electronics

MDPI

Article

GNSS Spoofing Modeling and Consistency-Check-Based Spoofing Mitigation with Android Raw Data

Enrique Takiguchi Medina [1,2] and Elena Simona Lohan [1,*]

[1] Tampere Wireless Research Center, Electrical Engineering Unit, Tampere University, 33520 Tampere, Finland; enrique.takiguchi@autonoma.cat
[2] Universitat Autonoma de Barcelona, 08193 Barcelona, Spain
* Correspondence: elena-simona.lohan@tuni.fi

Abstract: Spoofing events are increasingly affecting the performance of devices and operations relying on Global Navigation Satellite Systems (GNSSs). Developing powerful and robust GNSS spoofing detection and mitigation algorithms is an important endeavor in the GNSS community nowadays; some of the challenges in this field are limited access to spoofing measurement data, as spoofing over wireless channels is not legally allowed and in-lab spoofing emulators are not necessarily able to precisely capture the effects of radio channels, and the fact that classical Receiver Autonomous Integrity Monitoring approaches are typically quite complex, especially when dealing with complex or targeted spoofers. Our paper addresses these two challenges, first, by proposing a targeted spoofing model with a variable number of spoofed satellites, starting from Android raw pseudorange measurements, and second, by introducing a consistency-check-based iterative approach for spoofing detection and mitigation. We test our solution in various dynamic scenarios (bus, walk, ferry, car, flight, and bike), and we show that the positioning error correction rates depend on the number of spoofing pseudorandom (PRN) codes, as well as on the spoofing error introduced by our model. We also show that a large part of the spoofing errors can be mitigated with the proposed algorithms if the number of spoofed satellites (or pseudoranges) is sufficiently low with respect to the total number of visible satellites.

Keywords: global navigation satellite systems (GNSSs); GNSSlogger; measurements; raw Android data; spoofing

Academic Editor: Jung Min Pak

Received: 24 January 2025
Revised: 16 February 2025
Accepted: 20 February 2025
Published: 24 February 2025

Citation: Medina, E.T.; Lohan, E.S. GNSS Spoofing Modeling and Consistency-Check-Based Spoofing Mitigation with Android Raw Data. *Electronics* **2025**, *14*, 898. https://doi.org/10.3390/electronics14050898

1. Introduction

In a globalized world where Global Navigation Satellite Systems (GNSSs) are playing an increasingly important role in modern society, the increasing frequency and scope of spoofing attacks pose a serious threat to the integrity of GNSS-based systems. GNSS spoofing attacks were theorized with the inception of the Global Positioning System (GPS) in the 1970s; however, it was not until the full development and testing of the first spoofers that spoofing attacks in GNSS bands and their mitigation became relevant. Today, low-cost Software-Defined Radio (SDR) GNSS transmitters are relatively easy to acquire, making spoofing attacks feasible to reproduce with little budget. Although many spoofing models and solutions have been studied so far in the literature, as described in more detail in Section 2, the recent availability of GNSS pseudorange data on Android devices [1,2] has opened new avenues of research in the field of GNSS spoofing and spoofing mitigation. For example, the authors in [3] proposed four spoofing detection methods based on data collected through Android phones and developed a GNSSAlarm application to test them.

Out of the four proposed spoofing methods in [3], two of them rely on additional non-GNSS data (e.g., comparing the GNSS-based location with the non-GNSS location provided by the network based on cellular or WiFi data or using additional 'mock-location' data from external providers to cloak the true location); the other two methods rely purely on GNSS navigation data, such as the differences between the system time and the time stamps provided by GNSSs or the jumps or discrepancies in signal quality, measured via Automatic Gain Control (AGC) or the Carrier-to-Noise Ratio (C/N_0). One distinction from the prior literature on spoofing is that in our paper, we focus solely on GNSS data, and we do not rely on additional sensors or external information, such as information provided by the cellular network. In [4], the authors proposed a minor hardware enhancement in the form of a signal-blocking shield to be installed in GNSS receivers that take advantage of receiver rotations, creating a virtual antenna array in the receiver able to differentiate between genuine and fake signals. The method proposed in [4] emulates spoofing attacks and mitigate them, starting from GNSSlogger data. Their spoofing attack model is a complex model based on modifying the existing open-source HackRF One SDR and requiring, in addition to the HackRF One SDR, also a Raspberry Pi, a portable power bank, and an antenna. The spoofing modeling approach in [4] is completely different from the simple model proposed here, which relies only on the observation data received from GNSS satellites in the sky and a Matlab-based algorithm.

In addition to the above-mentioned few studies dedicated to spoofing models and spoofing mitigation with Android raw GNSS data, such raw GNSS data have also been investigated with various algorithms for robust positioning, such as in [5], which discusses various datasets and tools for evaluating positioning performance; in our previous work in [6], which explores the use of raw GNSS data along with other sensor data from Android devices; or in [7], which employs techniques such as Moving Horizon Estimation (MHE) and Extended Kalman Filter (EKF) to handle noise and improve accuracy, which can be relevant to detecting and mitigating interferences.

To sum up, compared with the above-mentioned spoofing detection studies from the literature, our method couples the statistical modeling of different variables with the use of historical data in order to optimize the search for a subset of genuine satellites, thus enabling fast and precise spoofing mitigation. Our method relies solely on the most basic information provided by the GPS and Galileo constellations, making the algorithm accessible to any GNSS receiver with minimal computational capabilities and without requiring additional sensor technology.

The goal in this paper is twofold: first, proposing a straightforward (and easy-to-replicate) model of spoofing starting from Android raw GNSS measurement data collected in various dynamic scenarios (such as walking, going by bus, traveling to a place, etc.), and second, introducing an algorithm based on statistical modeling and consistency checks on subsets of visible satellites for spoofing detection and mitigation. There are three main contributions of this paper: (i) adopting a straightforward spoofing model with a variable number of spoofed satellites, starting from Android raw GNSS measurements; (ii) studying the impact of increasing the number of spoofed satellites on positioning accuracy in a variety of dynamic scenarios; and (iii) proposing a model-driven consistency-check algorithm for spoofing detection and mitigation. The first author has initially proposed the spoofing model in his bachelor thesis at Tampere University [8]; however, the extensive testing in various scenarios, the relaxation of algorithm assumptions, and the complexity studies are proposed for the first time in this paper.

The rest of the paper structure is as follows: Section 2 gives an overview of the GNSS spoofing problem and a brief overview on the state of the art in this field; Section 3 describes the methodology adopted in this paper and the sources of measurements used in the result

parts; Section 5 presents the spoofing model with a user-defined number of spoofed satellites among the visible ones, starting from Android measurements; Section 6 describes in detail the proposed algorithm for spoofing detection and mitigation; Sections 7 and 8 show a scenario-by-scenario performance analysis and some comparative results for several scenarios, respectively; Section 9 relaxes the assumptions regarding the spoofing model and verifies that our spoofing mitigation model also holds under relaxed conditions; Section 10 addresses the complexity of the proposed algorithm; and Section 11 presents the conclusions.

2. GNSS Spoofing Problem—Overview

Spoofing attacks can affect any type of GNSS receiver, for example, causing a yacht to steer off its course or forcing a drone to fall. Although many spoofing attacks can be detected and palliated via human intervention and backup positioning systems, some other systems, such as UAVs (Unmanned Aerial Vehicles) or small airplanes relying solely on GNSS location, tracking, and navigation, are more susceptible to attacks.

A spoofer replicates the RF carrier(s), the pseudorandom (PRN) codes of satellites in the sky, and the navigation data (including ephemerides) from each satellite in the sky and generates fake signals to deceive the receiver [9]. The amplitudes, code phases, and carrier phases of the spoofing signals can be different from the genuine signals in the sky (e.g., a higher-amplitude spoofing signal could mask completely the genuine signal, while an equal- or smaller-amplitude spoofing signal with a different code phase could appear as a multipath component at the receiver). The number of spoofing signals can be equal to the number of genuine satellites in the sky, but spoofing with a partial number of spoofed satellites or with signals from satellites not visible in the sky is also possible. Spoofing attacks can be divided into several classes [10,11]:

- Simplistic attacks, such as meaconing or replay attacks: In a simplistic attack, a GNSS signal generator (e.g., SDR-based) is connected to an antenna, and it broadcasts spoofing GNSS signals unsynchronized to the sky signal. In the particular case of a meaconing or replay attack, a spoofer re-broadcasts the captured signals from the satellites in the sky with a certain delay; they are typically untargeted attacks, spoofing all GNSS receivers in range but not necessarily a particular target receiver.
- Intermediate spoofing attacks are more advanced than the simplistic or untargeted ones but not as sophisticated as the most complex attacks. Such spoofer is usually able to align the frequency and the code with those of the satellites in the sky.
- Advanced or sophisticated spoofing attacks: A spoofer mimics the genuine GNSS signals in the sky by generating single-frequency or multi-frequency GNSS signals that are synchronized in time and frequency with the legitimate signals at the correct receiver position. These are usually targeted attacks, when the spoofer also gathers information about the target receiver(s), it is usually in the vicinity of a target receiver and transmits the spoofing signals at higher power than the genuine/sky signals in order to mask the genuine ones.

Self-consistent spoofers [12] are those spoofers which generate small pseudorange residuals, which are able to pass the consistency checks of a GNSS receiver endowed with Receiver Autonomous Integrity Monitoring (RAIM). A meaconer, for example, produces self-consistent spoofing signals.

Regarding methods to deal with GNSS spoofing attacks, many different detection techniques have been developed, and they can be grouped as follows [10]:

- Pre-correlation techniques, which use the signal before the acquisition stage of a GNSS receiver [13,14]. Such techniques can rely, for example, on multiple antennas at

the receiver, on Automatic Gain Control (AGC) values, or on Radio Frequency (RF) fingerprints available in the spectrograms of the raw in-phase/quadrature (I/Q) data that enter in a receiver [14].

- Post-correlation techniques, which use the signal after the acquisition and possibly tracking units, but before the pseudorange computations. Examples include the receiver power or Carrier-to-Noise Ratio monitoring techniques, RF fingerprinting methods based on time–Doppler correlation matrices, statistics based on tracking-loop discriminator outputs, etc. [15–17].
- Navigation techniques, which use the navigation-domain data, such as pseudoranges and pseudorange residuals, clock and carrier-phase estimates, etc. [16].

While spoofing detection algorithms are typically concerned with identifying the presence of spoofers, additional steps need to be taken, once a spoofer presence is identified, in order to remove the impact of spoofing signals on positioning accuracy; this part is typically addressed in research works dedicated to spoofing mitigation. In terms of spoofing mitigation techniques in GNSSs, the main classes are usually as follows:

- Cryptographic techniques: These rely on cryptographic authentication to verify the integrity and authenticity of GNSS signals. This can involve digital signatures and encryption to ensure that the signals have not been tampered with. An example of cryptographic techniques in use in GNSSs nowadays is the Open Service Navigation Message Authentication (OSNMA) on Galileo receivers [18];
- The integration of GNSS with other sensors, such as Inertial Navigation Sensors (INSs) or 5G signals: When a GNSS is combined with INSs, one can increase the reliability of the positioning estimate, effectively mitigating spoofing [19]; similarly, 5G signals can be used to enhance GNSS-based positioning and mitigate the spoofing presence [20];
- Autonomous Integrity Monitoring (RAIM): Many research papers adopt methods based on Receiver Autonomous Integrity Monitoring (RAIM). For example, the authors in [21] propose a solution based on Satellites' Residual Vectors (SRV)-RAIM, based on the idea that there is some consistency between the spoofing pseudoranges and that they are spatially distributed in a different manner from genuine satellite pseudoranges; such assumption is also adopted in our paper. A sub-category of RAIM includes receiver consistency checks, based on subsets of visible satellites in the sky, a method that we adopt in our paper. The main limitation of receiver consistency checks is that they can only cope with a limited number of spoofed PRN satellites (as we will show for different scenarios), but they usually fail when all satellites in the sky are spoofed.

Overviews of various spoofing detection, localization, and mitigation algorithms can be found, for example, in [10,22,23].

3. Data Collection and Pre-Processing

To develop and test our methods, two types of GNSS datasets were collected:

1. GNSS data collected by us, in various scenarios and various countries, using Android phones and the GNSSlogger app. The GNSSlogger app captures various data from the visible satellites at each time step, including pseudoranges, atmospheric errors, and Carrier-to-Noise Ratios (CNRs), from all visible satellites in the sky. Additionally, the app records the phone position estimated by intrinsic, manufacturer-dependent algorithms based on a combination of GNSSs and the device's different sensors (e.g., accelerometer and gyroscopes), as well as cellular and Wireless Local Area Network (WLAN) or WiFi signals. These phone-based position estimates are stored in National Marine Electronics Association (NMEA) format and will represent the benchmark to

compare our results with (i.e., a sort of dynamic ground-truth concept, based on the idea that an estimate relying on integrating GNSSs with additional sensors should be more accurate than an estimate based solely on GNSS signals; this approach has been also previously used in our research, e.g., [6,24]). We used a total of six datasets, representing six types of scenarios: car ride, plane flight, sea ferry, bike, bus, and walking. These author-collected datasets include the collected GNSS data sampled at 1 Hz, comprising approximately 22,000 samples in total, which correspond to about 6 h of total recorded data. They are referred to as the walk or lake walk scenario (measurements with a Xiaomi 12 smartphone in Tampere, Finland), the bus scenario (measurements with a Google Pixel 7 smartphone in Cerdanyola del Valles, Spain), the flight scenario (measurements with a Samsung A52 smartphone on a 1 h flight from Riga, Latvia, to Tampere, Finland), the bike scenario (measurements with a Redmi Note 5G smartphone), the Hvar car scenario (measurements with a Redmi Note 5G smartphone in Hvar, Croatia), and the ferry scenario (measurements with a Redmi Note 5G smartphone, also in Croatia, on a ferry between the towns of Split and Stari Grad). These datasets have been published as open-access datasets by us on Zenono (https://zenodo.org/records/14712684 (accessed on 19 February 2025)).

2. An open-access dataset referred to as the MTV car scenario was also collected via the GNSSlogger app and having a more precise ground-truth NMEA estimate (based on differential positioning). This dataset, provided by Google and collected with a Google Pixel 4 phone, is available in Kaggle (https://www.kaggle.com/datasets/google/android-smartphones-high-accuracy-datasets (accessed on 19 February 2025)) under CC 4.0-BY license and is described in detail in [25]. It includes measurements from a car ride through the US Bay Area, near San Francisco, Mountain View, in 2020.

GNSSlogger can collect data from all the four GNSSs, namely, GPS, Galileo, Beidou, and GLONASS, and at both the L1/E1/B1 and L5/E5 carrier frequencies, depending on the Android phone models. As not all phones support E5/L5 recordings, our developments were performed by using only the L1/E1 measurements from GPS and Galileo satellites. In general, using multiple constellations translates into a higher number of visible satellites, which is highly beneficial for GNSS spoofing detection, as will be proven later.

The GNSSlogger data were then processed with MATLAB R2024b, using in-house GNSS software developed at Tampere University; the user positions are computed with an implementation of a Weighted Least Squares (WLS) algorithm [6,24] based only on the GPS and Galileo code pseudoranges (no dual frequency and no carrier-phase information were used here). It is to be noted that further algorithm improvements can be obtained by combining both code and carrier-phase measurements, as well as both L1/E1 and L5/E5 measurements, but this is a topic for future research, not addressed here.

The modeling comprised two parts: introducing a spoofing model with various range shifts and varying number of spoofed PRN codes (among the visible GPS and Galileo satellites) and developing a consistency-check spoofing detection and mitigation algorithm. Both of these steps are described in what follows.

4. List of Main Symbols and Notations

In what follows, we will adopt italics notation for scalars and boldface notation for vectors and matrices. Typically, the upper-script index $^{(i)}$ will refer to the i-th satellite. The main used symbols are summarized in Table 1; additional symbols can be encountered with their definitions in the next sections.

Table 1. Main symbols and notations in our paper.

Notation	Explanation
$\|\| \cdot \|\|$	Euclidean norm
C/N_0	Carrier-to-Noise Ratio
d	Distance
ε_{type}	Various error types, defined in the type field
G	Geometry matrix
h	Spoofing history index
L	Linking history index
lr	Historical linking ratio
n_{spf}	Number of spoofed satellites
N_{SV}	Number of visible satellites in the sky
N_{sys}	Number of considered systems (e.g., 2 for GPS + Galileo)
p	3D position vector
ρ	Satellite pseudoranges
$t_{1,2}$	Half lives for increase/decrease and decay models
W	Weighting matrix
x, y, z	x, y, and z Cartesian coordinates
ξ	Generic history index

5. Spoofing Model

A vital part of the development of our spoofing mitigation algorithms is the modeling of a spoofing model, which allows for the emulation of a spoofing attack. For this, we consider two parameters which will define in each attack scenario:

- Number of spoofed satellites: A fixed number n_{spf} of spoofed satellites is chosen, with $n_{spf} \leq min_k N_{k,SV}$, and $N_{k,SV}$ = the total number of visible satellites in the sky at each iteration (or observation interval) k. For most of our tests, the subset of n_{spf} spoofing signals was fixed for all the duration of the considered scenario, and they were selected among the visible pseudorandom (PRN) satellites in the sky. Section 9 also briefly addresses the situation where the PRN codes of the spoofers change during a scenario. Since we want to simulate a worst-case scenario, we choose as 'spoofers' the satellites which appear the most across the considered scenario, in order to maximize the presence of spoofers. A 'spoofer' PRN satellite is modeled by intentionally forcing a pseudorange error in that particular PRN satellite, computed based on a target location error defined by the user-defined parameter called position deviation and explained next.
- Position deviation: This represents the Euclidean distance between the real position and a position chosen (randomly) by the spoofer in our spoofing modeling. The spoofer will recalculate the pseudorange based on the chosen PRN satellite and the deviated receiver positions, as well as ionospheric and tropospheric errors. Thus, the spoofing signal will mimic a true PRN satellite, but with the wrong pseudoranges. The spoofed satellites are not limited to a single GNSS, meaning they can consist of either GPS, Galileo, or both satellite constellations' satellites.

The spoofers will choose a random position modeled as follows:

$$\mathbf{p}_{spf} = [x_0 + d\sin\theta, y_0 + d\cos\theta, z_0] \tag{1}$$

where the following apply:

- $\mathbf{p}_{spf}$ is the x, y, z spoofed position vector;
- $\mathbf{p}_0 = [x_0, y_0, z_0]$ is the NMEA position vector of the receiver;
- d is a fixed, user-defined position deviation introduced by the spoofed satellites;
- θ is a random x, y angle, uniformly distributed in the $[0, 2\pi]$ interval. Note: using a fixed shift ($\theta = 0$) showed insignificant differences with random θ in the results.

The pseudorange $\delta^{(i)}$ from the i-th satellite, before any spoofing, can be modeled as

$$\delta^{(i)} = r^{(i)} + \varepsilon^{(i)}_{iono} + \varepsilon^{(i)}_{tropo} + \varepsilon^{(i)}_{clock} + \varepsilon^{(i)}_{other} \tag{2}$$

where $r^{(i)}$ is the Euclidean distance between the true receiver position $\mathbf{p}_0$ and the i-th satellite position $\mathbf{p_{sat}}^{(i)} = [x^{(i)}, y^{(i)}, z^{(i)}]$; $\varepsilon^{(i)}_{iono}$, $\varepsilon^{(i)}_{tropo}$, and $\varepsilon^{(i)}_{clock}$ are the ionospheric, tropospheric, and clock errors, respectively, corresponding to the i-th satellite; and $\varepsilon^{(i)}_{other}$ are the other sources of error (e.g., multipath, hardware, background noises, and interferences).

Following this, each modeled spoofer in our model will calculate its biased pseudorange $\delta^{(i)}_{spf}$ of its corresponding i-th satellite following the equation

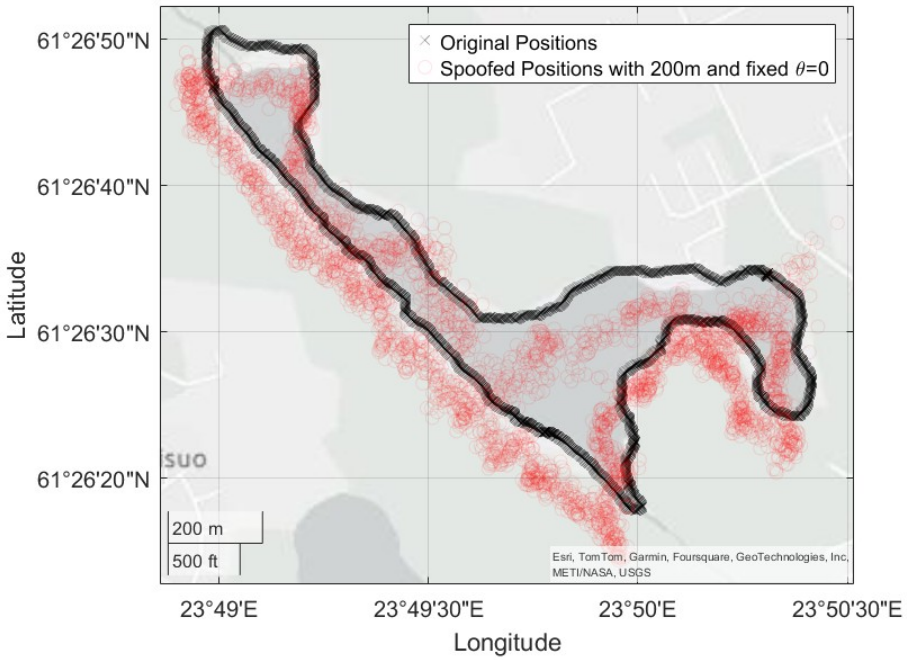
$$\delta^{(i)}_{spf} = ||\mathbf{p}^{(i)}_{sat} - \mathbf{p}_{spf}|| + \varepsilon^{(i)} \tag{3}$$

where $||\cdot||$ is the Euclidean distance, and $\mathbf{p}^{(i)}_{sat}$ and $\varepsilon^{(i)}$ are the i-th satellite position and estimated error, respectively; the estimated error $\varepsilon^{(i)}$ is computed via

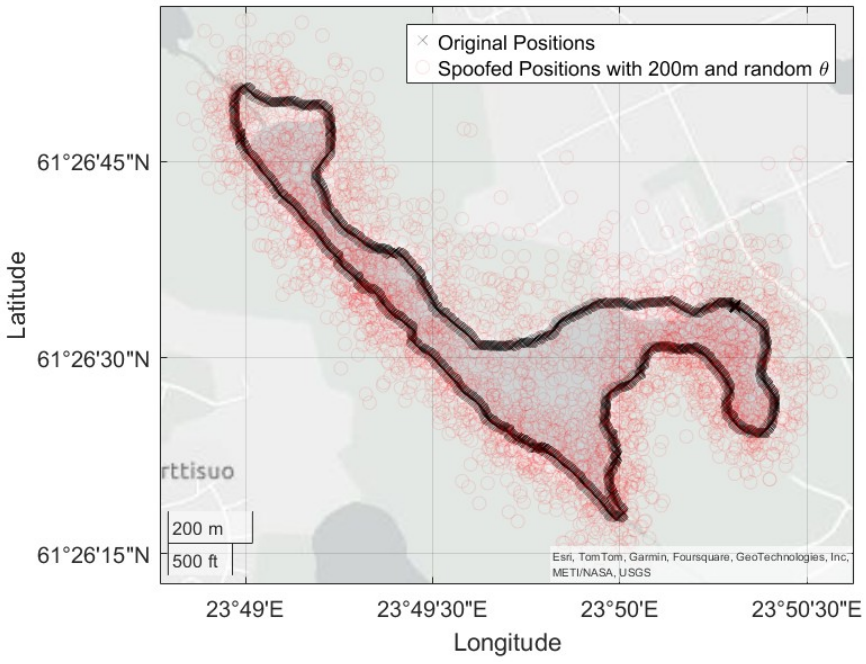
$$\varepsilon^{(i)} = \delta^{(i)} - ||\mathbf{p}^{(i)}_{sat} - \mathbf{p}_0|| \tag{4}$$

As mentioned above, the number of spoofed satellites n_{spf} will be one of the variables of our models, and it will vary from 0 to a maximum level lower than or equal to the number of visible satellites in the sky. For the purpose of spoofing modeling, we took $\mathbf{p}_0$ as the NMEA receiver position calculated by the Android phones. This kind of spoofing model matches in fact a targeted spoofing attack, where spoofing signals mimic a true satellite; thus, the spoofed PRN code will have similar biases and errors to a genuine PRN code, making this model match a sophisticated spoofing attack. One limitation of our model is that we assume that the PRN indices of the spoofed satellites are distinct from the PRN indices of the genuine satellites, in order to be able to better study the impact of increasing the number of spoofing signals.

For illustrative purposes, Figure 1 shows two examples of the generated spoofing positions following the above-mentioned methodology, for a fixed and a random θ. Our model can be expanded in a straightforward manner to situations where the spoofed satellites have different clock errors than the genuine ones by simply adding another summation term in Equation (4) to model the clock bases between the genuine and spoofed satellites; however, in all our tests, we used the assumption of equal clock errors in the spoofed and genuine pseudoranges, which mimic a worst-case spoofing situation.



(**a**) Fixed θ



(**b**) Random θ

Figure 1. Examples of generated spoofing positions with fixed θ (**a**) and uniformly distributed θ (**b**); in this example, the position deviation $d = 200$ m; lake walk scenario.

6. Proposed Consistency-Check-Based Spoofing Mitigation Algorithm

6.1. General Algorithm

The PVT solution is computed via the well-known WLS algorithm, described, for example, in [6,26–28], and based on information received by the different satellites in view, as well as some other known GNSS data (such as satellite positions extracted from broadcast ephemeris data, ionospheric and tropospheric error models, etc.). The ionospheric

corrections are based on Klobuchar ionospheric models, and the tropospheric model is based on the dry/hydrostatic Saastamoinen model [28]. The WLS solution [26,27] is basically an iterative approach to estimating the increments Δx_j Δy_j Δz_j of the estimated receiver position at the j-th iteration, as well as the $c\Delta t_j$ clock increments (with c being the light speed and Δt_j being the estimated clock bias between the receiver and the satellite system of the reference (e.g., GPS) at j-th iteration). Then, the WLS formula to estimate the increment vector $\Delta \mathbf{x}_j$, with $\Delta \mathbf{x}_j = [\Delta x_j \ \Delta y_j \ \Delta z_j \ \Delta t_j]^T$ and ^{T} being the transpose operator, is

$$\Delta \mathbf{x}_j = (\mathbf{G_j^T W G_j})^{-1} \mathbf{G_j^T W y_j} \tag{5}$$

where $\mathbf{G}_j$ is the geometry matrix at the j-th iteration and $\mathbf{W}$ is a weighting matrix, explained in detail after Equation (6); in the case of two systems of interest (GPS and Galileo) with the first $n1$ satellites belonging to the GPS constellation and the remaining $N_{SV} - n1$ satellites belonging to the Galileo constellation, the geometry matrix $\mathbf{G}_j$ at the j-th iteration is given by

$$\mathbf{G}_j = \begin{bmatrix} \frac{x^{(1)} - \hat{x}_{j-1}}{||\mathbf{p}_{sat}^{(1)} - \hat{\mathbf{p}}_{j-1}||} & \frac{y^{(1)} - \hat{y}_{j-1}}{||\mathbf{p}_{sat}^{(1)} - \hat{\mathbf{p}}_{j-1}||} & \frac{z^{(1)} - \hat{z}_{j-1}}{||\mathbf{p}_{sat}^{(1)} - \hat{\mathbf{p}}_{j-1}||} & 1 & 0 \\ \frac{x^{(2)} - \hat{x}_{j-1}}{||\mathbf{p}_{sat}^{(2)} - \hat{\mathbf{p}}_{j-1}||} & \frac{y^{(2)} - \hat{y}_{j-1}}{||\mathbf{p}_{sat}^{(2)} - \hat{\mathbf{p}}_{j-1}||} & \frac{z^{(2)} - \hat{z}_{j-1}}{||\mathbf{p}_{sat}^{(2)} - \hat{\mathbf{p}}_{j-1}||} & 1 & 0 \\ \cdots & & & & \\ \frac{x^{(n1)} - \hat{x}_{j-1}}{||\mathbf{p}_{sat}^{(n1)} - \hat{\mathbf{p}}_{j-1}||} & \frac{y^{(n1)} - \hat{y}_{j-1}}{||\mathbf{p}_{sat}^{(n1)} - \hat{\mathbf{p}}_{j-1}||} & \frac{z^{(n1)} - \hat{z}_{j-1}}{||\mathbf{p}_{sat}^{(n1)} - \hat{\mathbf{p}}_{j-1}||} & 1 & 0 \\ \frac{x^{(n1+1)} - \hat{x}_{j-1}}{||\mathbf{p}_{sat}^{(n1+1)} - \hat{\mathbf{p}}_{j-1}||} & \frac{y^{(n1+1)} - \hat{y}_{j-1}}{||\mathbf{p}_{sat}^{(n1+1)} - \hat{\mathbf{p}}_{j-1}||} & \frac{z^{(n1+1)} - \hat{z}_{j-1}}{||\mathbf{p}_{sat}^{(n1+1)} - \hat{\mathbf{p}}_{j-1}||} & 0 & 1 \\ \cdots & & & & \\ \frac{x^{(N_{SV})} - \hat{x}_{j-1}}{||\mathbf{p}_{sat}^{(N_{SV})} - \hat{\mathbf{p}}_{j-1}||} & \frac{y^{(N_{SV})} - \hat{y}_{j-1}}{||\mathbf{p}_{sat}^{(N_{SV})} - \hat{\mathbf{p}}_{j-1}||} & \frac{z^{(N_{SV})} - \hat{z}_{j-1}}{||\hat{\mathbf{p}}_{sat}^{(N_{SV})} - \mathbf{p}_{j-1}||} & 0 & 1 \end{bmatrix} \tag{6}$$

where the upper indices are associated with the ECEF positions of a satellite (e.g., $\mathbf{p}_{sat}^{(n1)} = [x^{(n1)}, y^{(n1)}, y^{(n1)}]$ is the ECEF 3D position of the $n1$-th satellite) and the lower indices are associated with the receiver position estimate at a certain iteration (e.g., $\hat{\mathbf{p}}_{j-1} = [\hat{x}_{j-1}, \hat{y}_{j-1}, \hat{z}_{j-1}]$ is the estimated 3D receiver position in ECEF coordinates at iteration $j-1$). The starting point in our WLS solution was $\hat{\mathbf{p}}_0 = [0, 0, 0]$, and we used a maximum of 500 iterations in the WLS algorithm for each estimated receiver track point, with the exit condition that either the maximum number of iterations was reached or the Euclidean difference between the current WLS estimate and the previous WLS estimate was smaller than 10^{-3}. For clarity, the receiver track point index k was dropped from the above models; however, if a prior track point position is known, the algorithm is initialized to the previous track position instead of the $[0, 0, 0]$ position.

The weighting matrix $\mathbf{W}$ in Equation (5) is given by the diagonal matrix $diag\left(1/w_\sigma^{(1)}, 1/w_\sigma^{(2)}, \ldots, 1/w_\sigma^{(N_{SV})}\right)$, where $w_\sigma^{(i)}$ is a weighted noise variance factor for the i-th visible satellite, computed via $w_\sigma^{(i)} = a + b \cdot 10^{-SNR_{dB}^{(i)}/10}$, with $SNR_{dB}^{(i)}$ being the signal-to-noise ratio (in dB) of the i-th satellite, $a = 10$, and $b = 150^2$. The pre-fit residual at step j, namely, the $\mathbf{y_j}$ factor in Equation (5), is given by the measured pseudoranges minus the distance between the visible satellites and the estimated receiver position at the previous iteration and minus the estimated ionospheric and tropospheric corrections (performed based on the well-known Klobuchar and Saastamoinen models [26,27], respectively):

$$\mathbf{y_j} = \begin{bmatrix} \delta^{(1)} - ||\mathbf{p}_{sat}^{(1)} - \hat{\mathbf{p}}_{j-1}|| - \hat{\varepsilon}_{tropo}^{(1)} - \hat{\varepsilon}_{iono}^{(1)} \\ \delta^{(2)} - ||\mathbf{p}_{sat}^{(2)} - \hat{\mathbf{p}}_{j-1}|| - \hat{\varepsilon}_{tropo}^{(2)} - \hat{\varepsilon}_{iono}^{(2)} \\ \cdots \\ \delta^{(n1)} - ||\mathbf{p}_{sat}^{(n1)} - \hat{\mathbf{p}}_{j-1}|| - \hat{\varepsilon}_{tropo}^{(n1)} - \hat{\varepsilon}_{iono}^{(n1)} \\ \cdots \\ \delta^{(N_{SV})} - ||\mathbf{p}_{sat}^{(N_{SV})} - \hat{\mathbf{p}}_{j-1}|| - \hat{\varepsilon}_{tropo}^{(N_{SV})} - \hat{\varepsilon}_{iono}^{(N_{SV})} \end{bmatrix} \quad (7)$$

If any of the pseudoranges received by the satellites is spoofed, the position calculated by the receiver will be erroneous; thus, the objective of our algorithm will be detecting and ruling out the faulty or spoofed satellites. In order to identify whether a satellite is spoofed or not, we developed a consistency-check-based method that estimates the so-called Candidate Reliability Probability (CRP), which will be described later, in Section 6.6. The CRP will be computed first at the level of a candidate solution (i.e., a subset of the visible satellites in the sky) and then at the level of each satellite in that candidate set. The objective of the algorithm is to identify the largest number of candidate sets that are strongly linked to each other; this way, the rule of majority will decide which candidate sets are reliable and which are not; this is performed through an iterative process of evaluating subsets of satellites (candidate sets), and the next subsection details the procedure.

With respect to the impact of the weighting matrix on result accuracy, we would like to highlight two aspects: First, in order check the contribution to accuracy that the weights on WLS provide, we first calculated the average 3D Euclidean distance between the NMEA positions and the positions obtained with the normal WLS implementation; a second calculation was performed again by averaging the 3D Euclidean distance between the NMEA positions and the positions obtained with the Least Squares (LS) implementation, where the weighting matrix is replaced with the unit matrix; the difference between WLS and LS 3D varied between scenarios, with WLS consistently providing better results than LS, with positioning error differences ranging from ≈0.1 m to ≈15 m. Second, the choice of the weights in the weighting matrix has already been optimized in [29], and we have chosen the optimal weights according to the research performed in [29] and references therein.

6.2. Spoofing and Linking History Indexes

Although we try to make as few assumptions on the spoofing attack as possible, we are going to assume that as happens in the simulated spoofing attack, the spoofers are not going to change throughout a scenario. This means that we can use information from previous predictions to make a faster and more precise prediction in the current prediction. This information is aggregated in a unique HRS index per satellite, $HRS^{(i)}$, which is explained later in Section 6.4, and it relies on two history indices, as follows:

- Spoofing history index $h_k^{(i)}$: This will assign an index to each satellite i at each iteration step $k = 0, 1, 2, \ldots$ after identifying the genuine and spoofed satellites which will undergo (following Section 6.4):
  - Index increase if the satellite is identified as genuine;
  - Index decrease if the satellite is identified as a spoofer;
  - Index boosting if the satellite is not visible—this situation is similar to the index increase, but with a lower increase due to uncertainty.

  The history indexes represent a "probability" of being spoofed. In the absence of additional information, the initial indices $h_0^{(i)}$ are set to 0.8.

- Linking history index $L_k^{(i,j)}$: This is a symmetrical matrix that defines the relationship between two satellites i, j at each iteration step $k = 0, 1, \ldots$. An iteration step corresponds to a measurement point in the track; in our data, this step is 1 s, equal to the distance between any two successive measurements. Each index will undergo one of the following:
  - Index increase if both satellites i and j are either found to be genuine or spoofers at the same prediction step;
  - Index decrease if only one of the satellites is found to be genuine, while the other is found to be faulty;
  - Index boosting if at least one of the two satellites is not visible.

  Similarly to a correlation matrix, the indexes represent a "probability" of being linked together and have values between -1 (i.e., if one of the two satellite is a spoofer, then the other one is genuine, or vice versa) and 1 (i.e., the two satellites are either both spoofed satellites or both genuine satellites). Small absolute values of $L_k^{(i,j)}$ mean that there is no relationship between satellite i and satellite j at step k. In the absence of additional information, the initial values are also set to 0: $L_0^{(i,j)} = 0; \forall i, j$.

The increase and decrease in and the boosting of the spoofing and linking history indexes will be iteratively defined as follows:

- Index increase:

$$\xi_{k+1} = 1 - (1 - \xi_k)0.5^{\frac{1}{t_1}} \tag{8}$$

- Index decrease:

$$\xi_{k+1} = \xi_k 0.5^{\frac{1}{t_1}} \tag{9}$$

- Index Boosting;

$$\xi_{k+1} = 1 - (1 - \xi_k)0.5^{\frac{1}{t_2}} \tag{10}$$

where ξ_{k+1} is the considered index (i.e., $h_k^{(i)}$ or $L_k^{(i,j)}$) at the $(k+1)$-th iteration, ξ_k is the considered index at the k-th iteration, t_1 is the chosen half life for the increase/decrease (a half life proportional to the CRP was used: $t_1 = -10 \log_2 \text{CRP}$), and t_2 is the chosen half life for the decay case (for our experiments, we used $t_2 = 60$).

An illustration of index increase, decrease, and boosting is provided in Figure 2.

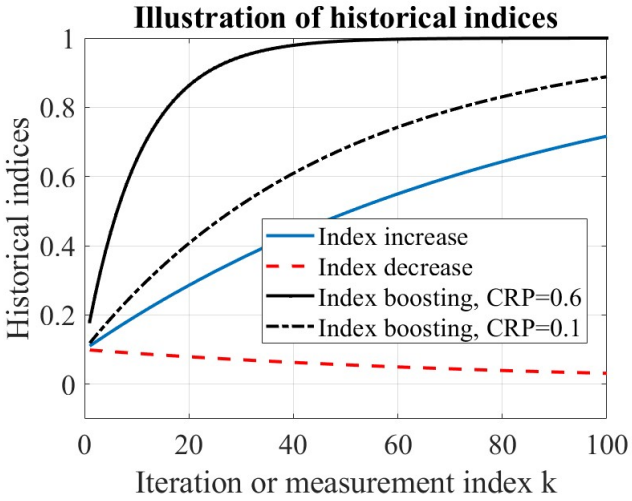


Figure 2. An illustration of the rules for index increase, decrease, and boosting, assuming a starting index probability of 0.1.

The above-mentioned indices are computed for all N_t possible satellites (visible or not) in the considered constellations (e.g., if only the GPS and Galileo constellations are used, $N_t = 61$ satellites), because some satellites can appear and disappear from the sky, and this dynamic is also important in spoofing detection and mitigation. The total number of visible satellites in the sky at each step is denoted by $N_{k,SV}$. Out of the visible satellites, we have n_{spf} spoofed PRN codes and $N_{k,SV} - n_{spf}$ genuine PRN codes at each k-th step.

6.3. Initial Evaluations

Assuming that the number of genuine satellites is larger than the number of spoofed satellites ($N_{k,SV} - n_{spf} > n_{spf}$) and assuming that the previous predictions are correct predictions, we can expect to find in the linking history a group of positive linked satellites, which are negatively linked to a smaller set of satellites. From this assumption, we can build two initial separate groups, with the larger being presumably the true set and the smaller being the spoofed set. Ideally, this will allow the algorithm to make an initial evaluation from the subset of genuine sets, leaving out the spoofed satellites.

6.4. Iterative Search

If a valid candidate set is not found in the initial evaluations, we start an iterative phase, where we generate subsets of candidates from the visible satellites and evaluate each one until one is found to be valid.

This iterative phase will generate candidates from a smaller to a larger number of satellites per candidate set, i.e., from $m = 5$ to $m = N_{k,SV}$, at each k-th step. A valid subset is one for which the CRP is above a threshold (i.e., 0.05 threshold for our experiments). By starting the search with the smallest possible subset, we will increase the probability of finding a valid subset (either of all spoofed or of all genuine satellites) faster, which will then allow us to subsequently classify the rest of the satellites. This has a small risk, since the CRP is less accurate for smaller sets of satellites, but the reduction in average detection speed is lowered significantly. For each size m of subsets, all the possible combinations of m satellites are generated. Ordering the candidates adequately is key to making predictions as fast as possible, since it allows us to evaluate first the combinations with the highest reliability. Thus, we define a metric based on the spoofing and linking history, the historical reliability score, which can be calculated for every i-th satellite at each step k as

$$\mathrm{HRS}_{k+1}^{(i)} = \sqrt{\frac{h_k^{(i)}}{1 + \exp\left(-25(lr_k^{(i)} - 0.5)\right)}} \tag{11}$$

where $h_k^{(i)}$ is the spoofing history index defined in Section 6.2 and $lr_k^{(i)}$ is the historical linking ratio up to the k-th step of all positively related satellites (out of N_t possible ones) with the i-th satellite:

$$lr_{k+1}^{(i)} = \frac{0.5 + \sum_{j=1}^{N_t} \max(L_k^{(i,j)}, 0)}{0.5 + \sum_{j=1}^{N_t} L_k^{(i,j)}} \tag{12}$$

where $L_k^{(i,j)}$ is the linking history index defined in Section 6.2.

At each k-th step, a satellite with a value of $HRS_k^{(i)}$ below 0.5 is considered unreliable, while 0 and 1 are the minimum and maximum historical reliability values, respectively.

After the iterative search, if we have not found a valid candidate, we evaluate again the candidate that resulted in the highest CRP value. If the CRP is above a fraction of the threshold (e.g., 0.05) and the average HRS of the satellites is above 0.5, we accept the candidate; if not, we will consider all satellites to be spoofed (see the flowchart in Figure 3 and the pseudocode given in Algorithm 1).

Algorithm 1 General algorithm

```
Start
if N_SV < min_satellites then
    All satellites with HRS > 0.5 are compromised
    Stop
else
    Build initial list of candidates
    while For each candidate do
        Calculate CRP (Following Figure 4)
        if CRP > threshold then
            Evaluate remaining satellites and update history
            Stop
        end if
    end while
    Iterative candidate search
    for m from 5 to N_SV do
        Build list of candidates of size m and order based on history
        while For each candidate do
            Calculate CRP (Following Figure 4)
            if CRP > threshold then
                Evaluate remaining satellites and update history
                Stop
            else
                Continue
            end if
        end while
    end for
    No valid candidates found
    Select candidate with highest CRP
    if (CRP > threshold/5) and (HRS > 0.5) then
        Settle for candidate
    else
        Determine that all satellites are compromised
    end if
end if
Satellites Identified
Stop
```

If a valid candidate has been found, we proceed to evaluate the remaining satellites. This is performed to assert all visible satellites, as this classification of satellites is later reflected on the history index. Similarly to the candidate evaluations, the evaluations of individual remaining satellites against the contender solution is performed through the same CRP estimation. After this step, satellites with a CRP above the threshold are added to the final subset of true satellites, while the reset are considered spoofed.

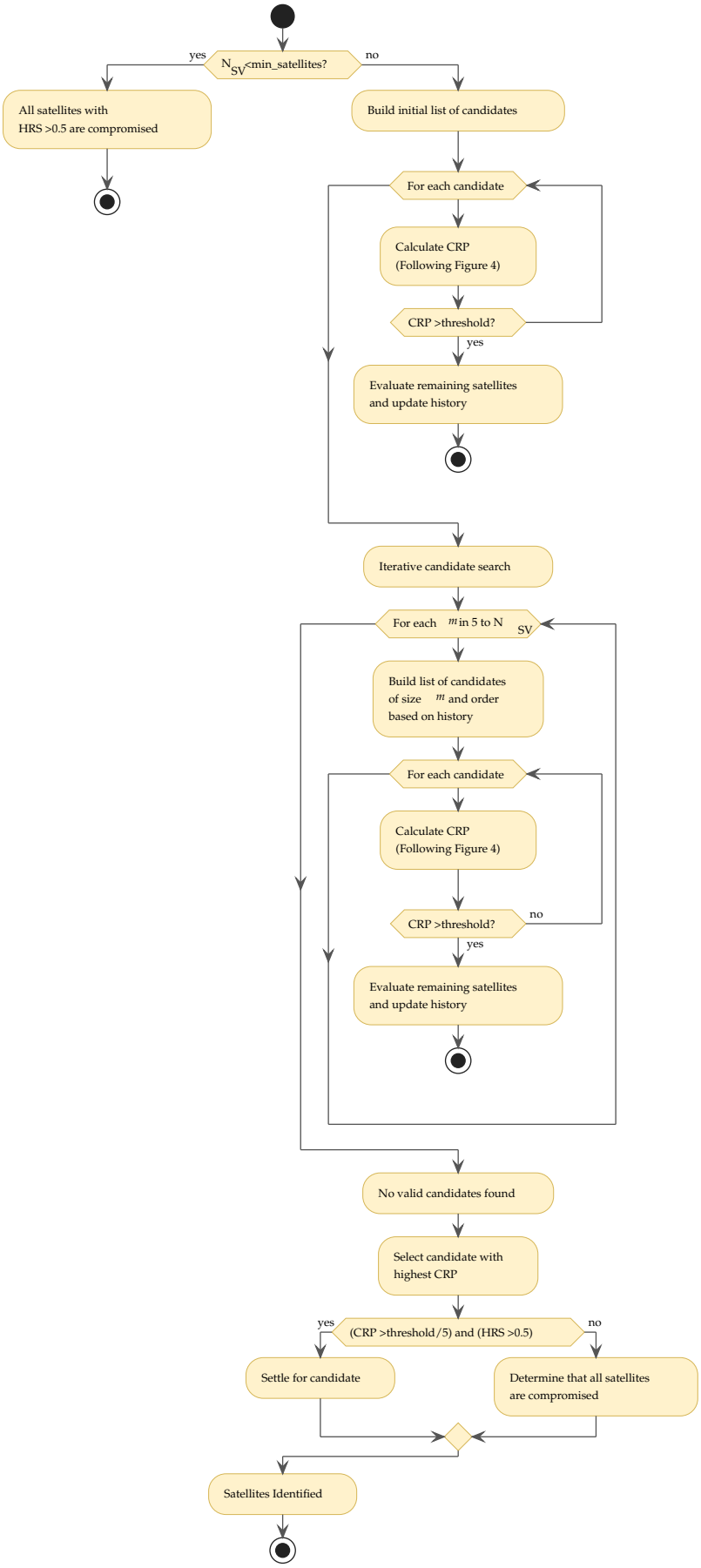


Figure 3. Flowchart of the proposed algorithm. The experiments were executed with a threshold of 0.05, and the CRP calculation is detailed in Figure 4.

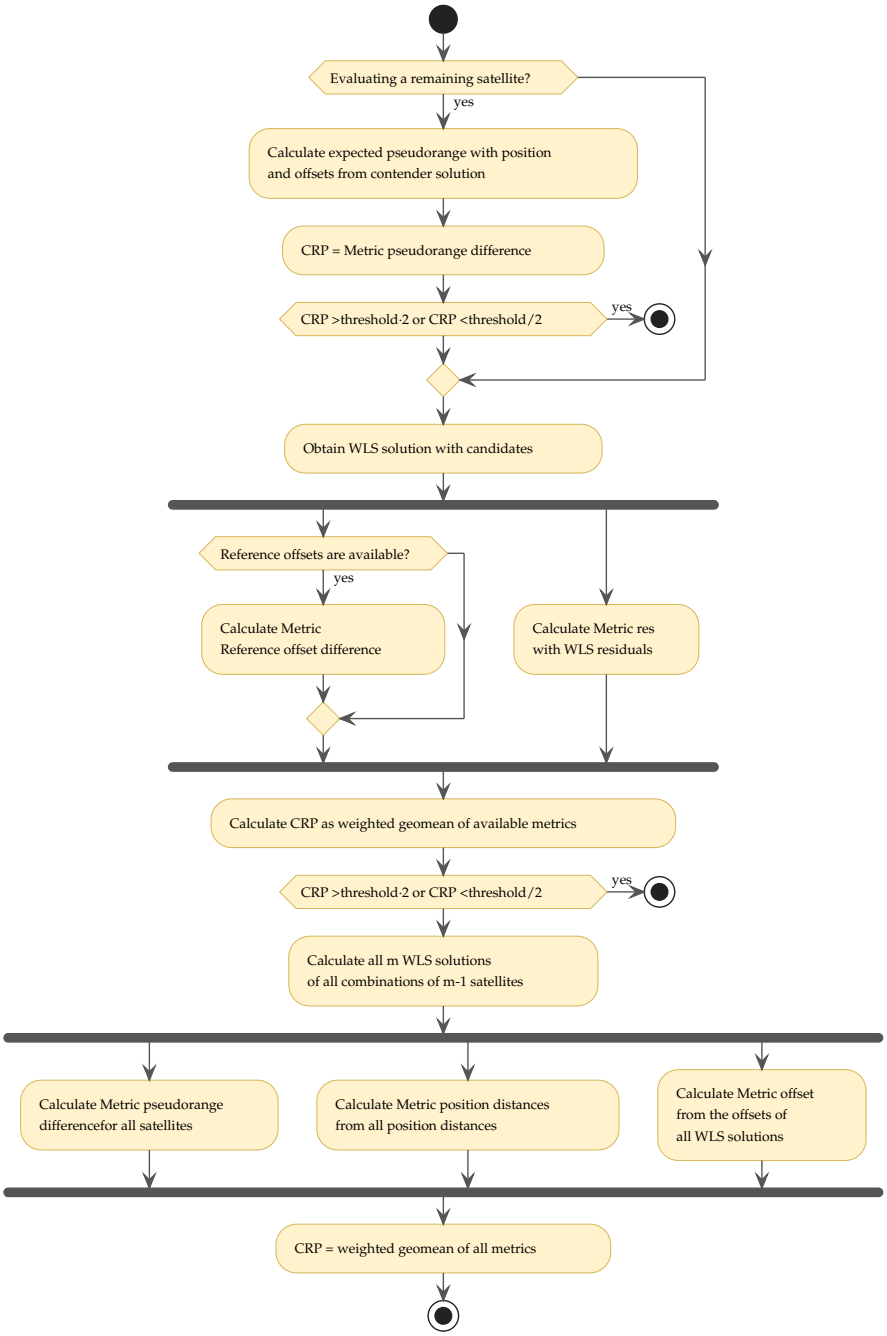


Figure 4. Flowchart of the CRP calculation, which details the evaluation of a candidate; CRP calculation was one of the blocks in Figure 3.

6.5. Multiple-Solution Case

In the case where the number of spoofed satellites is large enough to generate a valid WLS solution (i.e., strictly higher than 4 for only one considered system, strictly higher than 5 if we consider both GPS and Galileo satellites, etc.,) we are at a risk of considering a subset containing spoofers to be valid, given that this would likely result in a CRP below the threshold. In order to avoid this, a final step is performed, checking if the number of spoofed satellites is large enough to generate a valid WLS solution (i.e., if the number of spoofed satellites > 5 in our case when we consider both GPS and Galileo satellites). In the positive case, the prediction algorithm is re-run with the spoofed satellites. Finding that this re-run of the algorithm results in a valid solution means that we have two different possible valid solutions, one of which is the true solution, while the other is the spoofed subset. In this situation, we look at the average ratio $lr_k^{(i)}$ of the i-th satellite at the k-th step

and consider the set of satellites with the highest average ratio to be the true or genuine set. This means that even if at a certain time step the number of spoofed satellites is larger than the number of true satellites, we can still find the true set of satellites. Finally, after finding a definitive set of true satellites and false satellites, we update the linking and spoofing history according to Equations (8)–(10). Figure 3 gives the flowchart of the full algorithm, while the next section details the CRP calculation, and Figure 4 describes the flowchart of CRP calculation.

6.6. CRP Calculation Based on Candidate Subsets

At each iteration k (recall that an iteration corresponds to one measurement index), we start by forming all possible candidates of m satellites among $N_{k,SV}$ visible satellites for the selected N_{sys} systems of interest (for example, if we focus only on GPS and Galileo, we have $N_{sys} = 2$ systems of interest). With m subsets out of $N_{k,SV}$ visible satellites, we have $N_{cand} = \frac{N_{k,SV}!}{m!(N_{k,SV}-m)!}$ possible combinations (called candidates), where ! stands for the factorial operator.

With each of these candidates, we form a WLS estimate (see Equations (5) and (6)) of the position and clock offset corresponding to the n_c-th candidate based on the m pseudoranges taken out all of the $N_{k,SV}$ available pseudoranges; $n_c = 1, \ldots, N_{cand}$.

The final WLS estimate will contain an estimate of the receiver position $(\hat{x}, \hat{y}, \hat{z})$ and of the receiver–satellite clock offset (Δ_t) based on each candidate set out of the N_{cand} sets. After these WLS estimates are computed for each candidate set, we will recompute the pseudorange residuals $\mathbf{y} - c\Delta t$ based on Equation (7) and the receiver position and clock WLS estimates based on each candidate subset.

We can make the following observations:

- A minimum of $3 + N_{sys}$ pseudoranges are needed to form WLS N_{sys} systems; additionally, if the number of visible satellites minus the number of different GNSSs N_{sys} is equal to 3, we will have a unique solution, as there are as many equations as parameters, resulting in residuals ≈ 0.
- Since the residuals represent the fitting quality of the solution, lower residuals will translate into a more precise solution (except for the previously mentioned case).

As mentioned above, the residuals $\mathbf{y} - c\Delta t$ should be low if the candidate set is reliable (i.e., there are no spoofers in it). Additionally, if all the m satellites in a candidate subset are genuine, we can also expect all subsets of $m - 1$ satellites among the m ones to be genuine and to give similar residuals to the solution with all m satellites. Based on this, we will study and model the following variables given a set of m satellites:

1. Variable 1 = Position distances: We compare the distances between the position estimate with m satellites and the position estimates based on all subsets of $m - 1$ satellites among the m ones, and we study the distribution of these distances. The distribution of distances varies depending on the value of m. As it can be expected, on one hand, with higher m values, we find smaller distances, as the position solution is more precise and consistent; on the other hand, lower m values have less redundancy which leads to lower precision of the positions and thus a higher distance between them.
2. Variable 2 = Pseudorange difference: Instead of studying the differences between the final receiver position estimates, we can also focus on the residuals of pseudoranges; these are taken as the differences between the recalculated pseudoranges for each satellite, where the recalculated pseudoranges use the position estimates based on m satellites, as well as based on subsets of $m - 1$ satellites, chosen among the m ones.
3. Variable 3 = Clock offset difference: The WLS solution estimates both a receiver position and a clock offset; each of the $m - 1$ subsets considered among the m satellites will give a different set of clock offset estimates per satellite. As mentioned previously,

we expect all the clock offsets to have similar values in the case where all satellites are genuine (or if all are synchronized spoofers) and to have different values if the set of $m-1$ satellites contains combinations of spoofing and genuine PRN codes. In the case of two systems (GPS and Galileo), for each of the different clock offsets $\Delta_{t_{GPS}}, \Delta_{t_{Galileo}}$, we calculate the deviations from the mean of each clock offset solution.

The distributions of all three variables can be approximated to log-normal distributions which depend on the value of the candidate size m. The log-normal distribution is defined by the mean and standard deviation of the data logarithmically. Since the fitting is not exact and we are interested in correctly modeling the tails, instead of defining the standard deviation in the traditional way, we define it in terms of the 95th percentile:

$$\sigma = \frac{\ln Q_{95} - \ln \mu}{\sqrt{2} \cdot \text{erf}^{-1}(2 \cdot 0.95 - 1)} \tag{13}$$

Given a candidate set of m satellites, we consider the following metrics using the estimated distributions in order to build a CRP value based on a weighted geometrical mean of all these metrics:

1. Position distances (following the distribution of the above-mentioned Variable 1).
   This results in a list of $\frac{(m-1)!}{2!(m-3)!}$ distances, corresponding to distances between any two satellites picked from $m-1$ satellites; again, ! is the factorial.
2. Pseudorange differences (following the distribution of Variable 2).
   This results in m values, one for each satellite in the candidate subset.
3. Clock offset differences (following the distribution of Variable 3).
   This results in $m \cdot N_{sys}$, corresponding to the one or two values for each satellite in the candidate subset, depending on the candidate including one ($N_{sys} = 1$) or two GNSSs ($N_{sys} = 2$).
4. Residuals obtained based on the WLS solution formed with m satellites (following the distribution of Variable 2).
   This results in m values, one for each satellite in the candidate subset.
5. Difference between the offset obtained in the WLS solution using all m satellites and a reference true clock offset (if applicable, following the distribution of Variable 3).
   This results in one or two values, depending on the candidate including one or two GNSSs.

This last metric evaluates the consistency of the clock offsets. The clock offsets ($\Delta_{t_{GPS}}, \Delta_{t_{Galileo}}$) of the receiver should be almost constant between correct solutions, since this will only change if the receiver or satellites adjusts their internal clocks. Accordingly, we can estimate the true clock offsets as the median of the last n iterations or measurements (in the experiments, we use $n = 100$).

For each metric, we obtain a list of values, which are translated into lists of estimated probabilities, by calculating the complementary cumulative distribution functions of the values against the corresponding distributions. To measure the quality of each of the metrics, a value is calculated as the geometric mean of each of the corresponding lists of probabilities.

After some experimental analysis, the empirical weights shown in Table 2 were assigned to each of the metrics based on the precision they provide. It is important to note that the metrics can be divided into the different numbers of position calculations required, which can be either none, 1 (corresponding to the one calculated with the m satellites), or m (one for each of the possible subsets of $m-1$ satellites taken among the m ones).

Table 2. The metrics used in CRP calculation and their empirically assigned weights.

Metric	Weight	Number of WLS Calls
Position distances	5	m (based on the number of candidates)
Pseudorange differences	5	m (no reference position available) or 0 (reference position available)
Clock offset differences	1	1
Residuals	1	1
Reference offset difference	1	1

For the pseudorange difference, if the CRP is calculated for a remaining satellite (having already obtained a true position), we can just calculate the pseudorange difference with the reference position, thus needing no additional WLS solutions (i.e., the 0 value in Table 2).

The final CRP is computed as the weighted geometric mean of all metrics obtained up to that point. We recall that a weighted geometric mean of L variables denoted by $v_l, l = 1, \ldots, L$ and having weights w_i is given by

$$\text{CRP} = \exp\left(\frac{\sum_{l=1}^{L} w_l \ln(v_l)}{\sum_{l=1}^{L} w_l}\right) \tag{14}$$

where variables v_l and weights w_l are taken from Table 2) and $L = 5$ in our algorithm. Figure 4 gives the flowchart of CRP calculation, and Algorithm 2 shows its corresponding pseudocode.

Algorithm 2 Candidate evaluation

Start
if Evaluating a remaining satellite **then**
    Calculate expected pseudorange with position and offsets from contender solution
    CRP = Metric pseudorange difference
    if CRP $>$ threshold $\cdot$ 2 or CRP $<$ threshold/2 **then**
        Stop
    end if
end if
Obtain WLS solution with candidates
if Reference offsets are available **then**
    Calculate Metric Reference offset difference
end if
Calculate Metric res with WLS residuals
Calculate CRP as weighted geomean of available metrics
if CRP $>$ threshold $\cdot$ 2 or CRP $<$ threshold/2 **then**
    Stop
end if
Calculate all m WLS solutions of all combinations of $m - 1$ satellites
Calculate Metric pseudorange difference for all satellites
Calculate Metric position distances from all position distances
Calculate Metric offset from the offsets of all WLS solutions
CRP = weighted geometrical mean of all metrics
Stop

7. Scenario-by-Scenario Results with Android Data

After developing the spoofing detection and mitigation algorithm, we proceed to analyze its performance throughout a multitude of tests, based on Android data collected in various dynamic scenarios at low speed (walking), moderate speed (biking and riding

the bus), and high speed (traveling by plane). The main reason for selecting these scenarios was to be able to analyze environments which are different enough in terms of signal propagation (e.g., rural/urban and lake/road) and speed (walk, car, plane, etc.). In each test, we choose a dataset, a fixed number of satellites, and the position deviation. The spoofed satellites can be either GPS, GALILEO, or a permutation of both types.

7.1. Walk Scenario

For our first test, we focus on a scenario where the person carrying the receiver walks a closed-loop route along a lake in a forest located in Tampere, Finland. The whole route is about 4.5 km. This scenario includes of a considerably large average of satellites in view per time step (16.98 satellites in view/time step).

In Figure 5, we show the average 3D positioning error (in ECEF coordinates), before and after applying the spoofing correction algorithm and for one, two, four, and eight spoofed PRN codes. The minimum error indicated in Figure 5 is the Spoof-Free Error, defined as the Euclidean distance between the 'ground-truth' NMEA positions provided by the receiver and a WLS-based position estimate based on only the non-spoofed GPS and Galileo satellites (e.g., in the case of 2 spoofers and 17 visible PRN codes, it means that only 15 satellites were genuine ones, and the minimum error was obtained over those 15 satellites).

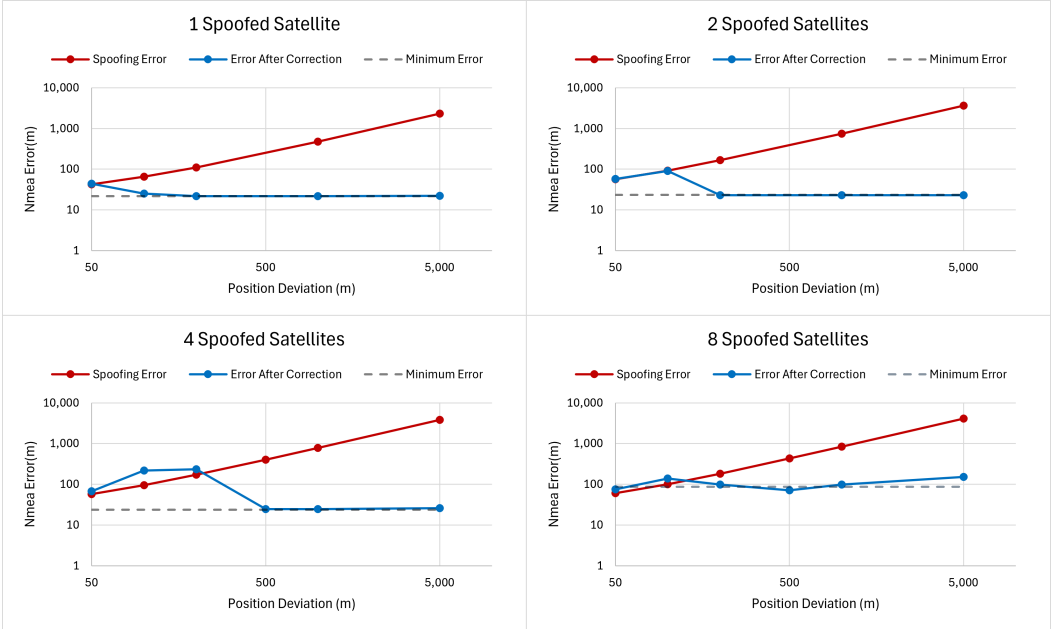


Figure 5. Three-dimensional positioning errors before and after correction in the walk scenario; the 'minimum error' bound or the 'Spoof-Free Error' is explained in the text. The 'spoofing error's is the error due to spoofing, if no correction (or mitigation) is applied.

As can be appreciated, with one, two and four spoofed satellites, the highest error after spoofing correction is obtained with the lowest position deviations. This shows that with a relatively low number of spoofed satellites, detection will work better the larger the position deviation is. The eight-spoofed satellites case shows that the algorithm can still correct (to some extent) the positions even if the number of spoofed satellites is larger than that required to obtain a valid WLS position.

Table 3 shows the error after correction (in this case, the range deviation introduced by the spoofer was assumed to be 5000 m), as well as the correction rates and the spoofing detection accuracy. We also show the correction rate as the total correction respect to the total possible:

$$\text{Correction Rate} = \frac{\text{Error Before Correction} - \text{Error After Correction}}{\text{Error Before Correction} - \text{Spoof-Free Error}} \quad (15)$$

Above, the Spoof-Free Error has the following meaning: if we were to discard all spoofing signals and compute a WLS-based position estimate with only the genuine satellites, after the spoofed ones are removed, then this Spoof-Free Error would be the distance between this position estimate and the NMEA position that we use as 'ground truth'. We can also remark that since the number of remaining genuine satellites decreases when the number of spoofed satellites increases, the 'Spoof-Free Error' can become slightly bigger than the 'error after correction', and this explains the situations when the correction rate is slightly above 1. In any case, a correction rate close to 1 means that the spoofing correction algorithm works well, as a positive but lower-than-one correction rate means that spoofing correction is able to correct only parts of the spoofing errors and a negative correction rate means that the correction algorithm has failed.

Spoofing detection accuracy has been defined as

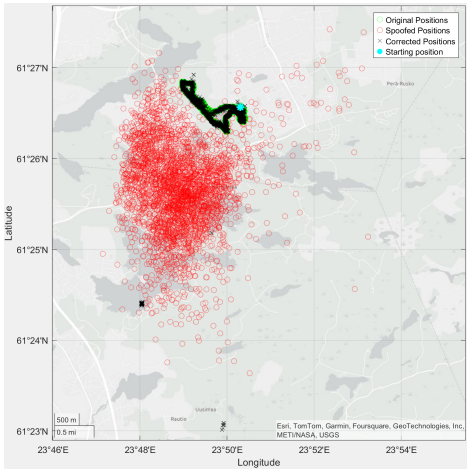
$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Population}} \quad (16)$$

Table 3. Three-dimensional positioning error in ECEF coordinates, before and after correction in the walk scenario; the walk scenario had an average of $\mathbf{E}_m(N_{m,SV}) = 16.98$ visible satellites, with $m = 1, 2, \ldots$ being the measurement index and $\mathbf{E}(\cdot)$ being the expectation operator.

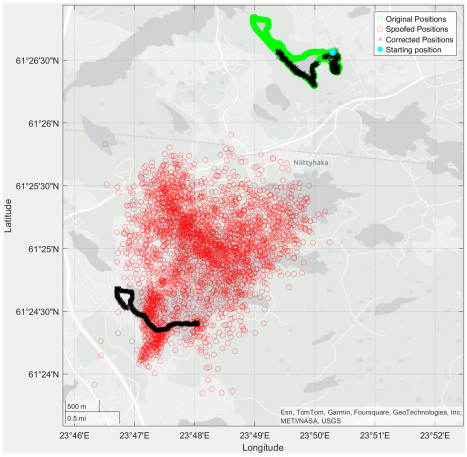
Number of Spoofed Satellites out of $N_{m,SV}$	**Error Before Correction (m)**	**Error After Correction (m)**	**Correction Rate (-)**	**Detection Accuracy (%)**
0	20.77	20.55	-	98.97
1	2307.15	22.28	0.9999	97.95
2	3667.62	23.03	1	98.3
3	3391.35	23.37	1.0001	99
4	3836.23	25.76	0.9997	99.32
5	3823	34.75	0.9811	98.38
6	3999.28	48.65	1.0013	98
7	4295.66	87.03	0.9896	97.58
8	4119.34	152.47	1.0097	97.6
9	3356.92	256.7	0.9329	87.21
10	3798.18	2501.41	0.3515	43.97
11	3799.92	5009.05	−0.4798	0.03

In the walk scenario, we can see how up to 9 spoofed satellites (our of the 16.98 observed on average in the sky), our algorithm can correct most of the spoofing errors (i.e., a correction rate close to 1, as seen in Table 3, as well as a spoofing detection accuracy above 87.2%). With 10 spoofed PRN codes, some correction is still achieved (with much smaller detection accuracy and correction rate), but with 11 spoofed satellites, the algorithm is completely unable to identify the genuine satellites, switching the roles and considering the spoofed satellites to be correct while identifying the genuine ones as spoofed. This happens because there are, on average, less than six genuine satellites when the spoofers' number is increased to 11 (or above). This effect is better understood when analyzing the true and corrected positions on 2D maps for 9, 10, and 11 spoofed satellites in Figure 6,

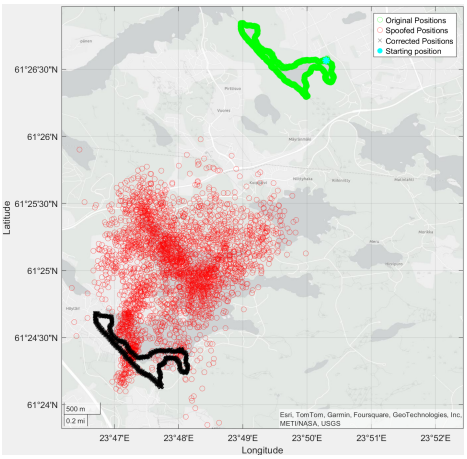
where for 9 spoofers, the corrected position is very close to the correct track; for 10 spoofers, only parts of the track are able to be corrected; and for 11 spoofers, the 'corrected' track converges fully to the spoofed positions (because in this case, the subset of spoofers is considered 'genuine').



(**a**) 9 spoofers



(**b**) 10 spoofers



(**c**) 11 spoofers

Figure 6. Comparison in the walk scenario of different numbers of spoofers and a spoofing error of $d = 5$ km; in this scenario, there were 16.98 visible PRN codes (satellites) on average.

7.2. Bus Scenario

In our second test, we analyze the data gathered during a bus ride in Cerdanyola del Vallès, Spain, in a university campus. The average bus speed was between 10 and 15 km/h due the campus speed limits. This scenario had an average of 11.35 visible satellites per time step.

In contrast with the walk scenario, from Figure 7, we can see a much more noticeable effect of the number of spoofers on the error after correction. This is mainly due to the lower average number of visible satellites in this scenario (e.g., with around 11 visible satellites on average in the sky, when 8 of them are spoofed, only 3 remain genuine, and this is insufficient for the correction algorithm to work). This result highlights both the impact of a higher number of spoofers and the importance of having a large set of visible satellites (which can be achieved by considering multiple GNSSs together). We remind the readers that for simplicity, in our studies, we only looked at GPS+Galileo satellites and we ignored the Glonass and Beidou satellites; the average number of visible satellites refers only to the GPS and Galileo satellites.

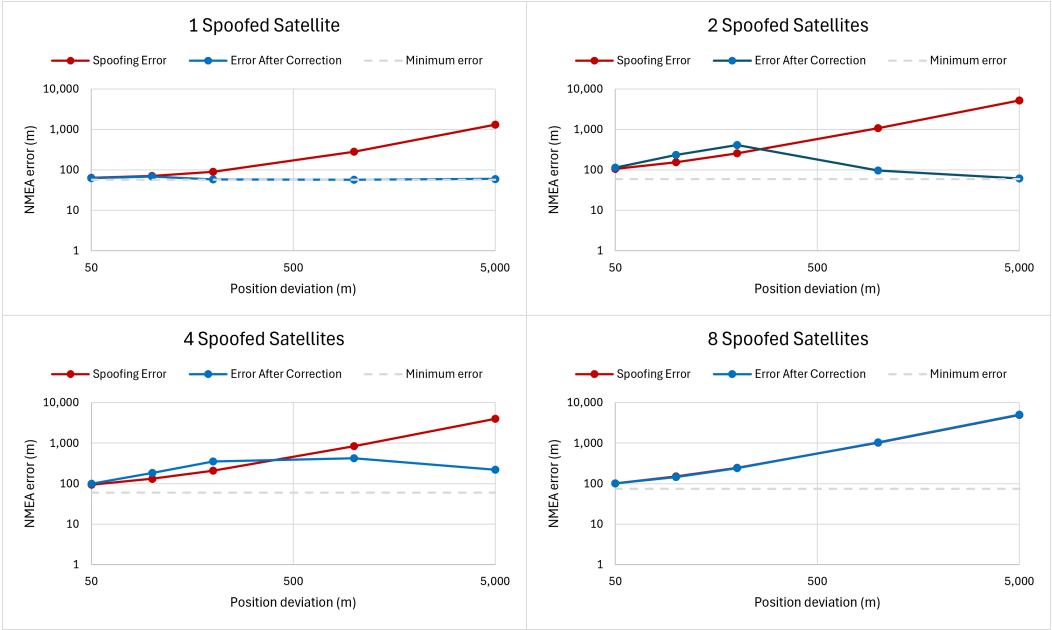


Figure 7. Three-dimensional positioning errors before and after correction for the bus scenario. For 8 spoofed PRN codes, the 'spoofing error' and 'error after correction' curves are overlapping, showing that the algorithm is not able to correct the spoofing error in this case, as there would be too few genuine satellites left.

7.3. Flight Scenario

In the last studied scenario in this section, we evaluated the performance of the algorithm on data collected during the whole duration of a flight between Riga and Tampere (the measurements were continuously taken during take off, cruising, and landing). For this scenario, we had an average of 12.14 GPS and Galileo satellites per time step (Figure 8).

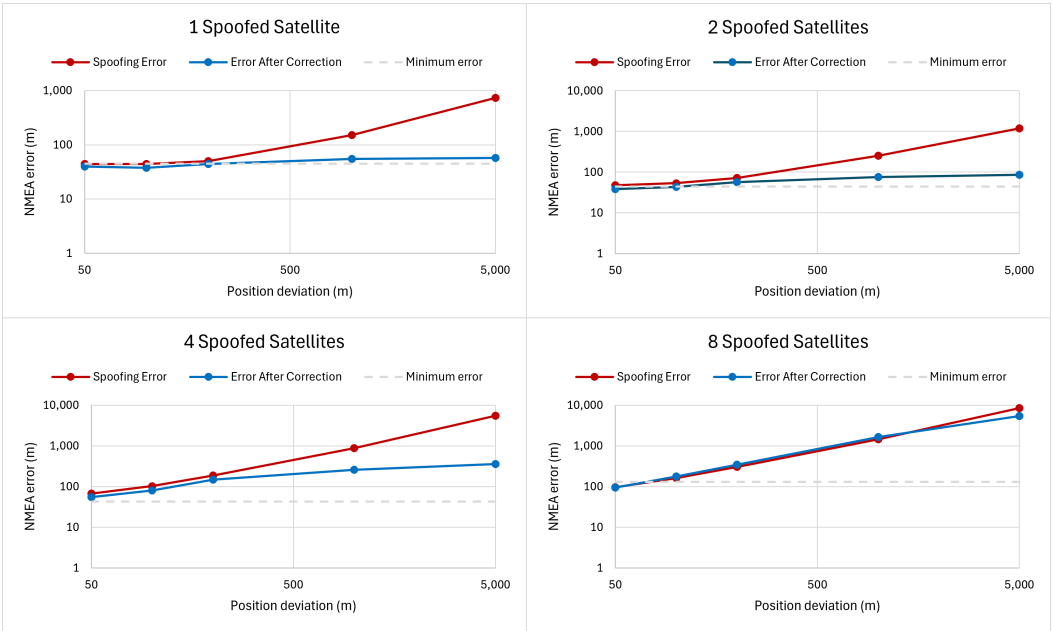


Figure 8. Three-dimensional positioning error in ECEF coordinates, before and after correction for the flight scenario, with a 12.14 average number of visible satellites.

Similarly to the previous tests, we can appreciate how spoofing correction becomes better the lower the number of spoofed satellites compared with the number of visible satellites is. Additionally, we can see how with eight spoofed satellites, there is almost no correction possible, given that in this case, the number of genuine satellites (around four) becomes smaller than the number of spoofed satellites (eight). By comparing Tables 4 and 5, we see a similar performance with one–five spoofed satellites, but then, the performance drops significantly in the bus scenario, while in the flight scenario, it only goes down to approximately 39% of the correction rate. This is due to the flight scenario being longer (4700 time steps vs. 840 time steps), since the algorithms performs worse initially and gradually improves its performance as it gathers more information from previous time steps.

Table 4. Three-dimensional positioning error in ECEF coordinates, before and after correction for the bus scenario, with 11.35 visible satellites on average.

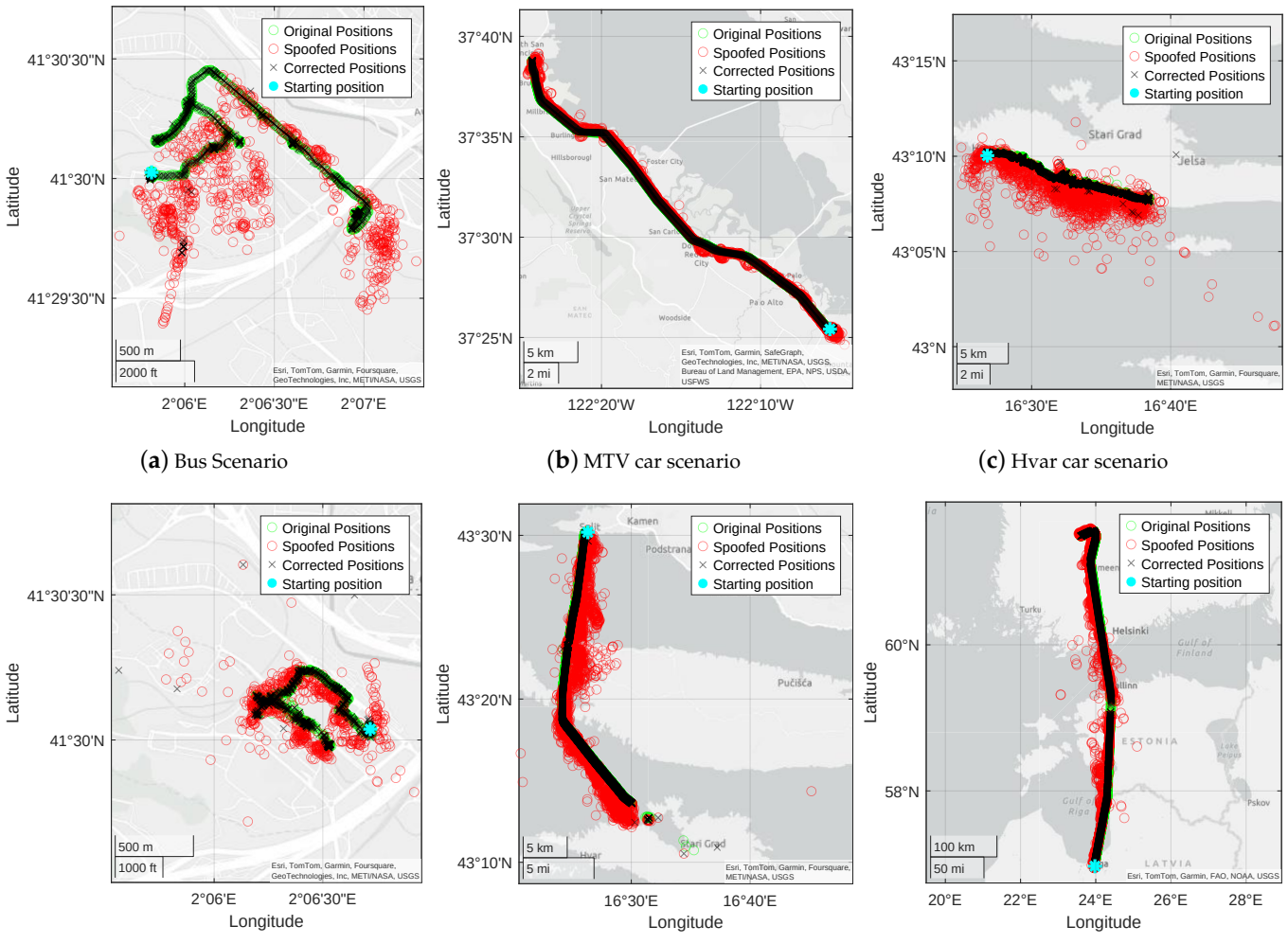
Number of Spoofed Satellites out of $N_{m,SV}$	Error Before Correction (m)	Error After Correction (m)	Correction Rate (-)	Detection Accuracy (%)
0	57.63	59.05	-	95.55
1	1305.63	59.72	0.998	98.54
2	5148.07	61.54	0.999	94.18
3	4859.17	148.43	0.982	88.51
4	4035.42	219.39	0.96	86.49
5	3397.82	365.59	0.912	86.64
6	4439.93	3144.02	0.297	42.9
7	5445.78	4933.5	0.096	3.23
8	4954.54	5041.54	−0.018	0.47
9	4686.72	5042.93	−0.077	0.64

Table 5. Three-dimensional positioning error in ECEF coordinates, before and after correction for Flight Scenario, with a 12.14 average number of visible satellites.

Number of Spoofed Satellites out of $N_{m,SV}$	Error Before Correction (m)	Error After Correction (m)	Correction Rate (-)	Detection Accuracy (%)
0	46.44	38.18	-	98.87
1	738.72	57.66	0.982	98.32
2	1191.12	85.71	0.964	95.83
3	2648.27	172.41	0.948	94.58
4	5516.44	354.9	0.943	94.08
5	6000.37	677.45	0.895	92.61
6	7232.33	1126.51	0.852	90.9
7	8048.48	2548.11	0.691	78
8	8397.27	5351.22	0.368	38.15
9	8303.17	5128.95	0.39	6.19

8. Comparative Results of Six Scenarios

This section now focuses on only one combination of spoofed PRN codes per scenario, and it compares six different scenarios. In Figure 9, we show the results of tests with these six different scenarios, and in Table 6, we specify the parameters and results of the tests. We can see good agreement among the considered scenarios, with spoofing correction rates of around 0.96–0.99; typically, scenarios with a lower number of visible satellites are more affected by spoofing presence than the ones with a higher number of visible satellites; also, longer tracks allow for more iterations and more updates in the historical indices and thus are able to offer higher correction rates. Similarly, we find consistency regarding the detection accuracy results: the detection accuracy is higher than 94% across all the tested scenarios. Both the correction rates and the detection accuracy show that our algorithm was able to successfully detect the spoofed satellites and exclude them in order to correct the spoofed positions in the majority of cases, as long as the number of spoofed satellites was lower than half of the average number of visible satellites in the sky. Our results also show an adequate mitigation capability of the algorithm for a wide range of scenarios, meaning scenarios with receivers at different speeds and in different urban or rural environments, and this suggests that the algorithm can perform reliably, independently of the conditions surrounding the receiver.



(**a**) Bus Scenario (**b**) MTV car scenario (**c**) Hvar car scenario

(**d**) Bike scenario (**e**) Ferry scenario (**f**) Flight scenario

Figure 9. Comparison of different scenarios.

Table 6. Comparative results at-a-glance, in terms of spoofing correction capabilities among 6 selected scenarios, illustrated in Figure 9.

Scenario	Bus	Car (MTV)	Car (Hvar)	Bike	Ferry	Flight
Number of spoofers	2	3	3	4	2	1
Deviation distance (km)	1	3	2	0.5	5	50
Average number of visible satellites	11.35	12.86	10.84	15.62	14.42	12.14
Error before correction (m)	1019.79	1112.65	3376.13	315.23	3079.19	15,511
Error after correction (m)	96.92	21.24	80.51	72.95	53.4	74.35
Correction rate (-)	0.963	0.991	0.994	0.986	0.998	0.998
Detection accuracy (%)	94.1	98.28	94.74	98.52	99.33	97.56

9. The Impact of Changing the Spoofed PRN Code and the Number of Spoofing Signals in a Spoofing Scenario

The developed algorithm assumes that the spoofed satellites (their PRN codes and number) do not change during the observation period. However, in this subsection, we also study how the algorithm behaves when this assumption does not hold and the spoofers change at some point.

To emulate this, we simulated the car scenario, and we added conditions for the spoofers change over time as follows:

- Initially, we have two fixed spoofed satellites, with indices 9 (picked among the GPS visible satellites) and 68 (picked among the Galileo visible satellites). We remark that these indices are based on some in-house mapping between PRN codes and the i-th index (e.g., for GPS, i is equal to the GPS PRN code; for GPS, i is equal to the Galileo PRN code plus 31, etc.); however, the mapping does not limit in any way our findings; thus, any mapping can be used as long as each satellite belonging to a different GNSS has a unique i index.
- After 20 min, we change the spoofing configuration and choose another subset of spoofed satellites; this time, we choose three different satellites to be spoofed, while the initial two go back to being genuine. Here, the spoofed satellites indices are 6 (GPS), 3 (GPS), and 62 (Galileo).
- Finally, after another 20 min, one satellite of each of the previous two subsets is spoofed, while the rest go back to the genuine state. Spoofed satellites: 9 (GPS) and 62 (GALILEO).

The considered scenario has an average of 10.84 visible satellites, out of which 2 or 3 are spoofed based on the above-mentioned models. The spoofing deviation was assumed to be 1km away from the correct position with a constant θ; see Equation (1).

Figure 10a shows the HRS in the considered scenario. We can see a significantly high rate of change in the HRS, which indicates that despite the changed assumptions, the proposed algorithm has good adaptability to this kind of dynamic scenarios with varying spoofing indices.

Figure 10b also shows the resulting positions calculated by the algorithm. The results are also in accordance with the HRS-based findings from Figure 10a and show good adaptability; indeed, our algorithm reduced the 3D WLS positioning error from 1185.23 m (without spoofing correction) to only 18.54 m. Taking into account that the minimum

achieved error in this track was 28.67 m, this translates into a correction rate, according to Equation (15), of more than 100%; this can be explained because the genuine satellite signals used in the *WLS* position estimate also contain some naturally faulty signals, which were discarded by our spoofing correction algorithm. By naturally faulty, we refer to the Radio Frequency Interference (RFI) that comes from the environment during the measurement process, and it was not introduced by us as spoofing signals. This ability of our algorithm to also correct some of the outliers or faulty signals (in addition to the spoofing signals) is discussed in the next subsection.

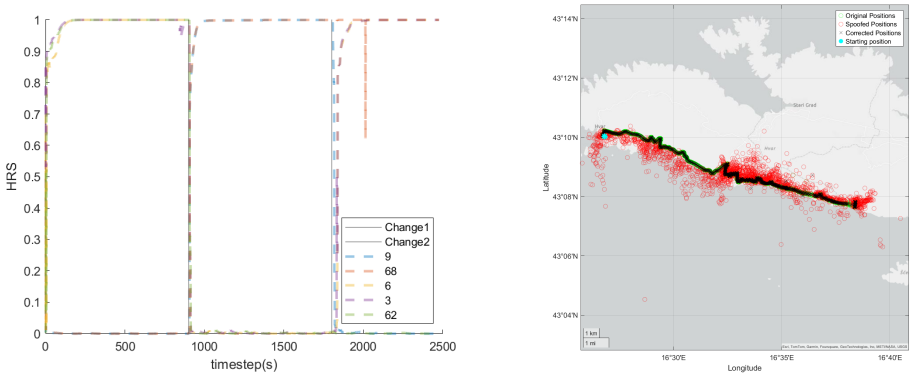


Figure 10. Impact of changing the spoofed PRN satellites during a spoofing scenario. (**a**) Change in HRS in dynamic spoofing scenario. (**b**) Position calculations in dynamic spoofing scenario.

Additional RFI Correction

In the flight scenario (between Riga and Tampere, on 22 March 2024), additional RFI (additional to that introduced by us, via the spoofing model described in Section 5) can be clearly appreciated at some track points. When running the algorithm with no spoofing, the positions calculated by the algorithm showed how these positions were corrected. This is shown in Figure 11, and the results are also reflected in the first row of Table 5, where we can see that the NMEA error is lower after the algorithm is run with no spoofing.

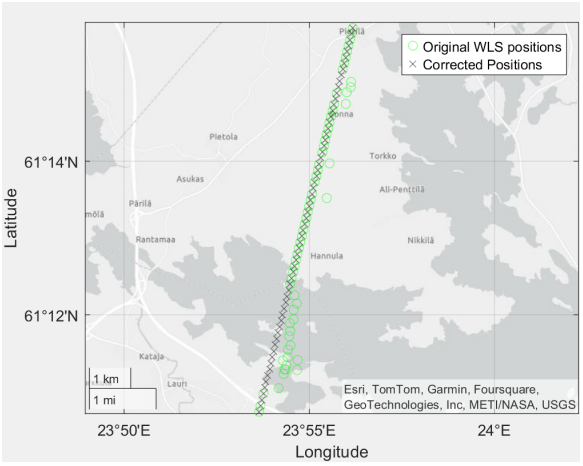


Figure 11. Correction of natural RFI with no spoofing.

10. Brief Discussion on Computational Complexity

In order to also address the algorithmic complexity, we adopted the method from [21]. The computational cost of any consistency-check-based algorithm is proportional to the number of candidate evaluations. Moreover, the computational cost of the candidate evaluation is dominantly affected by the computations required by the WLS solutions,

which correspond, on average, to about 50% of the total computational time. This estimate came from the following: When running the algorithm with a HP Pavilion Plus laptop with an Intel Ultra 7 at 3800 MHz, each candidate evaluation takes between 1 and 5 ms, while the rest of the algorithm takes about 20 ms (but the rest does not depend on the number of evaluations). At the same time, in terms of computing time, an average of 51.3% of the candidate evaluations corresponded to the WLS calculations. This means that in the worst-case scenario, where the solution is found with the iterative search, the computational cost will be dominantly determined by the number of candidate evaluations, which is proportional to the number of WLS calculations. For this reason, we will study the complexity of the algorithm in terms of WLS calculations n_s.

Subsequently, we analyze the computational complexity in terms of number of the candidates that we need to use a WLS solution (n_s), similar to [21].

The number of WLS positions will mostly depend on the probability of finding a valid subset of positions, which simultaneously depends on both the number of available satellites and the number of spoofed satellites. This is reflected in Table 7. It is interesting to see how in the walk scenario (where we had a high number of available satellites), with a higher number of spoofed satellites, the complexity decreased. This is because the probability of finding a valid WLS solution fast (i.e., candidate sets that either contain only genuine satellites or only spoofed satellites) increases with an increase in the number of spoofing signals, given that we would have enough signals to compute both a genuine and a completely spoofed position.

At the same time, in the walk scenario, we find that the largest complexity is with 0 spoofing. This is because the algorithm is developed to work with an approximate threshold of 5% false alarm rate; thus, with a high number of satellites, it is likely to find signals below that threshold. When the threshold is lowered to 1%, the number of WLS solutions is reduced to 4.21, but the position error after the spoofing correction is slightly increased from 20.55 m to 20.8 m. In the bus and flight scenarios, where we have a lower number of visible satellites (on average) than in the walk scenario, the complexity varies non-monotonically with the number of spoofers (it is to be recalled that increasing the number of spoofers also means decreasing the number of genuine signals, as the overall number of visible satellites is the sum between the number of spoofers and the number of genuine signals).

Table 7. Number of WLS solutions for three example scenarios, each with a position deviation of 5 km.

Number of Spoofed Satellites	Walk Scenario	Bus Scenario	Flight Scenario
0	15.71	4.03	5.25
1	11.4	2.19	4.1
2	9.89	4.03	3.65
3	7.87	13.72	3.83
4	5.85	18.35	5.27
5	6.17	17.67	12.05
6	8.6	6.28	21.29
7	5.09	10.07	46.03
8	3.85	10.62	10.6
9	3.55	11.93	7.72

The complexity values in Table 7 also match the results reported in [21] (the average number of *WLS* solutions in [21] was reported to be 23; in our case, most cases require less than 23 WLS solutions); the authors in [21] have compared these numbers with traditional RAIM algorithms and noticed that the complexity of traditional RAIM algorithms is typically one order of magnitude higher than 10–20 WLS computations.

11. Conclusions and Open Issues

This paper proposed a spoofing model with a varying number of spoofed PRN codes starting from data collected with Android devices, via the GNSSlogger app, and proposed a consistency-check-based spoofing detection and mitigation algorithm relying on historical probability indices associated to each of the visible satellite in the sky, as well as on linking indices between pairs of all possible satellites belonging to the selected constellations. The main findings of our work comprise three main aspects: proposing a spoofing model starting from raw GNSS Android data, making available several such raw GNSS Android datasets as open-access datasets on Zenodo (at https://zenodo.org/records/14712684 (accessed on 19 Feburary 2025)) for enabling result reproducibility, and proposing a consistency-check spoofing detection algorithm based on the assumption that only a part of the visible PRN satellites are spoofed. Our studies focused only on GPS plus Galileo constellations, but they are by no means limited to only one combination of GNSSs and can be applied for any combinations of the four GNSSs (GPS, Galileo, Glonass, and Beidou). The consistency checks relied on various estimates of receiver position and clock error, obtained with a simple Weighted Least Squares algorithm, and focused on how to select subsets of the visible satellites in an efficient manner, so that the WLS computations were kept to a minimum.

In terms of our main findings, the scenario-by-scenario results for three scenarios with different speeds (walk, bus, and flight) showed that there is an upper number of spoofers above which the algorithm is unable to detect or correct the spoofers, and this upper number is dependent on the average number of visible satellites in the sky; as a rule of thumb, we showed that we were able to detect and correct the spoofing errors if the spoofer number was no more than 50% of the average number of visible satellites in the sky. When comparing six scenarios of various multi-modal transportation and with a low number of spoofers (up to four), we noticed that the detection accuracy and correction rates were very similar between them and were not highly affected by the transportation mode, being able to reach correction rates above 0.96 and spoofing detection accuracy rates above 94%.

In terms of the possible model limitations, one limitation of our spoofing models is that we rely on the assumption that not all satellites in the sky are spoofed simultaneously; such assumption can serve well in studies pertaining robustness to an increase in the number of spoofers or in studies focusing on RF fingerprinting to classify genuine and spoofing signals, but it may be not applicable in all spoofing situations. The situation where the spoofers transmit the same PRN codes as the signals in the sky but at higher power remains to be further investigated. Another assumption used in our algorithms was that the spoofed PRN codes remain unchanged during the duration of the observation; however, we have also relaxed this assumption and showed that our algorithm remains valid also when the spoofing signals change during the duration of observation.

Last but not least, we also addressed the complexity of the proposed consistency-check algorithm. For the sake of the reproducibility of our results, we have be also shared our Android measurements on the Zenodo repository (link available at the Data Availability information below). Extensions of our algorithm which remain for further investigations include extensions to multi-frequency L1/L5 (or E1/E5) measurements. Validation with

real-life spoofers also remains an open topic, as this is conditional to the availability of real-life GNSS spoofing data. Further research directions are also about improving spoofing mitigation via machine learning and deep learning approaches, following, for example, the steps in [30], and looking at Jammertest spoofing datasets, when such datasets become available in the public domain.

Author Contributions: Conceptualization, E.T.M.; methodology, E.T.M.; software, E.T.M.; validation, E.T.M.; formal analysis, E.T.M. and E.S.L.; data curation, E.T.M.; writing—original draft preparation, E.T.M. and E.S.L.; writing—review and editing, E.S.L.; visualization, E.T.M. and E.S.L.; supervision, E.S.L.; project administration, E.S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been mainly supported by the LEDSOL project funded within the LEAP-RE program by the European Union's Horizon 2020 Research and Innovation Program under grant agreement 963530 and the Research Council of Finland, grant 352364. This work was also supported by the Tampere University TURNS Mobility fund (covering parts of the costs for measurement data collection) and by ECIU Y-MOVE seed funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: GNSSLogger datasets for the six scenarios where we collected the measurements are available as open-access datasets at https://zenodo.org/records/14712684 (accessed on 19 Feburary 2025).

Acknowledgments: The authors would like to thank the other members in our team at Tampere University who contributed to the Matlab in-house software application to process the raw Android data and for fruitful discussions regarding the GNSSLogger apps and pseudorange-based analyses in the GNSS.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kuna, D.; Perumalla, N.K. An Analysis of Multi-GNSS Observations From Android Smartphones. In Proceedings of the 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, 8–9 April 2023; pp. 961–965. [CrossRef]
2. Suzuki, T. Precise Position Estimation Using Smartphone Raw GNSS Data Based on Two-Step Optimization. *Sensors* **2023**, *23*, 1205. [CrossRef] [PubMed]
3. Spens, N.; Lee, D.K.; Nedelkov, F.; Akos, D. Detecting GNSS Jamming and Spoofing on Android Devices. *Navig. J. Inst. Navig.* **2022**, *69*, navi.537. [CrossRef]
4. Liu, S.; Cheng, X.; Yang, H.; Shu, Y.; Weng, X.; Guo, P.; Zeng, K.C.; Wang, G.; Yang, Y. Stars Can Tell: A Robust Method to Defend against GPS Spoofing using Off-the-shelf Chipset. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Online, 11–13 August 2021.
5. Fu, G.M.; Khider, M.; van Diggelen, F. Android Raw GNSS Measurement Datasets for Precise Positioning. In Proceedings of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2020), Online, 22–25 September 2020; Institute of Navigation: Long Beach, CA, USA, 2020. [CrossRef]
6. Grenier, A.; Lohan, E.S.; Ometov, A.; Nurmi, J. Towards Smarter Positioning through Analyzing Raw GNSS and Multi-Sensor Data from Android Devices: A Dataset and an Open-Source Application. *Electronics* **2023**, *12*, 4781. [CrossRef]
7. Weng, X.; Ling, K.V. Localization with Noisy Android Raw GNSS Measurements. *arXiv* **2023**, arXiv:2309.08936. [CrossRef]
8. Medina, E.T. GPS and Galileo Spoofing Attack Simulation. Bachelor's Thesis, Tampere University, Tampere, Finland, 2024. Available online: https://urn.fi/URN:NBN:fi:tuni-202407187670 (accessed on 15 January 2025).
9. Psiaki, M.; Humphreys, T. Civilian GNSS Spoofing, Detection, and Recovery; In *Position, Navigation, and Timing Technologies in the 21st Century: Integrated Satellite Navigation, Sensor Systems, and Civil Applications*; IEEE: Piscataway, NJ, USA, 2021; pp. 655–680. [CrossRef]
10. Morales-Ferre, R.; Richter, P.; Falletti, E.; de la Fuente, A.; Lohan, E.S. A Survey on Coping with Intentional Interference in Satellite Navigation for Manned and Unmanned Aircraft. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 249–291. [CrossRef]

11. Islam, S.; Bhuiyan, M.Z.H.; Pääkkönen, I.; Saajasto, M.; Mäkelä, M.; Kaasalainen, S. Impact analysis of spoofing on different-grade GNSS receivers. In Proceedings of the 2023 IEEE/ION Position, Location and Navigation Symposium (PLANS), Monterey, CA, USA, 24–27 April 2023; pp. 492–499. [CrossRef]
12. Psiaki, M.L.; Humphreys, T.E. GNSS Spoofing and Detection. *Proc. IEEE* **2016**, *104*, 1258–1270. [CrossRef]
13. Gallardo, F.; Pérez-Yuste, A.; Konovaltsev, A. Satellite Fingerprinting Methods for GNSS Spoofing Detection. *Sensors* **2024**, *24*, 7698. [CrossRef] [PubMed]
14. Wang, W.; Lohan, E.S.; Sanchez, I.A.; Caparra, G. Pre-correlation and post-correlation RF fingerprinting methods for GNSS spoofer identification with real-field measurement data. In Proceedings of the 2022 10th Workshop on Satellite Navigation Technology (NAVITEC), Virtual, 5–7 April 2022; pp. 1–10. [CrossRef]
15. Wang, Y.; Kou, Y.; Huang, Z. Necessary Condition for the Success of Synchronous GNSS Spoofing. *Chin. J. Electron.* **2023**, *32*, 438–452. [CrossRef]
16. Elango, A.; Landry, R.J. XAI GNSS—A Comprehensive Study on Signal Quality Assessment of GNSS Disruptions Using Explainable AI Technique. *Sensors* **2024**, *24*, 8039. [CrossRef] [PubMed]
17. Ji, N.; Rao, Y.; Wang, X.; Zou, D. Advanced GNSS Spoofing Detection: Aggregated Correlation Residue Likelihood Analysis. *Remote Sens.* **2024**, *16*, 2868. [CrossRef]
18. Galan, A.; Fernandez-Hernandez, I.; Cucchi, L.; Seco-Granados, G. OSNMAlib: An Open Python Library for Galileo OSNMA. In Proceedings of the 2022 10th Workshop on Satellite Navigation Technology (NAVITEC), Virtual, 5–7 April 2022; pp. 1–12. [CrossRef]
19. Hao, Y.; Shi, C.; Xu, A.; Sui, X.; Xia, M. Revealing Methods of GNSS Spoofing Mitigation Through Analyzing the Spoofing Impacts on Adaptively Robust Estimation-Based RTK/INS Tightly Coupled Integration. *IEEE Sens. J.* **2023**, *23*, 25165–25178. [CrossRef]
20. Bai, L.; Sun, C.; Dempster, A.G.; Zhao, H.; Feng, W. GNSS Spoofing Detection and Mitigation With a Single 5G Base Station Aiding. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 4601–4620. [CrossRef]
21. Wu, Q.; Cui, X.; Lu, M.; Yang, P.; Wu, P. A GNSS Anti-Spoofing Technique Based on the Spatial Distribution Characteristic of the Residual Vectors. *Tsinghua Sci. Technol.* **2024**, *29*, 457–468. [CrossRef]
22. Radoš, K.; Brkić, M.; Begušić, D. Recent Advances on Jamming and Spoofing Detection in GNSS. *Sensors* **2024**, *24*, 4210. [CrossRef] [PubMed]
23. Lee, J.; Schmidt, E.; Gatsis, N.; Akopian, D. Detection and Mitigation of Spoofing Attacks Against Time Synchronization and Positioning. *IEEE Access* **2023**, *11*, 138986–139003. [CrossRef]
24. Imad, M.; Grenier, A.; Zhang, X.; Nurmi, J.; Lohan, E.S. Ionospheric Error Models for Satellite-Based Navigation—Paving the Road towards LEO-PNT Solutions. *Computers* **2023**, *13*, 4. [CrossRef]
25. Van Diggelen, F. GNSS Measurements Update GNSS Raw Measurements from Android Phones. In Proceedings of the GSA Raw Measurements Workshop, Prague, Czech Republic, 30 May 2018. Available online: https://www.euspa.europa.eu/sites/default/files/expo/frank_van_diggelen_keynote_android_gnss_measurements_update.pdf (accessed on 19 February 2025).
26. Kaplan, E.; Hegarty, C. *Understanding GPS/GNSS: Principles and Applications*, 3rd ed.; Artech House: Norwood, MA, USA, 2017.
27. Borre, K.; Akos, D.M.; Bertelsen, N.; Rinder, P.; Jensen, S.H. *A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach*; Birkhäuser: Basel, Switzerland, 2007.
28. Lohan, E.S.; Kodom, T.; Lebik, H.; Grenier, A.; Zhang, X.; Cramariuc, O.; Mocanu, I.; Bierwirth, K.; Nurmi, J. Raw GNSS Data Analysis for the LEDSOL Project—Preliminary Results and Way Ahead—Trepo—urn.fi. In Proceedings of the CEUR WIPHAL 2023: Work-in-Progress in Hardware and Software for Location Computation, ICL-GNSS Conference, Castellon, Spain, 6–8 June 2023. Available online: https://urn.fi/URN:NBN:fi:tuni-202310118783 (accessed on 19 February 2025).
29. Grenier, A. Development of a GNSS Positioning Application under Android OS Using GALILEO Signals. Master's Thesis, Ecole Nationale des Sciences Géographiques, Champs-sur-Marne, France, 2019.
30. Li, R.; Liao, H.; An, J.; Yuen, C.; Gan, L. Intra-Class Universal Adversarial Attacks on Deep Learning-Based Modulation Classifiers. *IEEE Commun. Lett.* **2023**, *27*, 1297–1301. [CrossRef]