro, S., Laurenti, N., Wullems, C., Ioannides, R.T.: A novel navigation message authentication scheme for gnss open service. In: Proc. ION GNSS, pp. 2938–2947 (2016)
14. Caparra, G., Curran, J.T.: On the achievable equivalent security of GNSS ranging code encryption. In: Proc. 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), pp. 956–966. IEEE (2018)
15. Jahromi, A.J., Broumandan, A., Daneshmand, S., Lachapelle, G., Ioannides, R.T.: Galileo signal authenticity verification using signal quality monitoring methods. In: Proc. 2016 International Conference on Localization and GNSS (ICL-GNSS), pp. 1–8. IEEE (2016)
16. Pagot, J.-B., Thevenon, P., Julien, O., Amarillo-Fernandez, F., Maillard, D.: Signal quality monitoring for new gnss signals. In: Proc. ION GNSS 2016, 29th International Technical Meeting of the Satellite Division of the Institute of Navigation, pp. pp–1750 (2016)
17. Formaggio, F., Tomasin, S., Caparra, G., Ceccato, S., Laurenti, N.: Authentication of galileo GNSS signal by superimposed signature with artificial noise. In: Proc. 2018 26th European Signal Processing Conference (EUSIPCO), pp. 2573–2577. IEEE (2018)
18. Formaggio, F., Tomasin, S.: Authentication of satellite navigation signals by wiretap coding and artificial noise. EURASIP J. Wirel. Commun. Netw. **2019**(1), 98 (2019)
19. ftp://cddis.gsfc.nasa.gov/. Last accessed: 9/12/2019
20. https://www.gps.gov/technical/icwg/. Last accessed: 9/12/2019
21. Opencellid: https://www.opencellid.org
22. Mozilla Location Service: https://location.services.mozilla.com/
23. cellmapper: https://www.cellmapper.net
24. GSA: Using GNSS raw measurements on android devices, Jan 2018
25. Falco, G., Pini, M., Marucco, G.: Loose and tight gnss/ins integrations: Comparison of performance assessed in real urban scenario. Sensors (Jan 2017)
26. Gikas, V., Perakis, H.: Rigorous performance evaluation of smartphone GNSS/IMU sensors for ITS applications. Sensors (Aug 2016)

27. Starlino: A guide To using IMU (accelerometer and gyroscope devices) in embedded applications. https://www.starlino.com/imu_guide.html, [Posted: 29 December 2009]
28. Kok, M., Hol, J.D., Schön, T.B.: Using inertial sensors for position and orientation estimation. Foundations and Trends in Signal Processing, vol. 11, pp. 1–153 (2017). https://doi.org/10.1561/2000000094
29. Niu, X., Chen, Q., Zhang, Q., Zhang, H., Niu, J., Chen, K., Shi, C., Liu, J.: Using Allan variance to analyze the error characteristics of GNSS positioning. GPS Solutions **18**(2), 231–242 (2014)
30. Rankin, J.: GPS and differential gps: an error model for sensor simulation. In: Position Location and Navigation Symposium, pp. 260–260 (1994)
31. Liu, Y., Li, S., Fu, Q., Liu, Z.: Impact assessment of gnss spoofing attacks on ins/gnss integrated navigation system. Sensors (2018)
32. Novatel: Product home page. Available at https://docs.novatel.com/OEM7/Content/Home.htm