

**DE-41 (EE)**

**Usama,**

**Umer,**

**Hassan,**

**Mohsin**

## **GNSS Anti Spoofing Using GNSS SDR**



**COLLEGE OF  
ELECTRICAL AND MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY  
RAWALPINDI  
2023**

**COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING**



**DE-41 EE  
PROJECT REPORT**

**GNSS Anti Spoofing Using GNSS SDR**

Submitted to the Department of Electrical Engineering  
in partial fulfillment of the requirements  
for the degree of  
**Bachelor of Engineering**  
in  
**Electrical**  
**2023**

**Sponsoring DS:**

**Submitted By:  
Usama Hameed  
Umer Bukhari  
Hassan Tariq  
Mohsin Ali**

## **CERTIFICATE OF APPROVAL**

It is to certify that the project “GNSS Anti Spoofing Using GNSS SDR” was done by NS Usama Hameed, NS Umer Bukhari, NS Hassan Tariq and NS Mohsin Ali under the supervision of Dr. Fahad Mumtaz Malik. This project is submitted to Department of Electrical Engineering, College of Electrical and Mechanical Engineering (Peshawar Road Rawalpindi), National University of Sciences and Technology, Pakistan in partial fulfillment of requirements for the degree of Bachelor of Engineering in Electrical Engineering.

### **Students:**

**1. Usama Hameed**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

**2. Umer Bukhari**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

**3. Hassan Tariq**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

**4. Mohsin Ali**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

### **Approved By:**

Project Supervisor: \_\_\_\_\_

Date: \_\_\_\_\_

**Dr. Fahad Mumtaz Malik**

## DECLARATION

We hereby declare that no portion of the work referred to in this Project Thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning. If any act of plagiarism is found, we are fully responsible for every disciplinary action taken against us depending upon the seriousness of the proven offence, even the cancellation of our degree.

**1. Usama Hameed**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

**2. Umer Bukhari**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

**3. Hassan Tariq**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

**4. Mohsin Ali**

NUST ID: \_\_\_\_\_

Signature: \_\_\_\_\_

### **COPYRIGHT STATEMENT**

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the author.
  
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
  
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

## **ACKNOWLEDGEMENT**

In the name of Almighty Allah, the most gracious and the most merciful, we are grateful for the strength, health, and the opportunity to complete this Final Year Project.

Our deepest appreciation goes to our supervisors, Dr. Fahad Mumtaz Malik, and Muhammad Ali Akhtar. Their guidance, encouragement, and critiques have been invaluable to this project. Their in-depth knowledge and unwavering support enabled us to work efficiently on GNSS Anti-Spoofing using GNSS SDR, and their mentorship was pivotal in our understanding of this crucial subject.

We extend our sincere thanks to NESCOM for the opportunity to contribute to such an important project. Their invaluable support and provision of the necessary resources and facilities made this project possible.

To our teammates - Usama Hameed, Umer Bukhari, Hassan Tariq, and Mohsin Ali - our journey together has been intellectually enriching and inspiring. Their diligence, cooperation, and continuous support throughout this project have not only contributed to our academic growth but also made this a memorable experience.

We would also like to express our gratitude to our families for their unwavering support and faith in our abilities. Their constant encouragement has been a driving force behind our efforts.

Finally, we are grateful to everyone who was involved in this project either directly or indirectly. Their contributions have been integral to the successful completion of our project.

## **ABSTRACT**

The ubiquity of Global Navigation Satellite Systems (GNSS) in our modern world has undeniably enhanced various aspects of human life. However, the susceptibility of GNSS to spoofing attacks, where falsified signals mislead GNSS receivers, poses severe security threats. As these systems become increasingly integrated into critical infrastructure and operations, the need for robust spoofing detection mechanisms is more critical than ever. This thesis delves into the exploration, implementation, and evaluation of GNSS Anti-Spoofing Techniques on Software-Defined Radios (SDRs), a flexible and cost-effective approach to GNSS signal processing.

The study commences with a comprehensive examination of the nature of GNSS spoofing and the current anti-spoofing methodologies in use. Various GNSS anti-spoofing techniques are then scrutinized, followed by their implementation on an adaptable GNSS SDR platform. The performance of these techniques is critically evaluated to determine their efficacy in spoofing detection.

The results of this study demonstrate the potential of GNSS SDR in providing a dynamic and adaptable solution to counter GNSS spoofing. It is anticipated that the findings will help in advancing the current state of GNSS anti-spoofing mechanisms, promoting more secure and reliable use of GNSS. The implemented anti-spoofing techniques could serve as a foundation for further advancements in this field, ultimately contributing to the enhancement of GNSS security against the growing threat of spoofing attacks.

## **SUSTAINABLE DEVELOPMENT GOALS**

Considering all the possible sustainable development goals the following are directly/indirectly associated with our project:

SDG 9: Industry, Innovation, and Infrastructure: Anti-spoofing technologies contribute to the resilience and safety of infrastructure systems, including transportation and communication networks that rely on GNSS data. It can also be considered an innovation in the field of navigation technology, ensuring reliable and secure data for various industries.

SDG 11: Sustainable Cities and Communities: By ensuring the security of GNSS, this project indirectly supports the development of sustainable cities. Secure and reliable navigation systems are crucial for transportation, emergency services, and many urban services, which can make cities and communities safer and more sustainable.

SDG 16: Peace, Justice, and Strong Institutions: By preventing malicious activities (spoofing) which can disrupt social, economic, and political stability, the project indirectly contributes to peace and justice. Secure navigation and timing information is vital for many societal functions, including law enforcement and border control.



## **Table of Contents**

ACKNOWLEDGEMENT .....	i
ABSTRACT.....	ii
SUSTAINABLE DEVELOPMENT GOALS.....	iii
LIST OF FIGURES .....	vii
LIST OF TABLES .....	viii
LIST OF SYMBOLS .....	ix
Chapter 1: INTRODUCTION.....	1
1.1 Background .....	1
1.2 Motivation .....	2
1.3 Problem Statement .....	2
1.4 Objectives of the Study .....	3
Chapter 2: BACKGROUND AND LITERATURE REVIEW .....	5
2.1 Introduction to Global Navigation Satellite Systems (GNSS).....	5
2.1.1 Fundamentals of GNSS .....	5
2.1.2 Position, Velocity, and Time .....	5
2.1.3 Coordinate System.....	5
2.2 Status of Global Navigation Satellite Systems.....	5
2.2.1 Global Positioning System (GPS).....	6
2.2.2 European GNSS (Galileo) .....	6
2.2.3 GLONASS System .....	6
2.2.4 BeiDou System .....	6
2.3 GNSS Receivers.....	6
2.3.1 Hardware-based GNSS Receivers .....	7
2.3.2 Software-based GNSS Receivers.....	7
2.4 GNSS-SDR .....	7
2.4.1 An Overview.....	7
2.4.2 Fundamentals .....	9

2.4.2.1	Understanding the Basics .....	9
2.4.2.2	Design Principles of GNSS-SDR .....	10
2.4.2.3	Kahn's Process Networks .....	10
2.4.2.4	GNU Radio .....	10
2.4.2.5	Software Architecture in GNSS-SDR.....	12
2.4.3	The Control Plane .....	17
2.4.4	Signal Processing Blocks.....	18
2.5	Methods of Global Navigation Satellite System (GNSS) Spoofing .....	20
2.5.1	Introduction to Spoofing.....	20
2.5.2	Methods of Spoofing .....	21
2.5.2.1	Replay Spoofing Attack Strategies (RSA).....	21
2.5.2.2	Forgery Spoofing Attack Strategies (FSA).....	23
2.5.2.3	Estimation Spoofing Attack Strategies (ESA).....	24
2.5.2.4	Advanced Spoofing Attack Strategies (ASA) .....	25
2.6	Anti-Spoofing Techniques.....	27
2.6.1	Introduction.....	28
2.6.2	Methods Independent of Additional Hardware Facilities .....	28
2.6.2.1	Doppler Shift-Based Methods .....	28
2.6.2.2	Consistency Check-Based Methods .....	30
2.6.2.3	Signal Parameter Statistics Analysis-Based Methods .....	35
2.6.2.4	Arrival Time and Arrival Time Difference-Based Methods .....	37
2.6.2.5	Residual Signal Detection-Based Methods .....	39
2.6.3	Spoofing Detection Method Based on Signal Quality Monitoring.....	41
2.6.3.1	The Principle of C/N0 as an Anti-Spoofing Detection Method.....	42
2.6.3.2	Spoofing Attack Metric Model (SAMM) .....	43
2.6.3.3	Spoofing Signal Elimination .....	43
2.6.3.4	Time Hopping Anti-spoofing Signal Processing Algorithm.....	43
2.6.3.5	Carrier Phase Tracking Spectrum Analysis .....	44
2.6.3.6	Time-Authentication Algorithm .....	44
2.6.4	Spoofing Detection Based on Signal Arrival Direction.....	44
Chapter 3	Methodology .....	47
3.1	Hardware Components.....	47

3.1.1	Realtek RTL2832U USB Dongle .....	47
3.1.2	T-bias.....	48
3.1.3	GPS L1 Band Active Antenna .....	49
3.1.4	System Configuration and Preliminary Results.....	50
3.2	Implementation of various anti spoofing techniques .....	53
3.2.1	Power level check implementation .....	53
3.2.2	Carrier-to-noise density ratio (C/No) check: .....	53
3.2.3	Position Consistency Check.....	55
3.2.3.1	Static position check .....	55
3.2.3.2	Position Jump Check .....	57
3.2.4	Velocity Consistency Check .....	59
3.2.5	Abnormal Position Check.....	61
3.2.6	Clock Offset Check.....	62
3.2.7	Clock Jump Test.....	64
Chapter 4:	Results .....	67
4.1	Acquisition of PVT Information .....	67
4.2	Generation and Analysis of Spoofing Report.....	71
Chapter 5:	CONCLUSIONS AND FUTURE WORK.....	74
5.1	Conclusion.....	74
5.2	Future Work.....	74
REFERENCES	.....	76

## LIST OF FIGURES

Figure 1 GNU Radio Block Diagram.....	11
Figure 2 Block interface for all Signal Processing blocks.....	14
Figure 3 Class hierarchy in GNSS-SDR and its connection with GNU Radio .....	16
Figure 4 RSA Scenario.....	22
Figure 5 Schematic Diagram of FSA.....	23
Figure 6 Nulling Attack Signal Diagram .....	26
Figure 7 The categories of anti-spoofing technologies based on signal level [35] .....	28
Figure 8 Spoofing detection scenario based on Doppler shift. ....	29
Figure 9 Spoofing and Spoofing detection scenario based on consistency check. ....	30
Figure 10 Spoofing detection scenario based on signal parameter statistical analysis .....	35
Figure 11 Spoofing detection principles based on arrival time and arrival time difference.....	38
Figure 12 Residue signal detection block diagram.....	40
Figure 13 Spoofing detection scenario based on angle of arrival.....	45
Figure 14 RTL 2832U Dongle .....	48
Figure 15 RTL 2832U Dongle Internals.....	48
Figure 16 T-Bias .....	49
Figure 17 GPS L1 Band Active Antenna.....	50
Figure 18 Configuration File For Real Time Data.....	52
Figure 19 Carrier to Noise Density Ratio Check Flow Chart .....	55
Figure 20 Static Position Check flow chart.....	57
Figure 21 Position jump Test Flow Chart.....	59
Figure 22 Abnormal Position Check FlowChart.....	62
Figure 23 Clock Offset Check.....	64
Figure 24 Clock Jump Check Test .....	66
Figure 25 Configuration File part 1 .....	67
Figure 26 Configuration File part 2 .....	68
Figure 27 Configuration File part 3 .....	68
Figure 28 Configuration File part 4 .....	69
Figure 29 Start of Decoding.....	70
Figure 30 PVT Solution of GPS L1 data.....	70
Figure 31 Spoofing Detection Report.....	72

## **LIST OF TABLES**

Table 1 Comparison of anti-spoofing method based on consistency check .....	32
Table 2 Analysis and comparison for spoofing detection methods based on statical Analysis of signal Parameters .....	34

## LIST OF SYMBOLS

Greek	
$D_n(t)$	the navigation message
$C_n(t)$	the ranging code
$\tau_n(t)$	the signal propagation delay
$f_0$	the standard carrier frequency
$f_d$	the doppler frequency
$\theta_n$	the beat carrier phase
$I_n(t)$	potential interference
$N$	the number of constituents spreading-code specific signals.
$A_i$	the amplitude of the signal
$D_i$	the data of the signal
$C_i$	the PN code of the signal
$\tau_i(t)$	the code phase of the signal
$\phi_i(t)$	the beat carrier phase of the signal's
$f_0$	emitted frequency.
$v$	wave speed
$v_o$	speed of the observer (receiver)
$v_s$	the speed of the source (satellite)
$\Gamma(\tau, f)$	the cross-ambiguity function,
$\tau$	time delay,
$f$	frequency offset,
$s_1(t)$ & $s_2(t)$	received and expected signals respectively,
$j$	imaginary unit
$t$	time.
$P_t$	transmitted power,
$G_t$ & $G_r$	the gains of the transmitting and receiving antennas, respectively.
$\lambda$	wavelength,
$d$	distance between the transmitter and receiver

### Acronyms

GNSS	Global Navigation Satellite Systems
SDR	Software-Defined Radio
RSA	Replay Spoofing Attack Strategies
FSA	Forgery Spoofing Attack Strategies
ESA	Estimation Spoofing Attack Strategies
ASA	Advanced Spoofing Attack Strategies
SAMM	Spoofing Attack Metric Model
C/No	Carrier-to-noise density ratio
GPS	Global Positioning System
PNT	positioning, navigation, and timing
PVT	position, velocity, and time (PVT)
ECEF	Earth-Centered, Earth-Fixed
LLA	latitude, longitude, and altitude
ASICs	application-specific integrated circuits
RINEX	Receiver Independent Exchange Format
SMT	simultaneous multi-threading
UML	Unified Modeling Language
RFFE	radio frequency front end
SCER	Security Code Estimation and Replay
FEA	Forward Estimation Attack

NMA	Navigation Message Authentication
ASA	Advanced Spoofing Attack Strategies
PTD	Power Threshold Detector
DOD	Doppler Offset Detector
GLRT	Generalized likelihood ratio test
CAF	Cross-ambiguity function matrix
ARPSO	Attractive and Repulsive Particle Swarm Optimization
MLE	Maximum Likelihood Estimation
TDOA	Time difference of arrival
DCP	Differential Code Phase
SQM	Signal quality monitoring

# **Chapter 1: INTRODUCTION**

## **1.1 Background**

Modern society depends on the use of Global Navigation Satellite Systems (GNSS). Global positioning, navigation, and timing (PNT) data are provided to users of these systems, which include the Global Positioning System (GPS) of the United States, Galileo of Europe, GLONASS of Russia, and BeiDou of China. These systems are capable of supporting a wide range of applications, such as temporal synchronization for global banking and telecommunications networks, as well as personal navigation, aviation, maritime navigation, disaster relief, and scientific research..

However, the critical services offered by GNSS face a significant threat from spoofing attacks. In a spoofing attack, an adversary broadcasts counterfeit GNSS signals that mimic the properties of authentic signals but carry misleading PNT information. As a result, a spoofed GNSS receiver might output erroneous position or time data, potentially leading to catastrophic outcomes. For example, spoofing could misguide a ship into hazardous waters, redirect a drone, disrupt a telecommunications network, or cause a GPS-guided missile to miss its target.

Despite the significant risks associated with GNSS spoofing, the current state of anti-spoofing is not sufficient to mitigate these threats. Traditional GNSS receivers, which use hardware-based signal processing, lack the flexibility to adapt to new types of spoofing attacks that might emerge in the future. As a result, once a receiver is deployed, it may not be able to defend against new threats unless it is physically replaced or modified.

The utilization of Software-Defined Radio (SDR) presents a feasible resolution to the problem by replacing hardware-based components such as mixers, filters, amplifiers, and modulators/demodulators with software-based counterparts. The GNSS SDR enables the dynamic updating of signal processing algorithms to effectively address emerging threats, thereby offering a versatile and resilient framework for safeguarding against GNSS spoofing.

This project explores the potential of implementing GNSS Anti-Spoofing Techniques on GNSS SDR as a solution for effective and adaptable spoofing detection. The focus is to investigate the various anti-spoofing techniques, implement them on a GNSS SDR platform, and evaluate their performance against a range of spoofing scenarios. Through this work, we aim to contribute to the development of more robust GNSS spoofing defenses and help safeguard the integrity of GNSS services now and in the future.



## 1.2 Motivation

The motivation behind this research originates from the growing reliance on GNSS services across various sectors, coupled with the increasing sophistication of GNSS spoofing threats. The potential disruption that GNSS spoofing can inflict on safety-critical applications, such as air traffic control, maritime navigation, and autonomous vehicles, underscores the importance of developing robust and adaptable anti-spoofing mechanisms.

Moreover, the risk extends beyond these critical systems into everyday technology. Smartphones, wearable devices, and even some modern home appliances now incorporate GNSS services, and hence, are potential targets for spoofing attacks. The proliferation of GNSS-enabled devices amplifies the potential impact of spoofing attacks and the necessity of effective mitigation strategies.

While current anti-spoofing techniques offer some defense against these threats, they lack the flexibility to adapt to the evolving landscape of spoofing attacks. Existing hardware-based GNSS receivers are constrained by their static nature, limiting their ability to respond to novel attack techniques that were not foreseen when the receiver was designed and deployed.

In contrast, a GNSS SDR platform offers a dynamic and adaptable solution. The software-defined nature of SDRs allows for the implementation of updated anti-spoofing algorithms as new threats emerge, offering a promising pathway towards a robust defense against GNSS spoofing.

This project seeks to leverage the flexibility and versatility of GNSS SDR to enhance our defense against GNSS spoofing. By investigating, implementing, and evaluating various anti-spoofing techniques on a GNSS SDR platform, we aim to provide a contribution to the ongoing efforts to safeguard GNSS services against the threat of spoofing. The ultimate goal is to help ensure the continued reliability of GNSS, supporting the countless applications that depend on these services in our increasingly interconnected world.

## 1.3 Problem Statement

While the issue of GNSS spoofing has been widely recognized, the practical solutions to this problem remain challenging. Current anti-spoofing techniques, while effective against known forms of spoofing, often lack the flexibility and adaptability to contend with new, more sophisticated spoofing methods. Moreover, traditional GNSS receivers, implemented primarily in hardware, have inherent limitations in their ability to be modified or upgraded post-deployment. This characteristic severely restricts their effectiveness in responding to evolving spoofing threats.

On the other hand, GNSS Software-Defined Radios (SDRs) offer a promising solution due to their adaptability and flexibility. As the signal processing in SDRs is software-based, these systems can be upgraded or modified dynamically to incorporate new anti-spoofing techniques or adapt to novel spoofing threats. However, the full potential of SDRs in the context of GNSS spoofing detection and mitigation is yet to be realized.

The overarching problem that this project addresses is the development of robust, adaptable anti-spoofing techniques that can be implemented on a GNSS SDR platform. Specific challenges include:

- Understanding the full spectrum of potential GNSS spoofing attacks, including novel and sophisticated methods that could be employed by malicious actors.
- Investigating current GNSS anti-spoofing techniques, understanding their strengths and limitations, and identifying potential improvements.
- Implementing these improved anti-spoofing techniques on a GNSS SDR platform, addressing technical challenges associated with real-world conditions, such as signal noise and multipath effects.
- Evaluating the performance of the implemented techniques under a variety of spoofing scenarios and identifying any remaining vulnerabilities.

By addressing these challenges, this project aims to contribute significantly to the field of GNSS anti-spoofing, paving the way for more secure and reliable GNSS services.

#### **1.4 Objectives of the Study**

The primary goal of this project is to develop effective and adaptable GNSS spoofing detection methods using GNSS Anti-Spoofing Techniques on a GNSS SDR platform. The study aims to not only enhance the understanding of GNSS spoofing and its detection but also to provide a blueprint for future research and development in this critical area of study. The specific objectives of this research are:

- To gain an in-depth understanding of the nature, types, and impacts of GNSS spoofing. This includes studying different spoofing methods, their underlying principles, and the associated potential threats they pose to various GNSS applications.
- To carry out a comprehensive review of the existing GNSS anti-spoofing techniques, identifying their strengths and weaknesses. This objective also involves a detailed study of how these techniques work, their effectiveness against various types of spoofing attacks, and areas where they fall short.
- To implement selected or improved anti-spoofing techniques on a GNSS SDR platform. This step involves overcoming the technical challenges associated with SDR implementation and integrating the chosen techniques into the GNSS SDR software stack.
- To evaluate the performance of the implemented anti-spoofing techniques under a variety of spoofing scenarios.
- To contribute to the broader academic and industry knowledge in the field of GNSS anti-spoofing by sharing the findings of the research.

Through the successful achievement of these objectives, this study hopes to contribute to the ongoing efforts in combating GNSS spoofing, ultimately enhancing the security and reliability of GNSS services worldwide.

## **Chapter 2: BACKGROUND AND LITERATURE REVIEW**

### **2.1 Introduction to Global Navigation Satellite Systems (GNSS)**

#### **2.1.1 Fundamentals of GNSS**

Globally accessible autonomous geospatial location is made possible by the Global Navigation Satellite Systems (GNSS) family of satellite navigation systems. Using signals from several satellites, they enable electronic receivers to pinpoint their location (longitude, latitude, and altitude). The term "GNSS" refers to a variety of systems, including the Global Positioning System (GPS) of the United States, Galileo of Europe, GLONASS of Russia, and BeiDou of China.

At the heart of GNSS functionality is the principle of trilateration, where the position of an object can be determined by measuring its distance from several known points. In the context of GNSS, these points are the satellites themselves. Each GNSS satellite broadcasts signals that include the satellite's current time and position. A GNSS receiver picks up these signals, calculating the distance to each satellite based on the speed of light and the time it took for each signal to arrive.

#### **2.1.2 Position, Velocity, and Time**

GNSS receivers calculate their position, velocity, and time (PVT) based on signals from at least four GNSS satellites. The position is determined through trilateration, as mentioned above. Velocity is calculated by measuring the Doppler shift of the received signals, and time is derived from the timestamps in the received signals. Accurate PVT information is essential for a wide range of applications, from navigation and geodesy to time synchronization and disaster relief.

#### **1.1.3 Coordinate System**

The Global Navigation Satellite System (GNSS) employs Earth-Centered, Earth-Fixed (ECEF) coordinates as its inherent coordinate system. The system under consideration defines the Earth's center of mass as the point of origin, with the equatorial plane serving as the plane for the X and Y axes. Additionally, the Z-axis aligns with the Earth's rotational axis. Notwithstanding, in practical applications, GNSS receivers frequently transform these ECEF coordinates into latitude, longitude, and altitude (LLA) for the sake of convenience, as these are more comprehensible to human users.

### **2.2 Status of Global Navigation Satellite Systems**

Each GNSS has its unique status and development path. As of my knowledge cutoff in 2021, all of them (GPS, Galileo, GLONASS, and BeiDou) are fully operational, with continuous upgrades and improvements being made.

### **2.2.1 Global Positioning System (GPS)**

The Global Positioning System, developed by the United States, was the first operational GNSS. As of 2021, GPS has a constellation of 31 active satellites, providing global coverage and serving as a critical component in a wide range of applications.

### **2.2.2 European GNSS (Galileo)**

The GNSS created by the European Union is called Galileo. It is intended to offer a very accurate positioning system on which European countries may rely without the assistance of the Chinese BeiDou, American GPS, or Russian GLONASS systems. Galileo will be completely operational by 2021.

### **2.2.3 GLONASS System**

GLONASS is Russia's GNSS. It was developed during the Soviet era as an alternative to the United States' GPS system. GLONASS is fully operational as of 2021 and continues to be maintained and improved by the Russian government.

### **2.2.4 BeiDou System**

BeiDou is China's GNSS. It was developed to provide an independent navigation system for Chinese military and civilian users. As of 2021, BeiDou provides global coverage and continues to be upgraded by the Chinese government.

## **2.3 GNSS Receivers**

GNSS receivers are electronic devices that receive and process signals from GNSS satellites to determine the receiver's position, velocity, and time. Receivers vary greatly in complexity and precision, ranging from simple, single-frequency receivers used in smartphones to complex, multi-frequency receivers used for geodesy and other high-precision applications. The development of GNSS receivers is an active area of research and development, with ongoing efforts to improve receivers.

### **2.3.1 Hardware-based GNSS Receivers**

Hardware-based GNSS receivers, as the name suggests, rely on specific hardware components to receive and process GNSS signals. These receivers include specialized chips designed for signal acquisition, tracking, and navigation computation. A key characteristic of hardware-based receivers is that their signal processing algorithms are implemented in the hardware itself, often in application-specific integrated circuits (ASICs). These ASICs are designed for low power consumption and high-speed operation, making hardware-based receivers suitable for real-time applications and portable devices.

However, the static nature of hardware based GNSS receivers means that once they are designed and manufactured, their algorithms cannot be modified or updated without physically replacing or altering the hardware. This limitation presents a significant challenge in the face of evolving GNSS spoofing threats, as these receivers may not be able to adapt to new types of attacks that were not anticipated when the receiver was designed.

### **2.3.2 Software-based GNSS Receivers**

Software-based GNSS receivers, often referred to as Software-Defined Radios (SDRs), take a different approach to GNSS signal processing. Instead of using hardware ASICs, SDRs perform signal processing in software running on a general-purpose processor. This approach provides a high level of flexibility and adaptability, as the signal processing algorithms can be updated or modified through software changes, without needing to alter the hardware.

The dynamic nature of SDRs allows them to adapt to new GNSS signals or changes in the signal environment. For example, if a new type of GNSS spoofing attack emerges, an SDR could potentially be updated to detect and mitigate that attack, while a hardware-based receiver might be vulnerable to the attack until its hardware can be replaced or updated.

However, the flexibility of SDRs comes with some trade-offs. The computational demands of software-based signal processing can be higher than for hardware-based processing, leading to higher power consumption. SDRs also require a more complex design and validation process to ensure that the software algorithms function correctly in all situations. Despite these challenges, the adaptability of SDRs makes them a promising solution for combating the evolving threat of GNSS spoofing.

## **2.4 GNSS-SDR**

### **2.4.1 An Overview**

The GNSS-SDR project is an open-source initiative that seeks to develop a dependable software-defined receiver for the Global Navigation Satellite System (GNSS) using the C++ programming language. This software program facilitates the creation of a GNSS software receiver, which is conceptualized as a graphical representation consisting of nodes that correspond to signal processing blocks and lines that depict the flow of data between them. The software has the capability to establish communication with a diverse range of RF front ends that are compatible, encompassing all the links in the receiver's chain until the navigation system.

The GNSS-SDR's design allows for a high degree of customization, such as the option to interchange signal sources, apply diverse signal processing algorithms, ensure compatibility with a range of systems, and adapt output formats. In addition, it offers interfaces to all intermediary signals, parameters, and variables.

The core objective of this project is to generate effective, reusable, and maintainable code, which also minimizes bugs. The resulting software is designed to produce highly optimized executables that function effectively across diverse hardware platforms and operating systems. Some of the key considerations addressed in this challenge are efficiency, performance, concurrency, portability, real-time operation capability, and extendibility.

The software receiver connects to a regular personal computer and offers USB and Ethernet bus interfaces to pre-made or commercial RF front ends. In addition to processing raw data samples saved in a file, it may modify processing algorithms in accordance with various sampling frequencies, intermediate frequencies, and sample resolutions.

Signal collection, tracking of accessible satellite signals, navigation message decoding, and computation of necessary observables for positioning algorithms, which finally calculate the navigation solution, are all tasks carried out by the system. The GNSS-SDR is made to make it easier to include new signal processing methods and provides a simple way to assess how they will affect the performance of the receiver.

Every software component is subjected to rigorous functional validation, and the entire receiver is tested using both actual and artificial signals, all in the name of quality assurance. The output can be delivered as RTCM 3.2 messages over a TCP/IP server in real-time or saved in a variety of forms, including the Receiver Independent Exchange Format (RINEX), which is utilized by most geodetic processing applications for GNSS. KML and GeoJSON formats are used to store navigation results.

The architecture of GNSS-SDR is founded on the widely recognized framework of GNU Radio, which offers the essential signal processing runtime and processing blocks required for the implementation of software radio applications. The software-defined GNSS receiver is constructed by utilizing this framework in GNSS-SDR. The aforementioned architectural structure comprises

two distinct components, namely the Control Plane and the Signal Processing Plane. The Control Plane is accountable for generating a flow chart and flexibly enabling and disabling channels in response to the evolving composition of received GNSS signals. In contrast, the Signal Processing Plane comprises a set of modules that execute algorithms for digital signal processing.

This innovative approach allows GNSS-SDR to transform an abstract concept, like a Signal Source (which could represent an RF front-end or a file), into a functional software-defined receiver. The software reads samples from a Signal Source, processes GNSS signals, and computes a position fix, embodying a cutting-edge application of GNSS and SDR technologies.

## **2.4.2 Fundamentals**

### 2.4.2.1 Understanding the Basics

Baseband signal processing for Global Navigation Satellite Systems (GNSS) is a computationally intense operation. Achieving real-time processing can be a hurdle even on modern computer systems unless there's an efficiently designed software architecture capable of maximizing the functionality of the executing processors. Thus, it becomes crucial to leverage the inherent parallelisms within the processing platform to fulfill real-time requirements.

In architectural parallelism, one principal model can be observed in shared-memory parallel computers. These systems have the ability to simultaneously handle multiple tasks. This can be accomplished either by assigning different tasks to various processors, or through a process known as simultaneous multithreading (SMT), where multiple instruction streams are executed in an overlapping manner on a single processor. Another approach involves a combination of both these strategies.

The simultaneous execution of numerous independent instruction streams, often known as threads, is supported by concurrent multithreading platforms, multicore processors, and shared memory parallel computers. Task parallelism is what this is known as, and the top programming languages, compilers, and operating systems all handle it effectively. The program operating on the platform should be designed in a way that enables it to split its workload over numerous execution cores in order to take full advantage of this possible speed improvement. Applications and operating systems that are multi-threaded in nature support this functionality. The number of processing cores can practically linearly increase the execution speed of a well-designed software.

The following sections elucidate the fundamental concepts and software design principles that form the foundation of GNSS-SDR.



#### **2.4.2.2 Design Principles of GNSS-SDR**

The process of task parallelization involves the distribution of execution processes, also known as threads, among various parallel computing nodes or processors. This allows for the processing of distinct threads on the same or different data sets. The efficiency of the design is crucial to prevent bottlenecks that may impede the entire processing chain and hinder real-time operation.

#### **2.4.2.3 Kahn's Process Networks**

The methodology under consideration is based on Gilles Kahn's formal and mathematical representation of process networks. He aimed to develop a language that relies on accurate semantics of process interaction, enabling structured programming of process networks that evolve dynamically.

The Kahn process model exemplifies a computational paradigm in which processes are interconnected via communication channels to form a network. The aforementioned procedures produce discrete data entities, commonly referred to as tokens, which are subsequently conveyed through a communication conduit and subsequently employed by the recipient process. Communication channels are the exclusive medium through which procedures exchange information. According to Kahn's model, the execution of a process is required to be temporarily halted when it attempts to retrieve data from an input channel that is currently unoccupied. For example, it is impermissible for a procedure to inspect the input for the existence or nonexistence of information. At a particular point in time, a process has the option to either be in an enabled state or a blocked state, where it is waiting for data on a singular input channel. The process is constrained from engaging in a state of waiting for data from several channels. Systems that conform to Kahn's mathematical model exhibit determinism, whereby the production of tokens on communication channels is independent of the order of execution. With the implementation of an appropriate scheduling policy, it is possible to establish process networks for software-defined radio that exhibit two essential attributes.

Non-termination refers to the interpretation of a flow graph process that runs indefinitely without any scenarios resulting in deadlock.

The communication channels maintain a constant quantity of buffered data units, regardless of their potential execution orders, adhering to strict bounds.

#### **2.4.2.4 GNU Radio**

GNU Radio serves as a living embodiment of these ideas, providing a free and open-source framework tailored to the needs of software-defined radio applications. It offers a wide array of

signal processing blocks, such as filters, synchronization elements, demodulators, and decoders, among other features. Moreover, GNU Radio comes with an inbuilt runtime scheduler, designed to comply with the needs described earlier, letting developers concentrate on actual signal processing implementation rather than being concerned with their efficient integration into the processing chain.

By incorporating the signal processing framework of GNU Radio, GNSS-SDR aligns its software architecture with a well-established, supremely efficient design and rigorously validated execution.

A standard processing block, or a particular node in the flow graph as executed by GNU Radio, is depicted below:

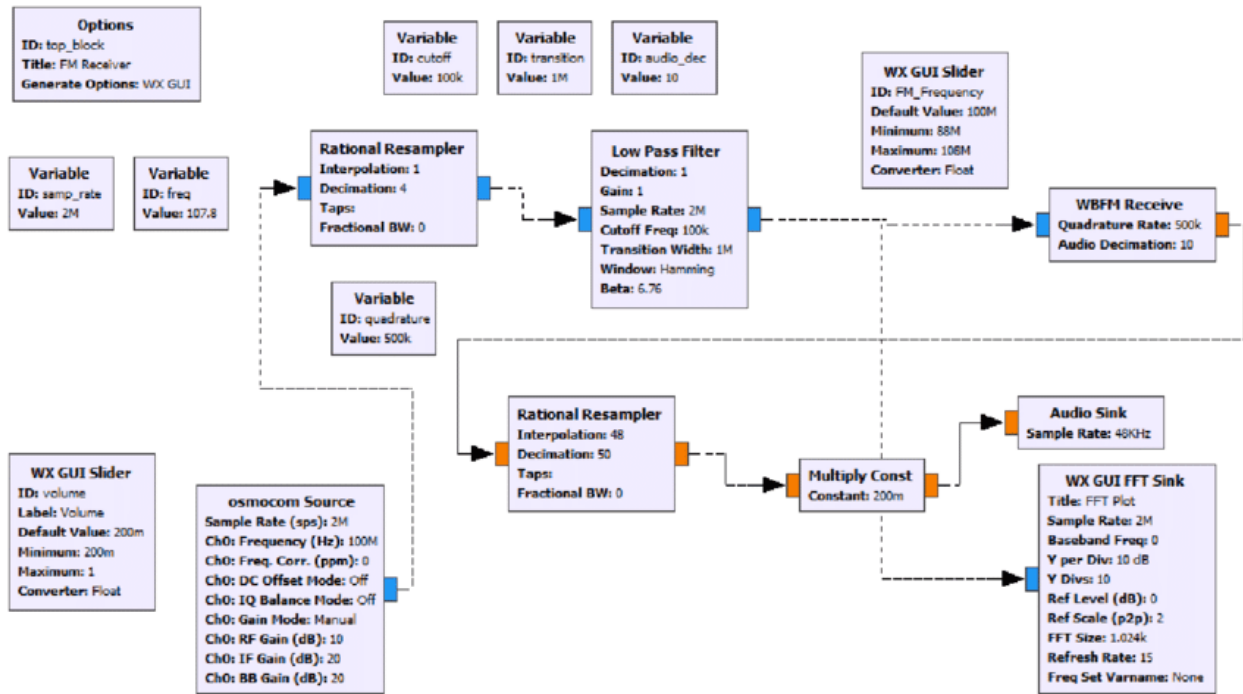


Figure 1 GNU Radio Block Diagram

Within GNU Radio, individual blocks function autonomously, each with its own scheduler that operates within a distinct execution thread. Additionally, it possesses an asynchronous messaging system that enables communication with other upstream and downstream blocks. The primary signal processing occurs within the 'work()' function.

A block has the capability to support multiple input and output ports for data, as well as asynchronous message communication with other blocks within the flow graph. The GNU Radio

framework incorporates an inherent process scheduler that facilitates the transfer of objects, or data units, from sources to sinks within any given application. The processing capacity of a block in a single cycle is contingent upon the quantity of items contained in the input buffer(s) and the available space in the output buffer(s) of the block. The enhancement in efficiency is directly proportional to the quantity due to the fact that the processing of samples constitutes a significant portion of the overall processing time, albeit with an accompanying delay. Conversely, a decrease in the number of items per cycle results in an increase in the scheduler's overhead.

For the processing chain to operate effectively, it is imperative that both the input buffers contain the required number of items, and the output buffer has sufficient space. The blocks within GNU Radio are equipped with a runtime scheduler that dynamically performs computations, aiming to optimize performance and manage network scheduling.

In this particular model, every processing block functions within its own autonomous thread, with the objective of expeditiously processing data from their respective input buffer(s), regardless of the input data rate. The data flow within the flow graph, spanning from the source(s) to the sink(s), is managed by a runtime scheduler that is integrated within the system.

In this particular arrangement, signal processing blocks that are software-defined are designed to scan the available samples in their input memory buffer(s), execute rapid processing, and subsequently store the results in the corresponding output memory buffer(s). Each individual block operates on its own distinct thread. Through the implementation of this approach, a software-based receiver is generated with the capability to consistently endeavor towards the maximization of its input signal processing capacity. Irrespective of the rate of input data, the operational speed of each block within the flow graph is determined by the processing capacity, data flow, and buffer space available. Achieving real-time processing involves deploying the complete processing pipeline of the receiver onto a resilient system capable of managing the requisite processing burden. Nevertheless, this does not impede the possibility of executing the identical process at a reduced pace, such as by retrieving specimens from a document on a system with limited computational capability.

#### **2.4.2.5 Software Architecture in GNSS-SDR**

In this section, we delve into an object-oriented programming-based software design that enables efficient building of software-defined GNSS receivers.

Notation:

A simplified version of the Unified Modeling Language (UML), a widely used modeling language in the field of object-oriented software engineering, is utilized in our approach. The representation of classes is depicted as rectangular shapes that are partitioned into two distinct sections. The upper

section denotes the name of the class, while the lower section enumerates the methods associated with the class..

A broken arrow from “ClassA to ClassB” signifies a dependency relationship, denoting ClassA's reliance on ‘ClassB’. In C++ language, this mostly leads to an `#include`.

Dependence between Classes: ClassA is dependent on ClassB.

Inheritance demonstrates is a and is like relationships, aiding in reusing pre-existing data and code. When ClassB inherits from ClassA, ClassB is the subclass of ClassA, while ClassA is the superclass (or parent class) of ClassB. A line with a closed arrowhead pointing from the subclass to the superclass indicates inheritance in UML.

Inheritance among Classes: ClassA is inherited by ClassB.

Class hierarchy:

The establishment of relationships among multiple classes is a fundamental aspect of object-oriented software architecture. The fundamental class for signal processing blocks within the GNU Radio framework is the abstract foundation class, `gr::basic_block`. This class serves to encapsulate an entity that possesses a name, as well as a collection of inputs and outputs. The abstract parent class of `gr::block`, which is the fundamental base class for all processing blocks, is represented by `gr::hier_block2`. Despite not being explicitly instantiated, `gr::hier_block2` functions as a recursive container that has the ability to add or remove processing or hierarchical blocks from the internal graph. A signal processing flow is represented through the creation of a hierarchical block tree, which may contain terminal nodes capable of executing signal processing operations at any level.

Class hierarchy in GNU Radio:

The `gr::top_block` class is the uppermost hierarchical block that symbolizes a flow graph. It establishes GNU Radio runtime functions utilized during program execution such as `run()`, `start()`, `stop()`, `wait()`, etc. It helps in designing the receiver flow graph, i.e., the interconnections among all necessary blocks as per the configuration.

`GNSSBlockInterface` is a common interface for all GNSS-SDR modules and defines pure virtual methods that need implementation by a derived class.

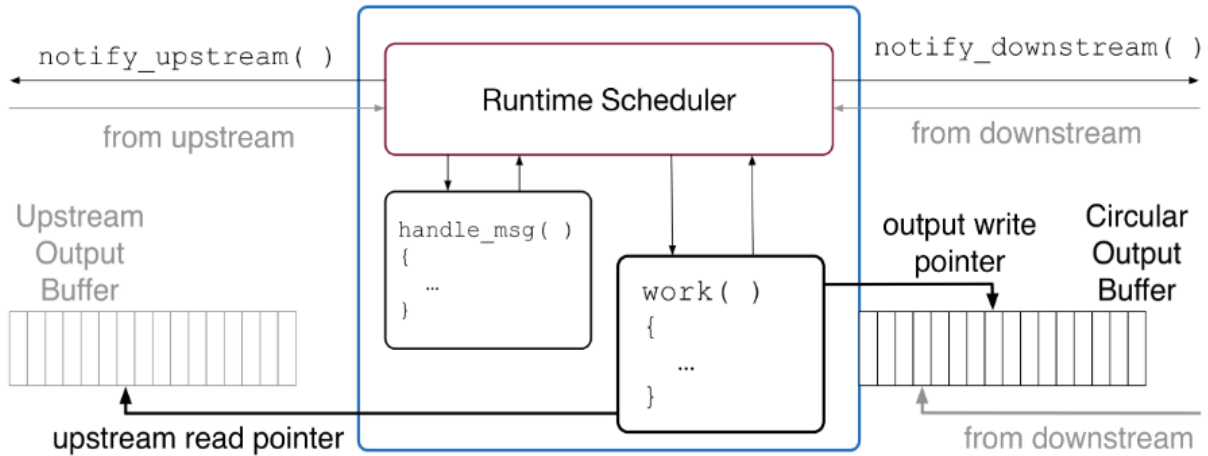


Figure 2 Block interface for all Signal Processing blocks.

Definition:

Pure virtual method classes are referred to as abstract. They can't be instantly created. Only after a parent class or that class has implemented all of the inherited pure virtual methods may a subclass of an abstract class be constructed directly.

Interfaces for the receiver's processing blocks have been defined as subclasses of GNSSBlockInterface. For each processing block, this structure enables the definition of an infinite number of algorithms and implementations, which are then instantiated according to the configuration. The purpose of separating interfaces from implementations is achieved by this strategy, which defines many implementations that share the same interface. It creates a group of algorithms, wraps them all, and allows for easy swapping between them. Consequently, the algorithm can change without affecting the software that uses it.

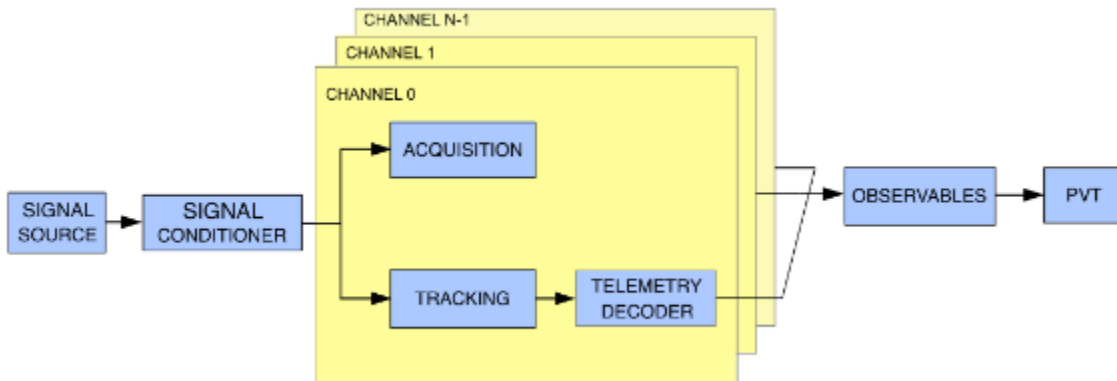


Figure typical GNSS SDR flow graph

Class hierarchy for the Signal Processing Plane:

For each block, this design pattern provides an infinite variety of algorithms and implementations. For instance, in order to design a novel method for signal acquisition, both an adaptor to check that it complies with a minimum AcquisitionInterface and the actual implementation in the form of a processing block for GNU Radio (i.e., one that derives from gr::block) are required.

GPS\_L1\_CA\_PCPS\_Acquisition is an illustration of a readily accessible implementation of an Acquisition block. It has an adaptor that descended from AcquisitionInterface, like other Acquisition blocks, and a matching GNU Radio block that descended from gr::block and carried out the processing..

General class hierarchy for GNSS-SDR

The following image illustrates the general class hierarchy for GNSS-SDR and its relation to the GNURadio framework:

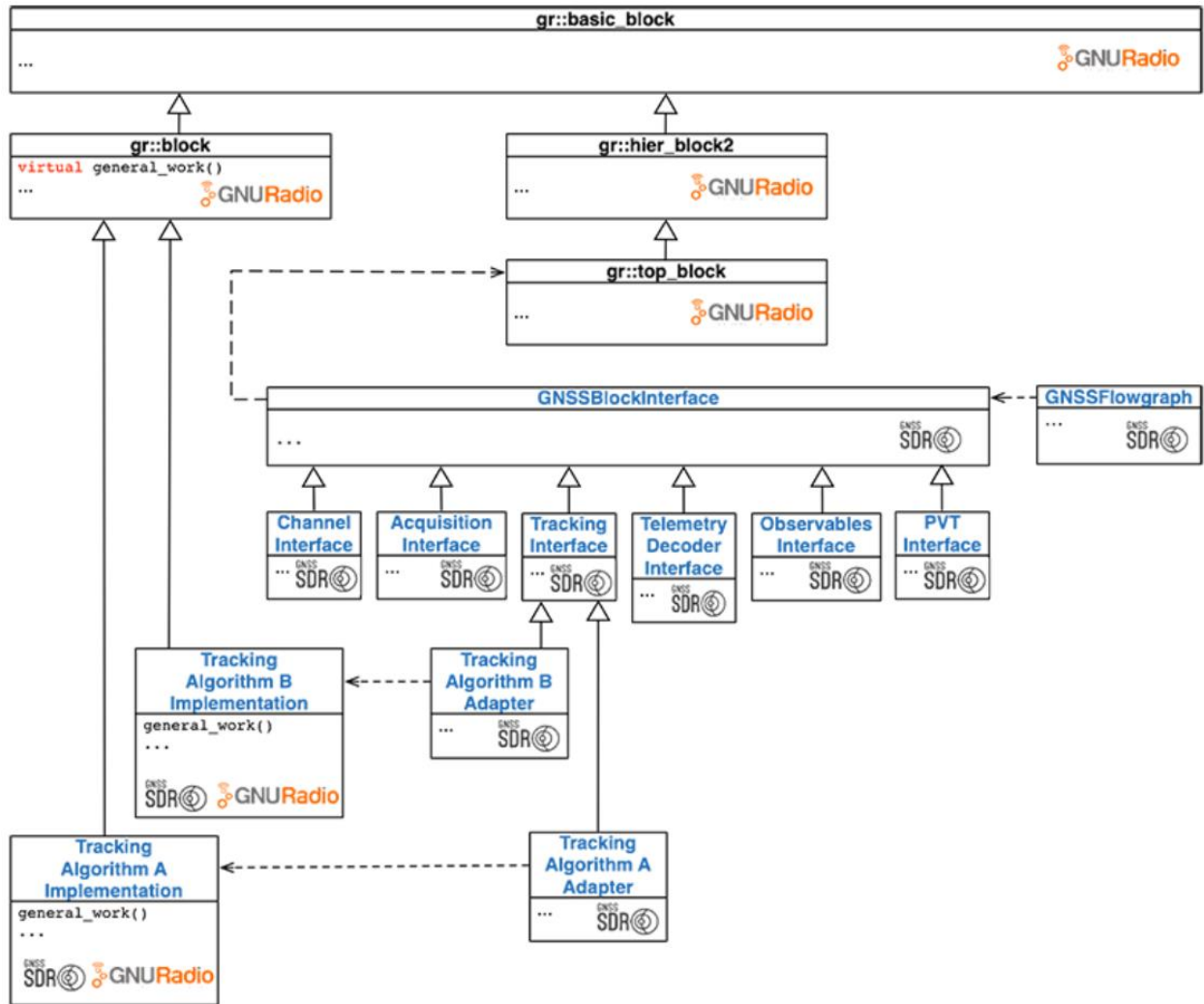


Figure 3 Class hierarchy in GNSS-SDR and its connection with GNU Radio

Thus far, we have examined a software architecture that achieves a harmonious equilibrium between efficacy and scalability. The implementation of efficient process scheduling techniques can involve the creation of a flow graph consisting of processing nodes that model the GNSS receiver. This flow graph comprises a source block that provides signal samples, a network of nodes that read input buffers and write to output buffers, and a sink block. Subsequently, a software architecture is proposed, which expands upon the GNU Radio framework and establishes interfaces for the fundamental building components of GNSS processing. Moreover, the utilization of this approach allows us to generate an unbounded quantity of executions for every fundamental GNSS signal processing component. All of these executions share the same fundamental structure, thus making them easily reusable. As an illustration, it is feasible to construct acquisition implementations for GPS L1 C/A, Galileo E1B, and other signals. These blocks can be utilized akin to any other pre-existing GNU Radio block, leveraging features such as their runtime scheduler or asynchronous message passing system. Further elaboration is required regarding the interconnectivity of the blocks, the operational mechanics of the system as a whole, and the

procedural steps for configuring the blocks to construct a distinctive software defined GNSS receiver. The responsibilities of the Control Plane have been enumerated above.

### **2.4.3 The Control Plane**

The Control Plane is responsible for establishing a flow graph in line with the configuration and then overseeing the operation of the processing blocks. It is made up of four primary components:

A flexible configuration mechanism that accommodates an indefinite number of algorithms, implementations, corresponding parameters, and use cases.

A GNSS block factory, which simplifies the process of creating processing blocks, pointing to the newly formed object via a standard interface. This feature enables the addition of new blocks without altering any code in the software that employs it.

A GNSS flow graph that organizes the creation of processing blocks as per the configuration and connects them to form a software receiver.

A Control Thread, which manages the entire system.

Now, we will delve into how these components are implemented in GNSS-SDR.

#### **Configuration Mechanism:**

The mechanism of configuration enables users to effortlessly define a personalized receiver. The document outlines the flow graph, encompassing various aspects such as the signal source type, channel count, algorithms utilized in each channel and module, strategies for selecting satellites, and the format type of the output, among other factors. The system was developed with the intention of being easily expandable to meet future needs, while minimizing the impact on existing code. This is achieved through the direct association of the variable names in the processing blocks with the names of the configuration parameters.

The Configuration Interface class is used to distribute properties within the program. File Configuration and In Memory Configuration are two implementations of this interface. File Configuration reads and internally stores properties from a file, while In Memory Configuration remains empty after instantiation and uses the `set_property` method to set property values and names.

When configuration parameters need to be read, instances of Configuration Interface will provide the values. A full GNSS receiver can be defined uniquely in a text file in INI format using the configuration mechanism, thus enabling a highly flexible and extendable system.

#### **GNSS Block Factory:**



The application defines a straightforward accessor class that retrieves configuration pairs of values and transfers them to the GNSSBlockFactory, which serves as the factory class. The factory, based on its configuration, determines the appropriate class to instantiate and the corresponding parameters to be passed to its constructor. Thus, the factory manages the complexity of instantiating blocks. The decoupling of the blocks' implementations from the configuration syntax enables significant expansion of the application's functionalities and simplifies the development of entirely personalized receivers.

#### GNSS Flow Graph:

The task of establishing the block graph in accordance with the setup, running it, making real-time modifications to it, and stopping it is under the purview of the GNSSFlowgraph class. To configure the generic graph, this class understands which roles must be created and how to link them. It applies the connections between the GNU Radio blocks, depends on the configuration to obtain the appropriate instances of the roles it requires, and gets the graph ready for launch. The GNSSFlowgraph also controls real-time modifications to the flow graph's setup, such as dynamic channel reconfiguration and selecting a satellite selection strategy..

#### Control Thread:

The Control Thread class assumes the responsibility of initializing the GNSS Flowgraph and providing the requisite configuration. Upon establishment of the flow graph and interconnection of the blocks, the processing of the incoming data stream is initiated. The Control Thread entity is responsible for overseeing the control queue and executing all messages transmitted by the processing blocks through a message queue that ensures thread safety.

Each individual configuration file serves as a distinctive definition for a Global Navigation Satellite System (GNSS) receiver. Illustrative instances of such files can be found at `gnss-sdr/conf`. The configuration mechanism of GNSS-SDR exhibits a high degree of flexibility, enabling the implementation of intricate flow graphs. These may include a multi-system receiver that caters to diverse signals with distinct channels or a dual-band GNSS receiver that operates across multiple systems. The subsequent section will delve into the existing implementations for every GNSS-SDR processing block and their corresponding configuration.

### **2.4.4 Signal Processing Blocks**

#### Global Receiver Parameters

Global receiver parameters are key parameters that apply across the entire GNSS-SDR system. These include parameters like system architecture, frequency bands, number of channels, and many more. Essentially, these parameters determine the general operation and performance of the receiver.

## Signal Source

The Signal Source block is responsible for reading data from the data source. This data could be read in real-time from a live GNSS antenna, or it could be read from a file that contains previously collected data. The signal source block takes care of reading this data and passing it along to the next block in the chain for further processing.

## Signal Conditioner

The Signal Conditioner represents a subsequent stage in the signal processing continuum. This module comprises a data type adapter, an input filter, and a resampler. The objective of this block is to ready the signal for the ensuing acquisition and tracking phases..

## Data Type Adapter

The Data Type Adapter, as part of the Signal Conditioner, is responsible for converting the input data into a format that can be processed by the subsequent blocks. This may involve converting the data into a specific data type or performing a scaling operation on the data.

## Input Filter

Following the data type adapter, the Input Filter's role is to limit the bandwidth of the incoming signal to reduce noise and interference, ensuring the signal quality is maintained for the next processing stages.

## Resampler

The Resampler adjusts the sample rate of the signal to match the rate required by the downstream processing blocks. This is necessary when the sample rate of the incoming signal doesn't match the rate required by the acquisition and tracking stages.

## Channels

Channels in GNSS-SDR are blocks that independently process a single satellite signal. Each channel includes acquisition, tracking, and telemetry decoding subsystems.

## Acquisition

Acquisition is the initial process in a channel, responsible for detecting the presence of a satellite signal and providing coarse estimates of the code phase and Doppler frequency. This allows the receiver to "lock on" to a specific satellite signal.

## Tracking

The Tracking block refines the estimates provided by the Acquisition block and maintains lock on the satellite signal. It accurately measures the code phase and Doppler frequency and provides these measurements to the Telemetry Decoder.

## Telemetry Decoder

The Telemetry Decoder decodes the navigation message contained in the satellite signal. This information includes the satellite's ephemeris (orbital information) and clock correction parameters, which are required for computing the user's position.

## Observables

The Observables block collects the measurements from all channels and forms the observation equations. These equations contain the pseudorange and carrier phase measurements from each satellite.

## PVT

PVT stands for Position, Velocity, and Time. The PVT block uses the observables from each satellite to calculate the user's position, velocity, and the precise time. This is the final step in the GNSS receiver, and the output of this block is typically the end product of a GNSS receiver.

## Monitor

The Monitor block is responsible for monitoring the performance and status of the receiver. It keeps track of key metrics like signal strength, signal-to-noise ratio, and the number of satellites being tracked. It can also provide alerts or diagnostics if any issues are detected with the receiver's operation.

## **2.5 Methods of Global Navigation Satellite System (GNSS) Spoofing**

### **2.5.1 Introduction to Spoofing**

Global Navigation Satellite Systems (GNSS) are a fundamental component of critical infrastructure across a range of industries, including aviation, telecommunications, maritime, and personal devices. These systems encompass GPS (U.S.), Galileo (EU), GLONASS (Russia), and BeiDou (China). The act of GNSS spoofing, which is deemed both illegal and unethical, entails the fabrication of counterfeit signals with the intention of misleading GNSS receivers with regards to their temporal or spatial coordinates..

## 2.5.2 Methods of Spoofing

### 2.5.2.1 Replay Spoofing Attack Strategies (RSA)

RSA involves capturing authentic GNSS signals and retransmitting them. Variations of this strategy include:

- **Direct Replay Interference:** The attacker records GNSS signals at one location and retransmits those signals at another, causing the receiver to think it's at the original location.
- **High Power Replay Interference:** The intentional retransmission of a previously recorded signal by an attacker at a substantially elevated power level compared to the original signal. This results in the overpowering of the authentic signal and the consequent locking of the receiver onto the spurious signal.
- **Selective Delay Replay Interference:** The attacker selectively delays some retransmitted signals, leading to incorrect positioning calculations by the receiver.
- **Multi-antenna Receiver Replay Interference:** The attacker uses multiple antennas to capture and replay signals, mimicking the spatial distribution of the original signals, fooling receivers that use antenna arrays to validate signals spatially.

In a standard RSA scenario, it is essential to acknowledge that the transmission of a satellite signal to a receiver is subject to an intrinsic propagation delay. The occurrence of a delay presents a potential opening for malicious actors, as it allows for the possibility of either replicating the initial signal or introducing an extra delay in order to carry out a spoofing assault. The reception of a GNSS receiver's conventional satellite signal is influenced by various crucial factors, such as the amplitude ( $A$ ) of the signal, the navigation message ( $D_n(t)$ ), the ranging code ( $C_n(t)$ ), the signal propagation delay ( $\tau_n(t)$ ), the standard carrier frequency ( $f_0$ ), the doppler frequency ( $f_d$ ), and the beat carrier phase ( $\theta_n$ ). The symbol  $I_n(t)$  is utilized to indicate the possibility of interference that

may occur in conjunction with the satellite signal, whereas  $nn(t)$  is indicative of noise.

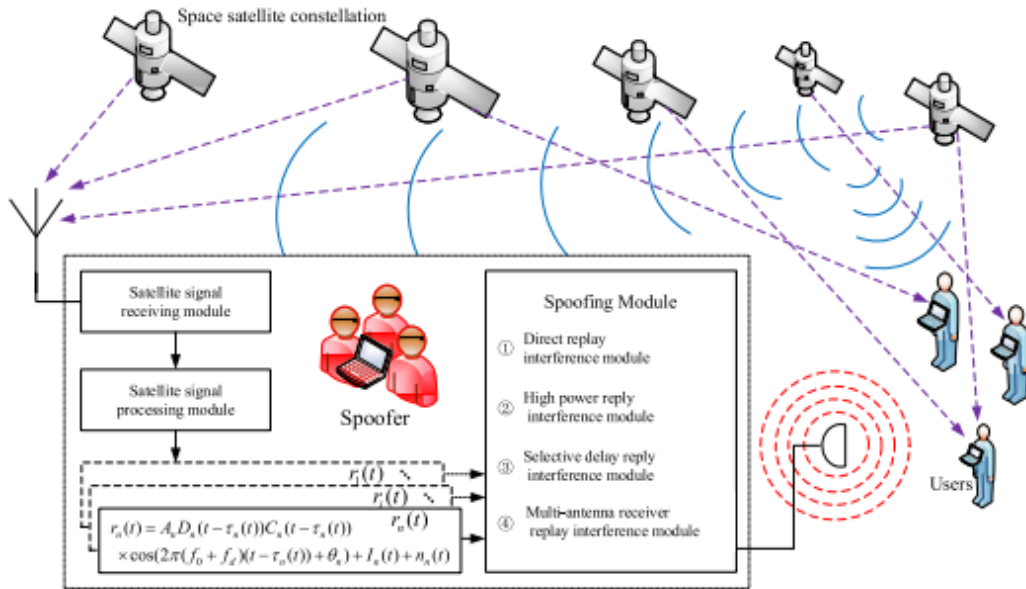


Figure 4 RSA Scenario

Various tactics can be employed by attackers in the context of RSA. To provide an example, Direct Replay Spoofing Interference involves the transmission of the authentic satellite signal directly to the receiver by the spoofer. The method known as High-power Replay Interference involves the artificial amplification of the amplitude of the satellite signal. The Selective Delay Replay Interference technique incorporates the propagation delay of the satellite signal. Multi-antenna receiver replay interference is a technique that employs multiple antennas to replicate a spoofing signal, thereby rendering signal consistency-based detection methods ineffective.

Investigations into these RSA techniques have unveiled fascinating findings. For instance, Huang et al.'s simulation of RSA[1] found that a spoofing signal stronger by 4 dB than the real signal could disturb the authentic signal reception within 50 minutes. In another experiment by Gao et al., it was revealed that a successful spoofing could be accomplished within just 4 seconds if the interference-to-signal ratio exceeded 14 dB.

Furthermore, it has been observed that the interference caused by Selective Delay Replay has a significant impact on timing receivers, thereby presenting greater detection difficulties. Shi and colleagues adopted a noteworthy methodology by utilizing genetic algorithms to optimize the duration of signal delay at various deception points. This approach resulted in improved concealment and reduced complexity as compared to the conventional point-by-point method.

To execute a successful spoofing attack with multiple sources, a perpetrator must employ a combination of delay and power adjustment techniques on intricate satellite navigation receivers. The perpetrator has the ability to intentionally modify the consequences of deceit. Wang et al. suggest that the manipulation of parameters within the inertial navigation system, in conjunction with the satellite navigation system, can be achieved through the gradual modification of the positional impact of the deception signal over an extended duration.

In summary, while the implementation of RSA may appear uncomplicated, the attainment of a prosperous spoofing endeavor necessitates adroit manipulation of the pertinent parameters of the spurious signal. It is noteworthy that certain auxiliary interference techniques employed in RSA may prove advantageous in other forms of spoofing, thereby contributing to the overall efficacy of the spoofing endeavor.

### 2.5.2.2 Forgery Spoofing Attack Strategies (FSA)

FSA involves generating false GNSS signals from scratch, mimicking the properties of a real GNSS signal. This involves generating the correct carrier frequencies, modulating the signal with pseudorandom noise codes, and properly aligning the signal with respect to time. An advanced version of this strategy, Full Channel Forgery, mimics all aspects of a GNSS signal, including signals from multiple satellites, multiple frequencies, and signals that appear to come from different directions.

The FSA model comprises three pivotal modules: the satellite signal receiving module, the spoofing signal generating module, and the spoofing signal transmitting module. Each module's functionality is integral to the operation and success of the FSA.

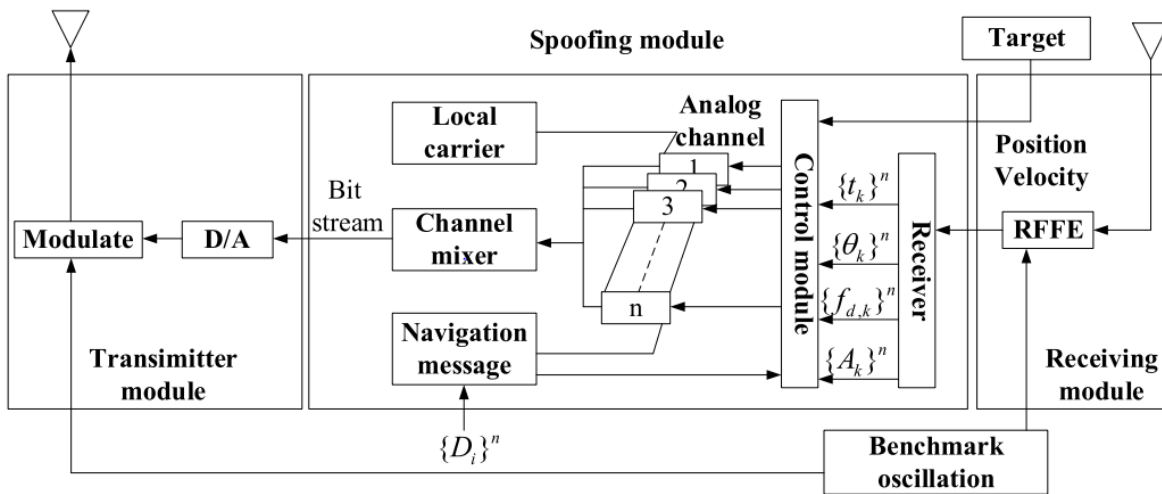


Figure 5 Schematic Diagram of FSA

The Satellite Signal Receiving Module obtains the authentic satellite signal through the antenna and transmits it to the receiver of the spoofing signal generator via the radio frequency front end (RFFE). Simultaneously, the spurious receiver observes the intended subject, acquiring its positional and speed information.

There are two ways the Spoofing Signal Generating Module functions. In the first case, a technique known as direct-generation forgery is used by the receiver to produce fake satellite signals.

However, the generated signals frequently show observable differences from the real ones. In the second situation, the spoofer performs frequency conversion on the received signal to create a signal that closely resembles the real one. Demodulation yields a baseband signal that contains information about the original satellite signal, allowing for the fabrication of a fake signal—a technique known as analysis-generation forgery.

The baseband signal processing enables the spoiling receiver to calculate four parameters. The act of increasing the volume of the spoofing signal, commonly referred to as Denial Environment Forgery, is a technique frequently employed by the spoofer to enhance the efficacy of the deception. Achieving a Full-Channel Forgery that is effective requires complete synchronization between the spoofing signal post-control module and the entire capture and tracking loop of the receiver being targeted. The N spoofed channels produce signals that are indistinguishable from the channel parameters associated with the signals monitored by the receiver module.

High-power replay interference and selective delay replay interference, discussed in the RSA context, are also applicable in FSA. The Direct-Generation FSA dates to Scotta's experiment [2], where he employed a GSS8000 GNSS simulator from Spirent Company to develop a viable signal spoofing apparatus. Still, the resultant signal lacked the relevant parameters of the authentic signal, rendering it detectable.

### **2.5.2.3 Estimation Spoofing Attack Strategies (ESA)**

ESA involves estimating certain parameters of the GNSS signal and creating a false signal based on those estimates. Security Code Estimation and Replay involves estimating the security codes used by some GNSS signals for authentication, and then generating a false signal that appears authenticated to the receiver. Forward Estimation Attack involves predicting future GNSS signals based on previously received signals, and then creating a false signal that aligns with this prediction.

ESA is a method where the spoofer leverages estimations of received navigation signals to perform the deception process. This is particularly relevant when anti-spoofing measures are implemented, such as inserting an unknown security code into the navigation message to enhance its security. Since the spoofer cannot predict the security code, they cannot generate a recognizable navigation message, thus necessitating the estimation of received navigation signals.

Security Code Estimation and Replay (SCER):

Professor Humphreys first suggested the SCER assault, a particular kind of ESA [3], [4]. The primary objective of this approach is to estimate security codes in navigation messages produced by an encryption algorithm. Once the security codes have been properly approximated, the spoofer creates spoofing signals using genuine satellite signal characteristics. The exact management of artificially induced delay is essential to this operation. A representation of SCER attack is:

$$SCER = f(Delay, Code Offset, Carrier Phase)$$

Where:

- Delay represents the artificial delay introduced by the spoofer
- Code Offset refers to the estimated code offset of range code
- Carrier Phase refers to the estimated carrier phase.

While the SCER attack provides a robust method for spoofing, its complexity and requirement for extensive knowledge about navigation messages, navigation signals, and signal estimation methods make it less commonly employed in actual projects.

#### Forward Estimation Attack (FEA)

The FEA is an advancement in ESA that is more recent [32]. The majority of receivers do not validate the navigation message before decoding, therefore the spoofer can trick the receiver by generating a bogus navigation message and combining it with the available previous information. In contrast to SCER, FEA does not call for the gathering of real signals before the spoofing procedure. As an alternative, the spoofer can send false information before the real information. A simple representation of FEA can be formulated as:

$$FEA = g(\textit{Previous Information}, \textit{Fake Information})$$

Where:

- Previous Information refers to the intrinsic relevance of the navigation message.
- Fake Information is the false navigation message generated by the spoofer.

It's worth noting that the FEA attack was tested via a simulated attack on a Galileo signal with Navigation Message Authentication (NMA) [5]. The results indicated that NMA was unable to perform its authentication function under FEA. However, adding anti-replay information to the navigation message can partly resist an FEA attack.

Both SCER and FEA attacks represent significant security concerns for modern navigation systems. Although these attacks are challenging to implement, and thus not commonly adopted by spoofers, they could become more potent should future cryptographic anti-spoofing methods be implemented. This report underscores the critical need for ongoing research and development of robust anti-spoofing techniques, specifically those that consider the resistance of SCER and FEA attacks, to ensure the security and reliability of global navigation satellite systems.

#### 2.5.2.4 Advanced Spoofing Attack Strategies (ASA)

ASA includes methods like Nulling Attack and Cooperative Interference Attack. The nulling attack uses an array of antennas to generate signals that cancel out the signals from real GNSS satellites, causing the receiver to lock onto false signals. The cooperative interference attack involves multiple attackers each transmitting a signal that, when combined, creates a powerful interference leading to disruption or false signals.



Each approach requires a high level of technical proficiency, and the complexity of execution is mostly influenced by the hardware and technical costs of the spoofing strategy. The harder it is to do, the higher the hardware's technical cost. Despite their sophistication, these techniques seriously endanger the reliability of GNSS systems and the industries that depend on them.. Therefore, it is crucial to develop and implement robust anti-spoofing techniques to safeguard these.

#### Nulling Attacks:

A nulling attack, as proposed in recent studies [6], is a sophisticated spoofing method designed to tamper with genuine satellite information, thus manipulating the target receiver's positioning and timing. This technique includes creating two nulling signals, the second of which destroys the receiver's actual signal and the first of which serves as a spoofing signal, altering the genuine satellite information that was received. The receiver pulls in just the initial nulling signal, ignoring the real signal, to pull off this deceit..

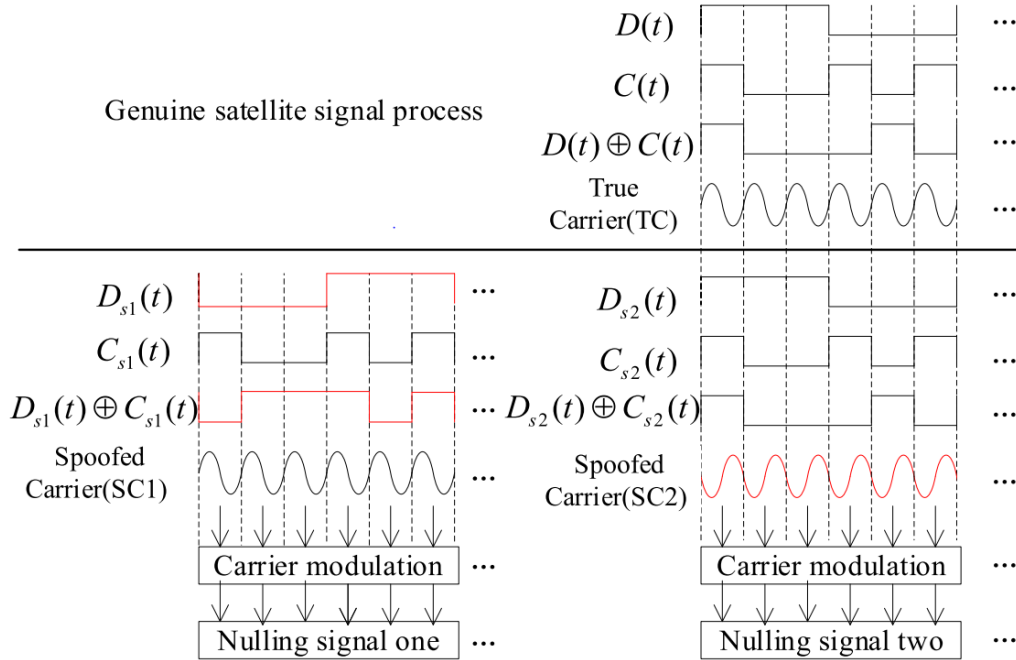


Figure 6 Nulling Attack Signal Diagram

The GNSS signal captured by the receiver can be expressed as:

$$R(t) = \text{Re}\{\sum_{i=1}^N A_i D_i[t - \tau_i(t)] C_i[t - \tau_i(t)] e^{j[\omega_c t - \phi_i(t)]}\}$$

Where:

- $N$  is the number of constituents spreading – code specific signals
- $A_i$  is the amplitude of the signal
- $D_i$  is the data of the signal

- $C_i$  is the PN code of the signal
- $\tau_i(t)$  is the code phase of the signal
- $\varphi_i(t)$  is the beat carrier phase of the signal

During the nulling process, signal one and signal two must satisfy the following relationship:

$$C_{i+N}(t) = C_i(t) \text{ and } D_{i+N}(t) = D_i(t) \text{ for } i = 1, \dots, N$$

Furthermore, the nulling signal obeys:

$$A_{s[i+N]} = A_i, \tau_{s[i+N]} = \tau_i(t) \text{ and } \varphi_{s[i+N]} = \varphi_i(t) + \pi$$

Despite the potential for high spoofing success rates, nulling attacks are complex and, to date, exist primarily as theoretical means of interference.

Cooperative Interference Attacks:

Cooperative interference attacks aim to stop anti-spoofing receivers like RAIM receivers from operating normally. A cooperative interference assault was used by Ledvina et al. [56] to attain sub-centimeter precision in three-dimensional deception. A complicated anti-spoofing technique, such as one based on predicted angle of arrival, can be defeated by such an assault. Their execution is difficult, nevertheless, and necessitates paying close attention to the surroundings.

Implications and Countermeasures:

While most advanced spoofing attacks remain in the realm of theory, the continuous advancement of computer technology and hardware implies that these theoretical models could potentially be realized in practical applications in the future. Consequently, during the development of anti-spoofing systems, it is imperative to consider diverse spoofing techniques.

Despite their complexities, both nulling attacks and cooperative interference attacks illustrate the evolving landscape of spoofing threats and underline the importance of continued research and development in securing navigation systems.

With the continual evolution of spoofing attacks, the need for robust and flexible countermeasures has never been more critical. Understanding advanced spoofing techniques such as nulling attacks and cooperative interference attacks provides a valuable foundation for developing effective anti-spoofing solutions. While these attacks currently exist primarily in theory, technological advancements could potentially turn them into practical threats, emphasizing the importance of proactive consideration and countermeasure development.

## 2.6 Anti-Spoofing Techniques

## 2.6.1 Introduction

As satellite navigation systems become more integral to many areas of life and industry, the threat of spoofing attacks - where malicious entities generate false signals to deceive navigation systems - grows correspondingly. To mitigate these risks, researchers worldwide have developed various techniques for detecting and suppressing spoofing signals. This report aims to provide a comprehensive overview of the anti-spoofing techniques based on signal-level, emphasizing two main categories: those that necessitate additional hardware facilities and those that do not.

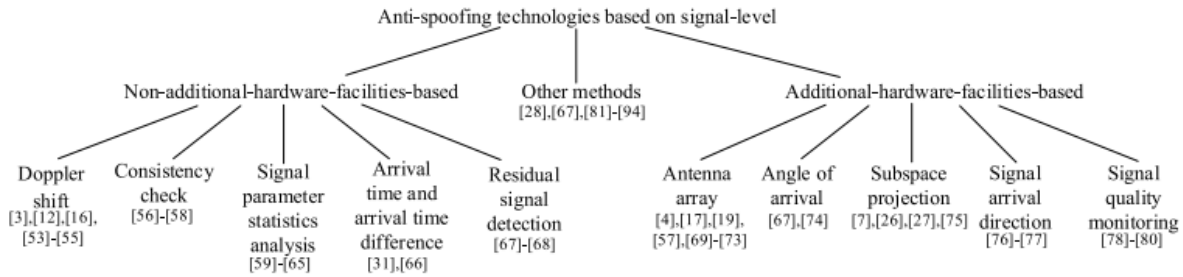


Figure 7 The categories of anti-spoofing technologies based on signal level [35]

## 2.6.2 Methods Independent of Additional Hardware Facilities

Most non-hardware anti-spoofing methods primarily depend on signal metrics such as carrier-to-noise ratio and signal strength for the identification of spoofing signals. This category comprises five additional subcategories, namely: Doppler shift-based methods, consistency checks, analysis of signal parameter statistics, examination of arrival times and arrival time differences, and identification of residual signals.

### 2.6.2.1 Doppler Shift-Based Methods

The techniques employed utilize the Doppler effect, a phenomenon characterized by the alteration in the frequency or wavelength of a wave due to the motion of an observer relative to the wave's source. Disparities in Doppler shift that are unanticipated during a transmission could potentially indicate the presence of spoofing. The phenomenon of the Doppler shift refers to the alteration in the frequency of a wave due to the relative motion between the wave source and an observer. In a satellite navigation system, the receiver located on the target is referred to as the observer, whereas the satellite is considered as the source. In the event that the Doppler shift value of a received satellite signal surpasses a predetermined threshold, it may indicate the possibility of a spoofing endeavor targeting the receiver.

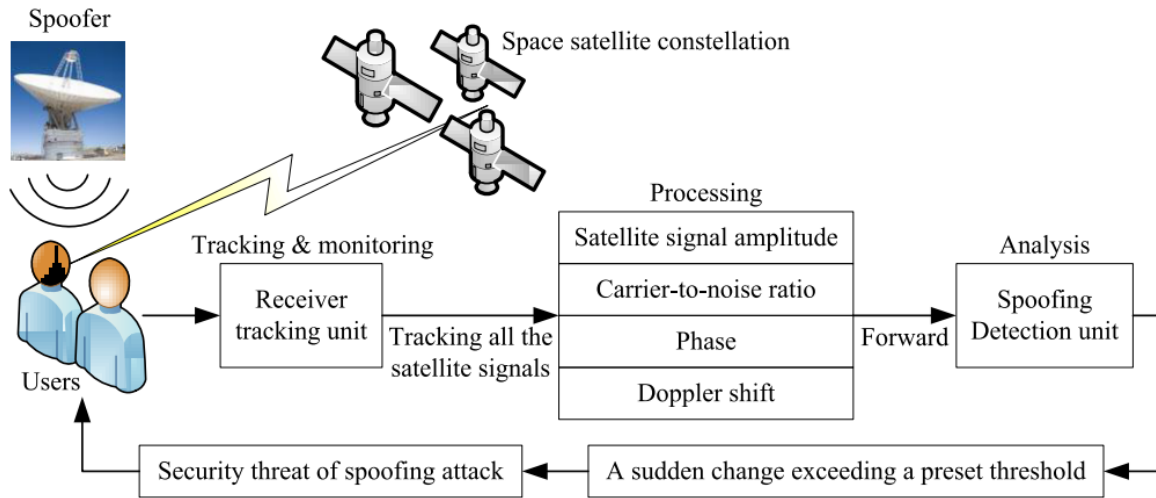


Figure 8 Spoofing detection scenario based on Doppler shift.

Mathematically, the Doppler shift ( $\Delta f$ ) is given by the formula:

$$\Delta f = f_0 * ((v + v_o)/(v - v_s))$$

where:  $f_0$  is the emitted frequency,  $v$  is the wave speed,  $v_o$  is the speed of the observer (receiver), and  $v_s$  is the speed of the source (satellite).

Given that both the wave speed ( $v$ ) and the emitted frequency ( $f_0$ ) are known constants for a satellite signal, a sudden, unexplainable change in the observed frequency (indicating a change in either  $v_o$  or  $v_s$ ) could indicate a spoofing attack.

### Case Studies in Doppler Shift-Based Spoofing Detection

1. Adaptive Tracking Algorithm (Jovanovic [7]): Jovanovic et al. initially introduced the concept of an adaptive tracking approach, which involved the utilization of a Power Threshold Detector (PTD) and a Doppler Offset Detector (DOD) to detect variations in signal power and carrier Doppler shift, respectively. The algorithm performs statistical checks on these variables, enabling the efficient detection of a replay attack.
2. Carrier Frequency Variation Analysis (Qi et al. [8]): The study conducted by Qi et al. involved an examination of the carrier frequency variation process obtained from a phase-locked loop of both a static and dynamic receiver. The researchers proposed a GNSS anti-spoofing technique that relied on the Doppler shift. Through the comparative analysis of frequency variations under distinct conditions, namely the reception of solely authentic signals versus the reception of both authentic and counterfeit signals, this approach effectively identifies instances of spoofing attacks.
3. Code and Carrier Doppler Shift Joint Consistency Check (Yuan et al. [9]): The authors, Yuan et al., have presented a technique that involves the computation of the code and carrier Doppler shift for both the authentic and counterfeit signals during the pre-capture phase. This approach enables a comprehensive verification of the code and carrier Doppler shift through a joint

consistency check. This approach has demonstrated efficacy in discriminating between authentic signals and those generated by spoofing.

4. Frequency-Domain Bimodal and Relative Velocity Residuals Technique (Tu et al. [10]): A spoofing detection method based on relative velocity residuals and frequency-domain bimodality was proposed by Tu et al. The method separates spoofing scenarios from multipath scenarios in addition to detecting spoofing. In order to establish if the doublet was faked, they employed a Fast Fourier Transform to extract the doublet's Doppler difference and detect the doublet.

### 2.6.2.2 Consistency Check-Based Methods

These techniques involve cross-verifying information from various sources or systems. Any inconsistencies detected may indicate a spoofing attack. The basic premise of consistency check-based spoofing detection lies in the expectation that, under normal circumstances, the signal parameters of a navigation satellite signal and the navigation information it carries remain consistent. In a spoofing scenario, these parameters can exhibit abnormal changes or even exceed normal limits, thus indicating a potential attack.

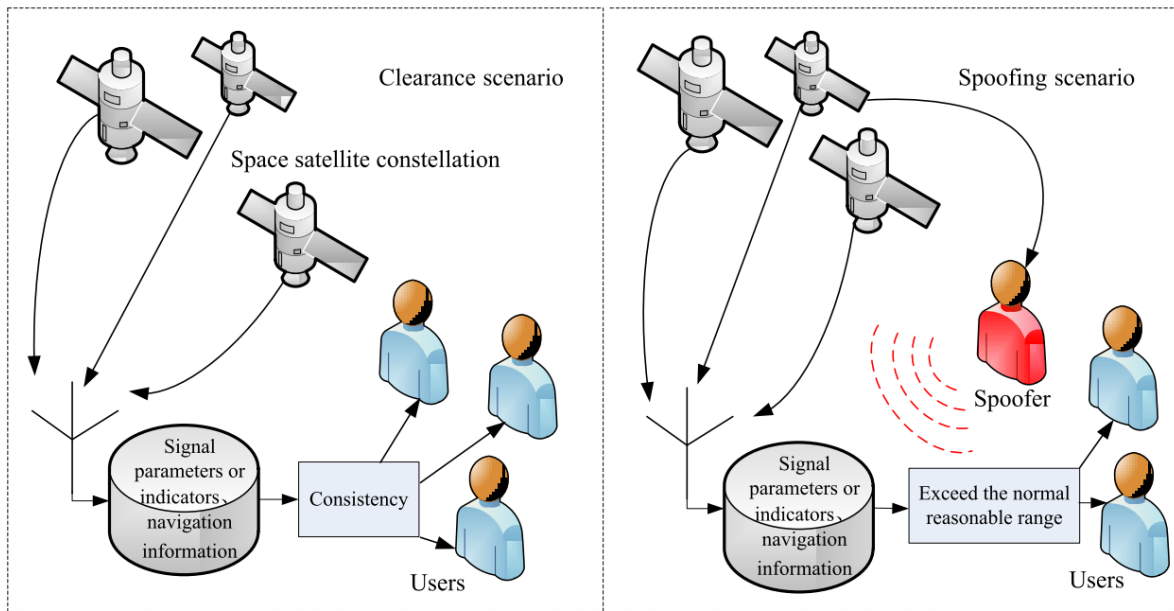


Figure 9 Spoofing and Spoofing detection scenario based on consistency check.

In the context of distinguishing between a single antenna spoofing signal and an authentic satellite signal, it is possible to discern a mapping characteristic of a region to a singular point on a coordinate system. This attribute can be utilized for the purpose of detecting spoofing.

1. Multi-Node Information for Spoofing Detection (Yao et al. [11]) The information consistency of signals received at various observation locations within a monitoring region is used by Yao et al.'s suggested spoofing detection technique. The method gathers data from several nodes to enhance the efficacy of spoofing detection. The idea behind this is that while a single receiver may

be tricked by a spoofing signal, it would be difficult for it to concurrently deceive several receivers dispersed over a wide region without displaying discrepancies.

2. Authentication by Receiver Mobile Antenna (A. Broumandan [12]): This technique uses a receiver mobile antenna to verify the signal. During the tracking phase, it identifies a strong correlation between the amplitude, phase, and Doppler variations and the mobile antenna of the receiver's channel response. With this technique, spoofing signals can be found at a level that the mobile receiver can see during the navigational phase.

3. Power and Distortion Detection Technology (Wesson et al. [13]): Wesson . The proposed spoofing detection technique aims to differentiate interference-free, multipath interference, and low-power spoofing scenarios through the identification of power and distortion features present in the satellite signals. This approach provides an additional layer of security by analyzing the physical characteristics of the signal, which can vary significantly in instances of spoofing.

4. Network-Based or Cloud-Based Verification Method ([14], [12]): This method assumes that receivers share measurement data through a low-rate communication link or cloud storage, enabling the separation of the carrier phase double difference between real and spoofing signals to detect spoofing attacks

Analysis of Consistency Check-Based Spoofing Detection Methods:

Consistency check-based methods have shown effective results in stable spoofing scenarios. However, they may struggle in dynamically changing or multi-antenna spoofing environments. The methods proposed in [35] tackle this problem by incorporating signal amplitude and phase change correlation and network or cloud-based verification for satellite signal authenticity. These methods utilize the receiver's different processing stages and spatial features, enhancing their adaptability to various spoofing scenarios.

Table 1 and Table 2 highlight the efficacy of these techniques in countering various forms of spoofing attacks on resistance performance. The detection capability of the method suggested in reference [36] is commendable for detecting forwarding spoofing attacks, however, it encounters difficulties in efficiently eliminating the spoofing signal.

Schemes	Selective power spoofing	Selective delay spoofing	Nulling attack	Denial environment
Spoofing detection method based on information consistency check [61]	Moderate	Moderate	Poor	Poor
Spoofing detection method based on correlation of signal amplitude and phase change [62]	Good	Moderate	Good	Moderate
Satellite signal authenticity verification method based network or cloud [62]	Moderate	Good	Poor	Moderate
Power and distortion monitoring technology [63]	Moderate	Good	Poor	Poor

Table 1 Comparison of anti-spoofing method based on consistency check.

Position consistency checks are crucial in anti-spoofing techniques for GNSS. By validating the consistency of the receiver's position over time, it is possible to identify abnormalities that might suggest a spoofing attack.

1. Static Position Check: The simplest type of position consistency check is a static position check. If the receiver is stationary, any significant changes in the reported position could indicate a spoofing attack. This can be formalized mathematically as follows:

Let  $p(t)$  represent the position reported by the receiver at time  $t$ . If the receiver is stationary, we expect.

$$p(t) = p(t + \Delta t)$$

for all  $\Delta t$ .

$$\text{If } |p(t + \Delta t) - p(t)| > \varepsilon$$

for some small  $\varepsilon$  and some  $\Delta t$ , a spoofing attack could be in progress.

2. Position Jumps: Even if the receiver is moving, we expect the position to change smoothly over time. If the reported position suddenly jumps by a large amount, this could indicate a spoofing attack. Mathematically, we can represent this as follows:

Let  $v(t)$  be the velocity of the receiver at time  $t$ . The position at time  $t+\Delta t$  should be approximately  $p(t) + v(t)\Delta t$ .

$$\text{If } |p(t + \Delta t) - p(t) - v(t)\Delta t| > \delta$$

for some small  $\delta$  and some  $\Delta t$ , a spoofing attack might be taking place.

3. **Velocity Consistency:** In addition to checking the position, we can also check the consistency of the receiver's velocity. A sudden change in velocity could indicate a spoofing attack. Formally, we can express this as:

Let  $a(t)$  be the acceleration of the receiver at time  $t$ . The velocity at time  $t+\Delta t$  should be approximately  $v(t) + a(t)\Delta t$ .

$$\text{If } |v(t + \Delta t) - v(t) - a(t)\Delta t| > \zeta$$

for some small  $\zeta$  and some  $\Delta t$ , a spoofing attack could be suspected.

4. **Abnormal Position Check:** Finally, we can check for positions that are simply impossible or highly unlikely. For example, if the reported position is outside the Earth's surface or is moving at a speed greater than the speed of light, a spoofing attack is almost certainly in progress.

Mathematically, let's denote the boundaries of a possible position space as  $P\_space$ . If at any given time  $t$ ,  $p(t)$  not in  $P\_space$ , it strongly suggests a spoofing attack.

Each of these checks helps to detect inconsistencies that can indicate a spoofing attack. However, none of them can definitively prove that a spoofing attack is in progress. Environmental factors such as multipath propagation or rapid changes in the receiver's velocity can also cause similar inconsistencies. Therefore, these checks should be used in conjunction with other anti-spoofing techniques for the best results.

#### 5. **Clock Consistency**

The maintenance of clock consistency is a crucial element in the implementation of anti-spoofing techniques for Global Navigation Satellite Systems (GNSS). The objective is to guarantee the veracity of received signals through an evaluation of the coherence between the drift of the receiver clock and the computed Coordinated Universal Time (UTC) information. Comprehending the



mechanism of clock consistency necessitates a comprehension of the interplay between the receiver clock and satellite signals within the Global Navigation Satellite System (GNSS).

Method	Characteristics	Defect	Required configuration	Implementation phase	Implementation difficulty
Spoofing detection method based on phase variance analysis [64]	The scheme uses the generalized likelihood ratio test method to derive decision statistics and provides approximate criteria for setting decision thresholds.	The spoofing detection performance for the method is poor when the maximum value of the available phase value is small.	Receivers need to add dual antenna system.	Capture	Medium
Spoofing detection method based on square sum [65]	The decision statistic is calculated using spatially separated receiver carrier phase measurements. The scheme derives detector classification method based on generalized likelihood ratio test and models random variables	In some spoofing environments, the spoofing detection performance for the method is poorly reliable.	The receiver needs to configure the antenna array.	Capture and track	Low
Spoofing signal detection method based Post-correlation [66]	The scheme considers static scenarios and dynamic scenarios. After appropriate improvement of the method, multipath interference and deception interference can be distinguished.	The method has a certain complexity, and the spoofing detection is ineffective in the spoofing scene with high real-time requirements.	Receivers need to have post-related inspection capabilities.	Capture	High
Receiver autonomous signal authentication method [67]	The scheme detects whether spoofing exists according to the estimated clock stability of the receiver.	This method needs to combine other detection methods or means to ensure the authenticity of the measured values.	Target receiver is moving.	Capture and track	Low
Spoof detection method based on sequence probability ratio test [68]	The scheme detects spoofing based on the relationship among the average spoofing detection time, the probability of spoofing detection, the false alarm probability, the signal-to-noise ratio, and the ratio of spoofing to the true signal.	The detection performance for the power of spoofing signal similar to the power of the real satellite signal is affected to some extent, and combined detection with other detection methods is also required.	Receivers need to have sequence probability ratio test capability.	Capture	Medium
Anti-spoofing method based on ARPSO-MLE [69]	The scheme uses particle swarm optimization and spatial interaction generalized expectation maximization algorithm (SAGE) for multimodal optimization.	This method is less stable against spoofing attacks at lower signal-to-noise ratios.	Receiver with ARPSO-MLE based position estimation capability.	Capture	Medium
PD-ML detector technology [70]	The scheme can accurately identify spoofing in the environment where satellite signals and interference signals coexist.	This method increases computational complexity and technical cost, lack of sensitivity assessment for applications with narrow front-end bandwidth receivers.	The maximum likelihood multipath interference estimator is required inside the receiver.	Track	High

Table 2 Analysis and comparison for spoofing detection methods based on static Analysis of signal Parameters.

## The Role of the Receiver Clock

The receiver's clock is crucial to this process. However, maintaining a high-precision atomic clock, like the ones onboard GNSS satellites, is impractical for most consumer devices. Instead, these devices use a less accurate local clock and include the clock offset as an additional unknown in the position calculation.

## Understanding Clock Drift and Offset

Even with a high-quality local clock, small inaccuracies can accumulate over time, a phenomenon known as clock drift. This drift must be accounted for and corrected regularly. The receiver estimates the clock offset from the GNSS time as part of the position solution. If the receiver is stationary, the clock offset should be relatively stable over time.

## Clock Consistency as an Anti-Spoofing Measure

In a GNSS spoofing attack, the attacker might transmit signals with manipulated timing information, causing the receiver to calculate incorrect position or time. A sudden or significant change in the estimated clock offset might indicate such an attack. The receiver can monitor the variance of the clock offset over time and flag any anomalies.

To formalize this, let's denote  $c(t)$  as the clock offset at time  $t$ . If the receiver is stationary or moving at a constant speed, we expect  $c(t)$  to be relatively constant.

$$\text{If } |c(t + \Delta t) - c(t)| > \varepsilon$$

for some small  $\varepsilon$  and some  $\Delta t$ , this could indicate a spoofing attack.

Another component of clock consistency involves comparing the calculated UTC time with a reliable external source. If the receiver's UTC time deviates significantly from the external source, it may be under a spoofing attack.

Mathematically, if the UTC time calculated from GNSS is  $T_{GNSS}$  and the UTC time from a reliable external source is  $T_{ext}$ , a significant difference.

$$|T_{GNSS} - T_{ext}| > \eta$$

for some small  $\eta$ , could suggest a spoofing attack.

In summary, clock consistency as an anti-spoofing measure focuses on monitoring and detecting anomalies in the receiver's clock behavior and comparing calculated UTC with a trusted external time source. Despite being a strong defense mechanism, it isn't foolproof. As such, it should be used alongside other anti-spoofing techniques for comprehensive protection against GNSS spoofing.

### 2.6.2.3 Signal Parameter Statistics Analysis-Based Methods

These techniques rely on analyzing various signal parameters' statistical properties. Anomalies that deviate from the expected statistical norms may suggest a spoofing attack.

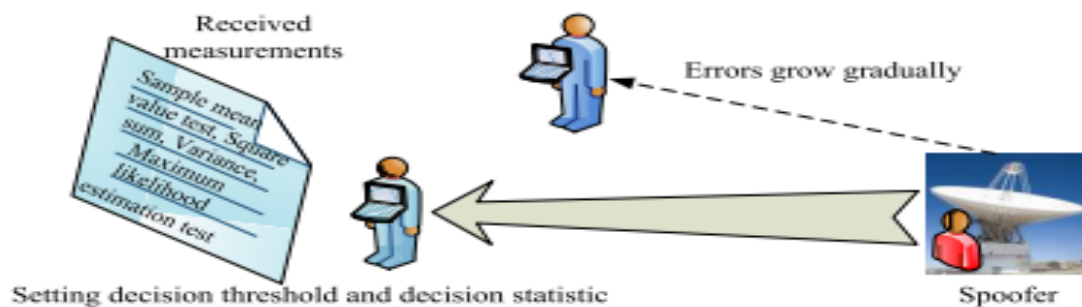


Figure 10 Spoofing detection scenario based on signal parameter statistical analysis

1. Phase Variance Analysis Test Method (Borio [15]):

In 2013, Borio introduced a new spoofing detection approach using a phase variance analysis test, known as PANOVA. This method uses the differences in sample mean values and phase spatial characteristics of satellite signals to detect spoofing attacks. The principle lies in the premise that spoofing signals may display significant deviations in the phase variance compared to genuine signals.

Equation for PANOVA:

$$Var(\Phi) = E[(\Phi - E[\Phi])^2]$$

where  $\Phi$  denotes the phase and  $E$  denotes the expected value operation. Variance,  $Var$ , calculates how far each number in the set is from the mean ( $E[\Phi]$ ) and squared.

2. Sum-of-Squares (SoS)-Based Angle-of-Arrival Spoofing Detection Method (Borio and Gioia [16]):

Borio and Gioia introduced a spoofing detection approach based on System of Systems (SoS) in 2016, which employs the Generalized Likelihood Ratio Test (GLRT) methodology. The square sum detector was developed utilizing the generalized likelihood ratio test (GLRT), which is a statistical hypothesis test employed to ascertain the most probable value of a parameter under the null hypothesis. The detection of nulling attack spoofing and selective power spoofing during receiver acquisition and tracking can be achieved with reduced implementation challenges.

The GLRT function is:

$$GLRT = argmax_{\theta_0} L(\theta_0 | x) / max_{\theta_1} L(\theta_1 | x)$$

where  $L$  denotes the likelihood function,  $\theta_0$  and  $\theta_1$  represent the parameters under null and alternative hypothesis, and  $x$  denotes the observed data.

3. Cross Ambiguity Function Matrix Check:

Post-Correlation Based Spoofing Detection Method (Falletti et al.), This approach, which has been validated through multiple static and dynamic field experiments, utilizes post-correlation techniques to detect and isolate spoofing signals. In instances where spoofing attacks are detected, the affected navigation satellites are removed in order to mitigate the potential for erroneous impacts on navigation outcomes. The Cross-ambiguity function matrix (CAF) verification is a complex technique employed for the purpose of detecting anti-spoofing in Global Navigation Satellite System (GNSS) applications. The underlying principle involves the identification of multiple peak values within the correlation function, which may suggest the presence of multiple received signals (both authentic and fraudulent)..

The evaluation of the cross-ambiguity function matrix (CAF) is based on the cross-ambiguity function, which serves as a metric for assessing the degree of resemblance between two signals with respect to temporal displacement and frequency deviation. Within the context of Global Navigation Satellite Systems (GNSS), this particular function is frequently employed to detect and ascertain the presence of counterfeit signals through a comparative analysis of the received signal and an anticipated genuine signal.

The CAF is given by the equation:

$$\Gamma(\tau, f) = \int s_1(t) * s_2(t + \tau) * e^{(-j2\pi ft)} dt$$

where:

- $\Gamma(\tau, f)$  is the cross – ambiguity function,
- $\tau$  is the time delay,
- $f$  is the frequency offset,
- $s_1(t)$  and  $s_2(t)$  are the received and expected signals respectively,
- $j$  is the imaginary unit, and
- $t$  is time.

In the context of anti-spoofing, the CAF check involves comparing the cross-ambiguity functions of the received signal with expected signals for each satellite in view. If a spoofing attack is ongoing, it is likely that there will be a significant difference in the CAF for the spoofed signal due to the presence of multiple correlation peaks.

For example, if a spoofing device is transmitting signals pretending to be multiple satellites, each with a slightly different time delay and/or frequency offset, this would result in multiple peaks in the CAF. By contrast, in the absence of spoofing, the CAF for each satellite signal would be expected to have a single peak, corresponding to the direct signal path from the satellite to the receiver.

After calculating the CAF, the presence of multiple peaks can be determined by a peak detection algorithm. If multiple peaks are detected in the CAF for a satellite signal, this can be an indication of a spoofing attack.

#### 4. Receiver Autonomous Signal Authentication Method (Hwang et al. [17]):

This methodology expeditiously assesses the clock stability of the receiver by utilizing its estimated clock state Allan variance and determines the presence of dynamic spoofing, which is instigated by the relative motion between the GNSS receiver and the spoofing source.

#### 5. Sequence Probability Ratio Test Based Spoofing Detection (Yuan et al. [18]):

This method doesn't require a predetermined number of observations, potentially reducing the number of required observations compared to other methods.

#### 6. Attractive and Repulsive Particle Swarm Optimization (ARPSO) and PD-ML Detector (Wang et al. [19], Gross et al. [20]):

The present methods concentrate on augmenting the Maximum Likelihood Estimation (MLE) technique, which exploits the coherence among diverse navigation satellites to mitigate the impact of spoofing assaults. The ARPSO technique is designed to mitigate the issue of premature convergence, whereas the PD-ML detector enhances the ability to detect spoofing in environments with multipath interference.

### 2.6.2.4 Arrival Time and Arrival Time Difference-Based Methods

These techniques assess the time of signal arrival. Significant differences in arrival times or expected arrival time patterns can indicate a spoofing attack. Several techniques have been proposed for detecting and mitigating these spoofing threats, among which the arrival time and arrival time difference-based methods stand out due to their inherent efficacy. This report dives into the details of these techniques, reviewing the principles, methods, and effectiveness in detecting spoofing activities.

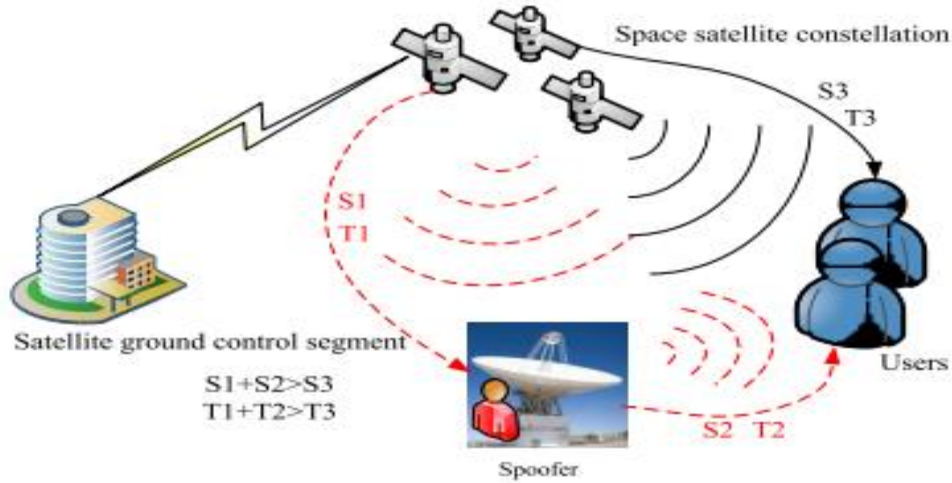


Figure 11 Spoofing detection principles based on arrival time and arrival time difference.

1. Principle of Arrival Time and Arrival Time Difference Detection:

As depicted in Figure 11, it can be observed that the transmission distance and time between the real satellite and the receiver (S3, T3) is comparatively shorter than the direct transmission distance and time between the real satellite and a forward-type spoofing source, followed by the transmission to the target receiver (S1, T1 + S2, T2). This phenomenon occurs due to a latency in the transmission of the forward-style spoofing signal, which results in its delayed arrival at the receiver. This methodology utilizes temporal discrepancies among satellite signals that are received at the phase center of the antenna. Identifying the origin of spoofing, rather than solely detecting its presence, is imperative in mitigating spoofing. Precisely discerning the origin of spoofing facilitates the efficient elimination of said spoofing. The process of identifying the origin of spoofing typically hinges on the estimation of time difference of arrival (TDOA), a metric commonly determined through the evaluation of signal cross-correlation..

2. Spoofing TDOA Estimation Method Based on Differential Code Phase (DCP) (Zhang et al. [21]):

Zhang et al. proposed a TDOA estimation method using differential code phase. The differential code phase is obtained by subtracting the pseudo ranges from different receivers, as described by the equation:

$$DCP = PR1 - PR2$$

where PR1 and PR2 are pseudo ranges of the satellite signal from two different receivers.

3. Principle of Power level Detection Method:

This approach pertains to scenarios of intermediate spoofing, wherein the strength of the spurious signal is marginally greater than that of the authentic signal. In circumstances of this nature, anti-spoofing methods that rely on power detection exhibit inadequacy. The determination of the peak value of the tracking signal acquired by the satellite signal acquisition module was proposed by Li. The proposed methodology employs a multimodal detection approach to identify potential spoofing signals by detecting an excessive number of threshold correlation peaks within any given signal interval.

In GPS and similar systems, the power level of the signals received from different satellites is generally constant, given a fixed receiving antenna gain pattern and no significant changes in atmospheric conditions. This is because all satellites in the system, like GPS, are set to transmit their signals at the same power level and the variations due to distance and relative speed (Doppler effect) are predictable.

In contrast, spoofed signals often exhibit significantly different power levels, as attackers might increase the signal strength to dominate the receiver's correlators or try to mimic the power variations to look more believable. Therefore, a large and/or inconsistent change in power level can be a strong indication of a spoofing attack.

Mathematically, the received power  $P_r$  of a signal is given by the Friis transmission equation:

$$P_r = P_t + G_t + G_r + 20\log_{10}(\lambda/4\pi d)$$

where:

- $P_t$  is the transmitted power,
- $G_t$  and  $G_r$  are the gains of the transmitting and receiving antennas, respectively,
- $\lambda$  is the wavelength,
- $d$  is the distance between the transmitter and receiver.

For an authentic GPS signal,  $P_t$ ,  $G_t$ , and  $\lambda$  are constant.  $G_r$  is fixed for a specific antenna, and  $d$  changes slowly and predictably as the satellite moves. Hence, under normal conditions, the received power  $P_r$  remains fairly constant or changes predictably.

In a spoofing scenario, an attacker, being much closer to the receiver, transmits a signal with power  $P_s$ . If  $P_s$  is much larger than the expected  $P_r$ , it is a clear indication of spoofing.

### 2.6.2.5 Residual Signal Detection-Based Methods

These techniques work by detecting residual signals left by spoofers. The presence of these residuals may indicate an ongoing or attempted spoofing attack. In these attacks, false signals are broadcasted to deceive GNSS receivers, leading to inaccurate navigation information. An effective approach to mitigating such threats is the use of Residual Signal Detection (RSD). This report details the principles, techniques, and performance of RSD methods in detecting and countering GNSS spoofing.

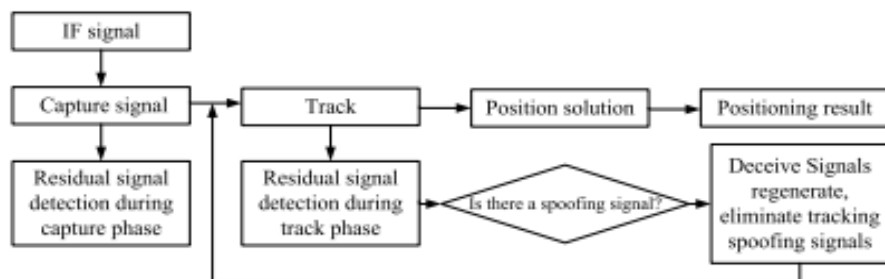


Figure 12 Residue signal detection block diagram

1. The Principle of Residual Signal Detection (RSD) is based on the notion that it is challenging for a spoofer to completely eliminate the authentic satellite signal while conducting a spoofing attack. The residual component of the actual signal present in the received signal can be utilized for the purpose of detecting the existence of a spoofing signal. During the phases of receiver acquisition and tracking, residual signal detection is executed to detect any potential spoofing signals that may be present in the received signals during startup and in real-time, respectively.
2. The residual signal detection (RSD) technique can be applied during the capturing and tracking phase to identify any counterfeit signals and obtain signal characteristics for the purpose of regenerating and eliminating the counterfeit signals if the receiver can still differentiate between the authentic satellite signal and the counterfeit signal.
3. The method proposed by Ali [22] integrates joint signal quality monitoring techniques and residual signal monitoring to enhance the effectiveness of spoofing detection. The utilization of a duo of supplementary correlators and two indicators founded on the ratio metric is implemented to differentiate between the corruption of the correlation function that is caused by the presence of multipath interference and the act of spoofing. The evaluation of the correlation function's quality is necessary to identify the remaining signal.

The ratio metric can be represented mathematically as:

$$R = P_s / P_r$$

Where  $P_s$  is the power of the spoofing signal, and  $P_r$  is the power of the real signal.

4. The article by Wei et al. [23] presents a method for identifying Global Navigation Satellite System (GNSS) spoofing through the use of profile estimation. The method involves spoofing profile estimation-based identification. The proposed methodology involves the reverse reconstruction of the spoofing profile and utilizes the residual distortion features resulting from spoofing attacks within an extended Kalman filter for the purpose of spoofing identification. The aforementioned approach presents a dependable resolution for integrated navigation systems that are closely linked, specifically those involving MEMS INS/GNSS.

Spoofing detection is performed by RSD techniques utilizing the composite signal comprising the authentic satellite signal and the spoofing signal that has been received. The underlying premise of this approach is that the spoofer is required to synchronize with the authentic satellite signal to

obstruct genuine signals. The procedure at hand is a challenging task that necessitates a comprehensive understanding of the three-dimensional location of the spoofing source and the phase center of the receiver antenna.

The receiver tracking loop can more easily track the spoofing signal when the power of the spoofing signal is higher than that of the true signal. As a result, it is technically challenging, places a greater burden on the computer, or necessitates the use of extra monitoring channels to detect the spoofing signal using a weaker actual signal component.

By observing the power change of the received satellite signal, signal quality monitoring equipment may successfully spot selective power spoofing. While the detection performance for denial environments is poor, it is average for selective delay spoofing.

The suggested approach, which combines residual signal monitoring technology with signal quality monitoring technology, offers greater opportunities for accurately identifying denial environment spoofing. This improves the viability and efficacy of RSD approaches in combating various spoofing attacks.

### **2.6.3 Spoofing Detection Method Based on Signal Quality Monitoring**

GNSS spoofing is a growing concern, wherein the GPS signals are manipulated or duplicated, causing the receiver to interpret false data. However, methods have been proposed to detect such manipulations, and one effective approach is the Spoofing Detection based on Signal Quality Monitoring (SQM).

#### **1. Theoretical Framework:**

The detection of spoofing in signal quality monitoring (SQM) techniques is achieved through the identification of distortions in the correlation peak of the multi-correlator output of the receiver during the tracking phase. The identification of spoofing signals is commonly feasible by virtue of the temporal discrepancy between the spurious signal's arrival time and that of the authentic signal at the receiving end. The occurrence of an anomaly in the correlation peak is a result of the disruption caused by a spoofing signal to the receiver correlator output.

Mathematically, if we denote the correlator output as  $c(t)$ , it can be represented as the convolution of the received signal  $r(t)$  and the locally generated replica signal  $s(t)$ , as given by the equation:

$$c(t) = \int r(\tau) * s(t - \tau) d\tau$$

In an ideal scenario without spoofing, the correlation output would peak at the point where  $r(t)$  and  $s(t)$  align perfectly. However, in the presence of a spoofing signal, the correlation peak would be distorted due to the overlapping of the authentic and spoofing signals.

#### **2. Application of Signal Quality Monitoring Technique:**

The utilization of the SQM technique enables the detection of diverse forms of spoofing attacks through the measurement and monitoring of multiple signal quality aspects, including the shape of the correlation peak and residual signals.

The authors Manfredini et al. (2019) introduced an algorithm based on SQM that evaluates the peak quality of the correlation function and utilizes supplementary correlators to detect the residual



signal. The aforementioned methodology exhibits considerable potential in terms of anti-spoofing efficacy while maintaining low complexity. This is achieved by identifying distortions in correlation shapes and residual signals in both static and dynamic scenarios.

Jahromi et al. [25] also used SQM metrics to analyze the effect of tracking-level spoofing on receiver correlator output. They designed an anomaly detection system which considers distortion anomalies or asymmetric correlation peaks caused by the interaction between real signal and spoofing signal peaks.

### 3. Analysis:

SQM-based spoofing detection methods offer an effective and relatively low-complexity way to detect and mitigate GPS spoofing. They can increase detection probability while reducing the false alarm rate.

While this approach demonstrates a satisfactory level of detection accuracy for selective delay spoofing, its performance is suboptimal for nulling attack spoofing. Nevertheless, when utilized in conjunction with other indicators, this technique displays encouraging outcomes. As demonstrated in reference [25], a spoofing detection approach has been developed that integrates four detection indicators and detection thresholds, resulting in enhanced spoofing detection capabilities in comparison to alternative methodologies. However, this method requires further improvement to enhance its robustness against various spoofing scenarios.

SQM-based methods provide an effective approach to detecting GPS spoofing by examining the quality and characteristics of the received signal. These methods show promise in improving the reliability and robustness of GPS systems, although further research and development are required to address their limitations and further enhance their performance.

#### **2.6.3.1 The Principle of C/N0 as an Anti-Spoofing Detection Method**

The C/N0 parameter denotes the quotient of the power of the carrier signal and the noise density in the receiver. Typically, the C/N0 value exhibits stability or a foreseeable trend, as it is contingent upon variables such as the satellite's placement, the receiver's antenna gain pattern, and atmospheric circumstances.

During a spoofing attack, the perpetrator frequently transmits signals with a greater power output than legitimate signals in order to overpower the correlator outputs of the receiver. As a result, the Carrier-to-Noise Density Ratio (C/N0) of the counterfeit signals could exceed anticipated levels, thereby serving as a potential indicator of spoofing. The authors Broumandan [26] introduced an enhanced SQM methodology capable of identifying spoofing in scenarios where both spoofing signals and multipath signals are present. The approach utilizes pre-dispersion metrics and post-dispersion metrics in a collaborative manner to identify instances of spoofing during the tracking phase. The presence of multipath interference can be inferred if and only if the SQM index and the carrier-to-noise ratio indicator surpass the designated threshold. Conversely, if the variance, SQM, and carrier-to-noise ratio all exceed the threshold, the occurrence of spoofing can be detected. Mathematical representation of the Carrier-to-Noise Density Ratio (C/N0) is feasible.

$$C/N0 = (Ps/No)$$

Where:

- $C/N0$  is the carrier – to – noise density ratio, typically measured in dB – Hz
- $P_s$  is the received signal power
- $N_0$  is the noise power density, typically given by  $kTB$  where  $k$  is Boltzmann's constant,  $T$  is the system temperature and  $B$  is the receiver's bandwidth.

### 2.6.3.2 Spoofing Attack Metric Model (SAMM)

[27] The technique suggested by Risbud et al centers on the significant alterations that a spoofing attack can bring about in the phasor measurement unit (PMU) readings, leading to potential disruption of the network's normal functioning. The SAMM model was introduced as a means of converting the problem of network state estimation and attack reconstruction into a non-convex restricted least squares problem. The main objective of this study is to employ a strategy that integrates state estimation and attack reconstruction interaction minimization to enhance the identification of the most vulnerable Phasor Measurement Unit (PMU) in the network.

In mathematical terms, if we denote the system state as  $x$ , the measurement vector as  $z$ , and the attack vector as  $a$ , the least squares estimation problem can be formulated as:

$$\min (z - Hx - a)^2 + \lambda \|a\|^2$$

Here,  $H$  is the measurement matrix and  $\lambda$  is a penalty parameter that balances the trade-off between the fit to the measurements and the size of the attack vector. The problem is to find the system state  $x$  and the attack vector  $a$  that minimizes this objective function.

### 2.6.3.3 Spoofing Signal Elimination

[28] Kim et al. presented a method for removing spoofed channels through signal processing. The method also considers the elimination of spoofing signals by manipulating the radio frequency phase. This technique allows the navigation positioning result to be converted from an abnormal state induced by the spoofing signal to a normal state. The core mathematics here involves phase adjustments that can offset the effects of spoofing.

### 2.6.3.4 Time Hopping Anti-spoofing Signal Processing Algorithm

[29] The algorithm introduced by Berardo et al. entails utilizing several correlators to examine the correlation function for detecting multipath between the input signal and the local copy. The aforementioned methodology enables the recipient to disengage from the spurious signal and re-establish synchronization with the authentic signal through the assessment of the temporal

discrepancy between the legitimate and counterfeit signals. The fundamental mathematical concepts involved in this context are correlation functions and delay estimations.

### **2.6.3.5 Carrier Phase Tracking Spectrum Analysis**

[30] The concept proposed by Zhao et al. rests on the tracking loop's phase detector output signal's spectrum properties, which are strongly correlated with the existence of a spoofing signal. They created a spoofing detection model by applying spectrum analysis to the phase detector's output signal. This plan outlines a technique for detecting spoofing signals based on carrier phase tracking spectrum analysis in conditions of moderate and low dynamics.

### **2.6.3.6 Time-Authentication Algorithm**

[31] Bhamidipati et al. employ a methodology that utilizes a widely distributed static receiver in conjunction with a network of known positions. The process of determining the anticipated time deviation of the P(Y) code received by different receivers involves primarily conducting pairwise cross-correlation procedures on the conditionally restricted four-phase carrier erasure input signal. This technique is capable of detecting spoofing by analyzing the weighted total of the cross-correlation peak offset and amplitude of each receiver and its common satellite.

In mathematical terms, the cross-correlation  $R_{xy}(\tau)$  of two signals  $x(t)$  and  $y(t)$  is given by the integral:

$$R_{xy}(\tau) = \int x(t) y(t + \tau) dt$$

*This integral is evaluated over all time, and  $\tau$  is the time difference applied to  $y(t)$ .*

## **2.6.4 Spoofing Detection Based on Signal Arrival Direction**

In a world dominated by satellite communication and navigation systems, the issue of signal spoofing has emerged as a severe problem, posing serious threats to the security and integrity of data transmissions. A spoofing attack involves transmission of counterfeit signals that mimic authentic satellite signals, leading to erroneous receiver measurements and potentially hazardous consequences. One promising method to mitigate these attacks is based on the direction of signal arrival. The basis of this method lies in the observation that real satellite signals lack spatial correlation observed in spoofed signals. The technique leverages the signal arrival direction to differentiate between legitimate and spoofed signals, thereby providing an efficient means to detect and eliminate spoofing threats. This report presents an in-depth exploration of this technique.

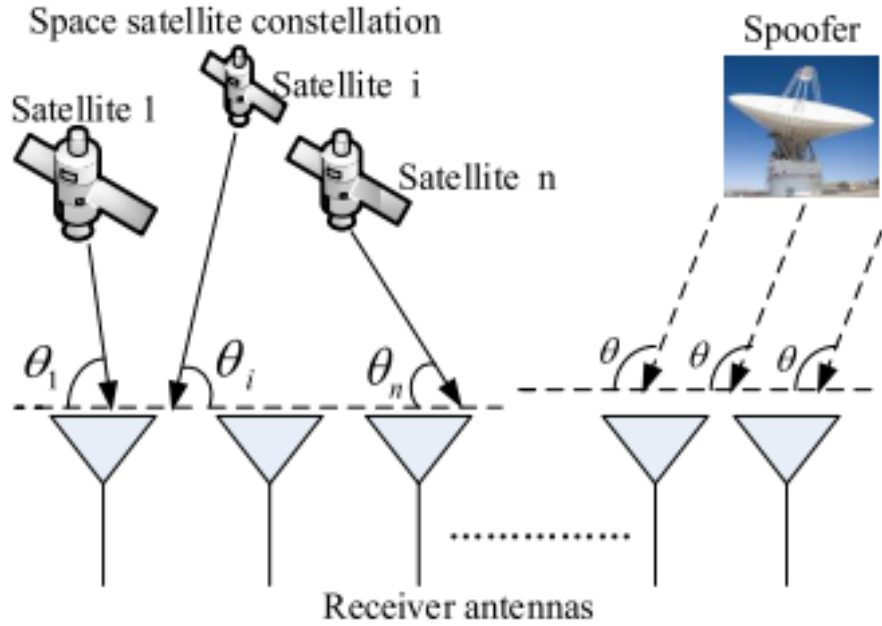


Figure 13 Spoofing detection scenario based on angle of arrival

## 2. Conceptual Framework

The fundamental concept underlying a spoofing detection mechanism that relies on signal arrival direction is spatial correlation. Given the existing technological limitations, it is common for a sole spoofer transmitter to transmit counterfeit signals from multiple satellites, thereby generating signals that exhibit spatial correlation. On the contrary, the absence of spatial correlation is a characteristic feature of genuine signals emanating from singular satellites. Hence, through the utilization of this inherent distinction, a system for detecting spoofing can proficiently differentiate between genuine and falsified signals.

Mathematically, this concept can be expressed as:

If  $S_i$  and  $S_j$  are two signals from satellites  $i$  and  $j$ , then the correlation,  $C_{ij}$ , is given by:

$$C_{ij} = \langle S_i, S_j \rangle / (||S_i|| * ||S_j||)$$

where  $\langle \cdot, \cdot \rangle$  denotes the inner product, and  $||\cdot||$  represents the norm.

In the case of spoofed signals,  $C_{ij}$  tends to be close to 1, due to the spatial correlation. Conversely, for authentic signals,  $C_{ij}$  is nearly 0, indicating no spatial correlation. By setting a threshold for  $C_{ij}$ , a spoofing detection system can effectively distinguish between authentic and spoofed signals.

## 3. Implementation Challenges

The deployment of a spoofing detection mechanism that relies on the direction of signal arrival entails a certain level of intricacy and supplementary expenses. Achieving precise measurement of the direction of signal arrival typically necessitates the cooperative involvement of other navigation systems, such as an attitude measurement instrument and an inertial navigation system.

The amalgamation of these systems results in an increase in the intricacy of implementation and expenses linked with this method of detection.

#### 4. Performance Evaluation

Research has demonstrated that a spoofing detection mechanism that relies on the direction of signal arrival provides a resilient performance in the face of spoofing attacks. A technique that observes the incoming direction of satellite signals and persistently monitors associated variables, such as signal strength, has exhibited encouraging outcomes in identifying instances of targeted power falsification [82]. An alternative approach involves the comparison between the anticipated direction of incoming waves, which is determined by combining ephemeris, and the factual direction of the received signal. This method exhibits promising results in mitigating nulling attacks [83].

Notwithstanding the encouraging outcomes, it is crucial to acknowledge that this approach may not effectively withstand selective delay spoofing or denial environments. Hence, it may be imperative to devise supplementary methodologies or augment the current techniques to attain a holistic anti-spoofing efficacy.

The utilization of signal arrival direction as the basis for spoofing detection method exhibits a potential strategy for safeguarding satellite navigation and communication systems from spoofing attacks. Although the method exhibits significant efficacy in detecting specific forms of spoofing, obstacles pertaining to the intricacy and expense of implementation, along with its incapacity to withstand certain types of spoofing, underscore the necessity for further research and advancement in this domain.

## **Chapter 3 Methodology**

### **3.1 Hardware Components**

#### **3.1.1 Realtek RTL2832U USB Dongle**

The Realtek RTL2832U is a high-performance DVB-T COFDM demodulator, packaged with a USB 2.0 interface. It's a product of Realtek Semiconductor Corp, a renowned player in the field of networking solutions and multimedia ICs. The RTL2832U showcases the firm's dedication to versatility and efficiency.

While its primary role lies in providing an affordable and compact means to receive Digital Video Broadcast Terrestrial (DVB-T) signals - widely used in Europe and other regions - the RTL2832U has found a significant application in the realm of software-defined radio (SDR), showcasing its versatility beyond its original purpose.

When the RTL2832U is paired with an RF tuner IC, such as the Rafael Micro R820T, these dongles are capable of receiving a broad frequency range, typically from around 24 - 1766 MHz. By using appropriate software like SDR#, HDSDR, or GNU Radio, the incoming data can be effectively processed and interpreted. The result is a low-cost, flexible SDR system that can perform a wide range of tasks, making the RTL2832U-based dongles a favorite among radio enthusiasts and hobbyists engaged in signals intelligence.

These devices are USB-powered, offering convenient plug-and-play functionality with personal computers, and have found a place in an array of DIY projects thanks to their low cost and flexibility. However, it's important to note that while the RTL2832U is indeed versatile, utilizing it for SDR purposes necessitates a level of familiarity with signal processing and the relevant software. Nonetheless, for those willing to navigate these technical aspects, the RTL2832U offers an economical entry point into the world of software-defined radio.



Figure 14 RTL 2832U Dongle

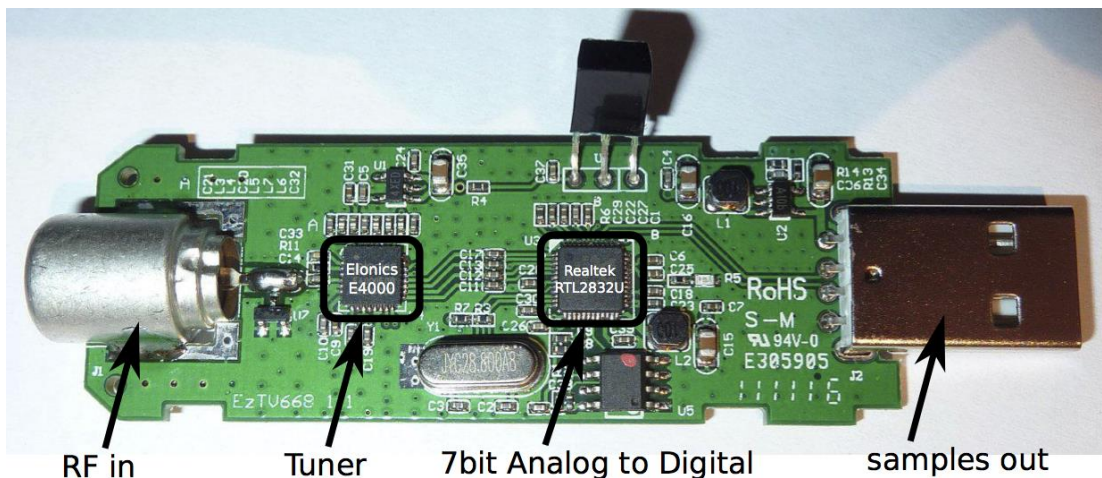


Figure 15 RTL 2832U Dongle Internals

### 3.1.2 T-bias

T-bias, also known as bias-T, is a configuration used in telecommunications and electrical engineering. It allows a device to receive power and data (or signal) simultaneously through a single cable. This is achieved by adding a DC supply to an RF signal, which is then separated into distinct DC and RF signals at the receiving end.

The design of a T-bias includes a capacitor and an inductor, connected in a T-shape configuration. The role of the capacitor is to block any DC component from reaching the RF device, while the inductor allows the DC power to pass through, but not the RF signals. This setup prevents the DC power from interfering with the signal and vice versa.

T-bias is commonly used in applications such as powering remote antennas (like in GPS or TV systems), which need both data signals and power supplied over a long cable. This technology

saves the cost and complexity of running separate power and data cables. It's crucial in these applications to choose components suitable for the frequency range and power level required.

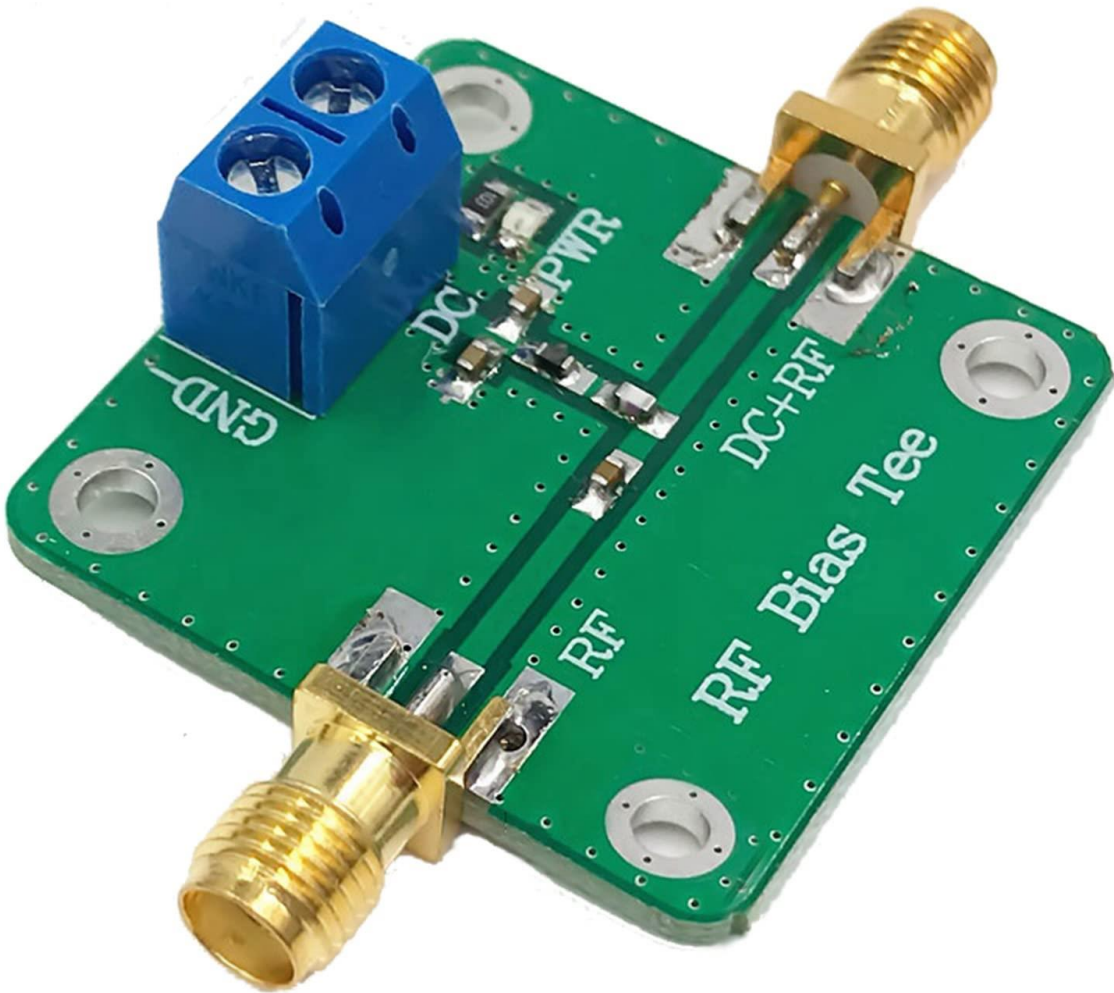


Figure 16 T-Bias

### 3.1.3 GPS L1 Band Active Antenna

The Global Positioning System (GPS) predominantly functions within two frequency bands, namely L1 (1575.42 MHz) and L2 (1227.60 MHz). The L1 frequency band is the predominant band employed by Global Positioning System (GPS) systems. The transmitted data includes the navigation message and the Standard Positioning Service (SPS) signals, and in recent times, the L1C civilian signal has also been incorporated.



A GPS L1 band active antenna is specifically designed to receive signals in this frequency range. Unlike a passive antenna, an active antenna includes a low-noise amplifier (LNA) that amplifies the GPS signals received. This helps to overcome the loss in the cable connecting the antenna to the receiver, thereby improving the overall signal quality and system performance.

The LNA requires power to operate, which is typically supplied through the antenna cable using a bias-T configuration (as mentioned above). The antenna itself is often housed in a weatherproof enclosure for outdoor mounting, as the GPS signals are best received with a clear view of the sky.

Active antennas are critical in many GPS applications where the receiver might be located some distance from the antenna or where the environment might lead to significant signal loss. The amplified signal from an active antenna helps ensure that a high-quality signal reaches the GPS receiver, enhancing the system's overall performance and reliability.



*Figure 17 GPS L1 Band Active Antenna*

### **3.1.4 System Configuration and Preliminary Results**

The experimental setup was designed to capture and process real-time GPS signals. This system arrangement incorporated a DVB dongle, linked to an active patch antenna via t-bias. The purpose of the t-bias was to supply a +5 DC voltage necessary to power the internal Low Noise Amplifier (LNA) within the active antenna. Further details on this setup can be referred from Arribas' PhD Thesis1.

The devices utilized for the experiments comprised Dell Latitude 7440, Dell Latitude 7250, and HP Envy x360 laptops, each furnished with Intel Core CPUs of the 4th, 5th, and 8th generations, along with 8 GB RAM. These machines were operating on Linux Ubuntu 20.04 with the GNU Radio version 3.10.4.

We leveraged GNSS-SDR for acquisition, tracking, and obtaining a position fix with both front-end configurations. The experiments were conducted with the antenna positioned statically atop the CTTC building. The real-time data acquired from this setup was subsequently employed for spoofing detection.

To summarize, this initial study demonstrates the viability of utilizing Realtek-based DVB-T dongles that are cost-effective for GNSS positioning. It is worth noting that, as far as our knowledge extends, this represents the initial occurrence of a GNSS software receiver facilitating instantaneous functionality with RTLSDR-based devices. This significant accomplishment presents opportunities to fully leverage GNSS services through the use of conventional laptops and highly economical hardware. Prospective endeavors encompass carrying out supplementary measurements and augmenting the provision of assistance for RTLSDR devices.

The configuration file for real time data is shown as:

```

##### SIGNAL_CONDITIONER CONFIG #####
SignalConditioner.implementation=Signal_Conditioner

DataAdapter.implementation=Pass_Through

##### INPUT_FILTER CONFIG #####
InputFilter.implementation=Freq_Xlating_Fir_Filter
InputFilter.input_item_type=gr_complex
InputFilter.output_item_type=gr_complex
InputFilter.taps_item_type=float
InputFilter.number_of_taps=5
InputFilter.number_of_bands=2
InputFilter.band1_begin=0.0
InputFilter.band1_end=0.85
InputFilter.band2_begin=0.90
InputFilter.band2_end=1.0
InputFilter.ampl1_begin=1.0
InputFilter.ampl1_end=1.0
InputFilter.ampl2_begin=0.0
InputFilter.ampl2_end=0.0
InputFilter.band1_error=1.0
InputFilter.band2_error=1.0
InputFilter.filter_type=bandpass
InputFilter.grid_density=16
InputFilter.sampling_frequency=2000000
InputFilter.IF=14821

##### RESAMPLER CONFIG #####
Resampler.implementation=Pass_Through
Resampler.dump=false
Resampler.item_type=gr_complex

```

Figure 18 Configuration File For Real Time Data

## 3.2 Implementation of various anti spoofing techniques

### 3.2.1 Power level check implementation

Power level check is implemented in tracking block of gnss sdr. Code for power level is implemented in line 1459 of file *dll\_pll\_veml\_tracking.cc* present at following path:

*gnss\_sdr/src/algorithms/tracking/gnuradio\_blocks/*

The anti-spoofing check implemented here is based on the variance of the prompt correlator in-phase component. This variance represents the power level of the signal. In typical conditions, the power level of a GNSS signal should remain relatively stable. Sudden increases in power might suggest a spoofing attack, where an attacker is transmitting a signal that is much stronger than the authentic signals from the GNSS satellites.

This part of the code monitors the 'Inphase' component of the prompt signal (prompt\_I). It keeps track of the prompt\_I signal values over a certain window (d\_amp\_vector\_size). If the size of the collected data (d\_prompt\_I\_vector) exceeds the window size, it removes the excess elements from the start of the vector, ensuring a rolling window of the latest signal data.

Once the window is full, it calculates the expected value (mean) and variance of these signal values and adds the variance to the d\_prompt\_I\_var\_vector. If this variance vector exceeds 1/10th of the window size, it removes the excess elements from the start and calculates the average variance. This value is then logged for monitoring.

The code also checks for bit synchronization. If this is enabled, it checks if the absolute value of the current prompt\_I signal is greater than a threshold value. If it is and spoofing has not been detected before, it flags spoofing as true, logs a message, and notes the current count in d\_spoofing\_mark. If the value is below the threshold and spoofing was previously detected, it marks spoofing as false and resets d\_spoofing\_mark. It adjusts the threshold value if the absolute prompt\_I signal is above the current threshold.

If bit synchronization is not enabled, it adds the absolute value of the current prompt\_I signal to a running sum (d\_prompt\_I\_sum) and adjusts the threshold if needed.

Finally, the current value of d\_spoofing\_mark is written to a dump file, and the prompt\_I count is incremented. It logs the current time, normalized prompt\_I signal (absolute value divided by the threshold), and whether spoofing is detected.

### 3.2.2 Carrier-to-noise density ratio (C/No) check:

This check is defined in line 134 of header file *spoofing\_detector.h* present at path

*Gnss\_sdr/src/algorithms/libs/*

This check is implemented in line 457 of *file spoofing\_detector.cc* present at path

*Gnss\_sdr/src/algorithms/libs/*

And this function is called in line 2070 of file *rtklib\_pvt\_gs.cc* present at path *src/algorithms/PVT/gnuradio\_blocks/*

In normal circumstances (i.e., when there is no spoofing), the C/N0 values from actual satellites tend to remain relatively stable over short periods of time, and they follow a specific statistical distribution. The standard deviation of these C/N0 values, in this case, would be relatively low because there are no drastic changes in signal strength that deviate from the mean C/N0 value.

However, when a spoofing attack occurs, the spoofer typically broadcasts counterfeit signals at a power level higher than that of the actual signals to ensure that the receiver locks onto the fake signals. This results in a sudden increase in the perceived C/N0 values. Because the spoofer's signal strength can vary and does not follow the same constraints as the actual satellite signals, the C/N0 values of the spoofed signals can have a larger spread, resulting in a higher standard deviation.

It's important to note that while a significant change in C/N0 values can be an indication of a potential spoofing attack, it's not definitive proof. Other factors, such as changes in the propagation environment (e.g., moving from an open sky to an urban canyon), can also cause significant changes in C/N0. Therefore, the C/N0 check should be just one of several checks performed by a comprehensive anti-spoofing solution.

The code working is as following:

1. Calculate the sum of all the CNO values by iterating through `cno_vector`.
2. Calculate the mean by dividing the sum by `cno_vector.size`
3. Calculate the standard deviation of the CNO values. First, calculate the squared differences from the mean. Then, add up these squared differences.
4. Compute the square root of the sum of squared differences divided by , `cno_vector.size()`,-
5. If the standard deviation is less than `d_cno_threshold`, it logs an info message, and assigns the standard deviation to `d_score.cno`

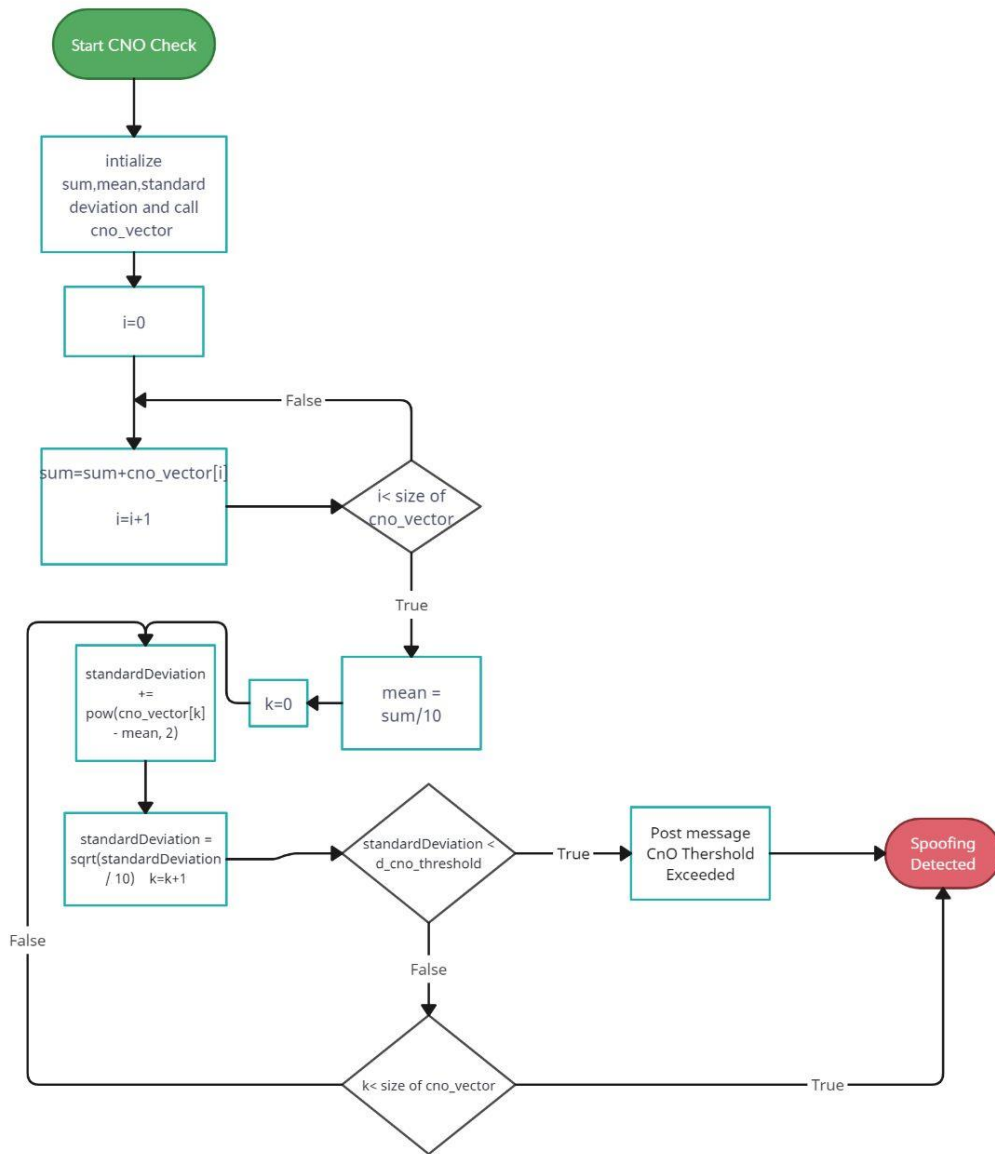


Figure 19 Carrier to Noise Density Ratio Check Flow Chart

### 3.2.3 Position Consistency Check

#### 3.2.3.1 Static position check

This check is defined in line 226 of header file *spoofing\_detector.h* present at path *Gnss\_sdr/src/algorithms/libs/*

This check is implemented in line 157 of file *spoofing\_detector.cc* present at path *Gnss\_sdr/src/algorithms/libs/*

And this function is called in line 104, 228, 236, 238, 295 of file *spoofing\_detector.cc* present at path *Gnss\_sdr/src/algorithms/libs/*

This anti-spoofing technique is particularly suited for stationary GNSS receivers, where the physical position of the receiver is known and considered fixed. The premise of this method is simple: continuously monitor the reported position of the receiver. Given the device's stationary status, its reported position should remain reasonably constant, barring minor variations due to factors such as signal noise or atmospheric conditions.

However, during a spoofing attack, the reported position will start to deviate significantly from the known static location. This deviation can be detected and, if it surpasses a specific threshold, can be flagged as a potential spoofing incident.

Here is how code works:

The function 'validate\_location\_proximity' is used to validate whether the positions provided are within an acceptable range. It accepts three parameters:

1. `const PvtSol* pvtSol1`: This is a pointer to the first PvtSol object, which presumably stores details about a GPS solution, such as latitude and longitude.
2. `const PvtSol* pvtSol2`: This is a pointer to the second PvtSol object to be compared with the first one.
3. `int test_id`: This is an identifier for the type of test to be conducted on the positions.

Here's how the function works:

First, it calculates the distance between `pvtSol1` and `pvtSol2` using the `calculate_distance` function.

Then, depending on the `test_id` parameter, the function performs one of three checks:

- If `test_id` is 1, the function checks if the distance between the positions is less than a configured `radius` (`d_geo_fence_radius`). If it is, the function updates `d_score.static_pos_check_score` to 0 and returns true, indicating the positions are close enough. Otherwise, it sets `d_score.static_pos_check_score` to the calculated distance and returns false.
- If `test_id` is 2, it compares the distance between the positions to a configured maximum jump distance (`d_max_jump_distance`). If the distance is less, it updates `d_score.position_jump_score` to 0 and returns true. Otherwise, it sets `d_score.position_jump_score` to the calculated distance and returns false.
- If `test_id` is 3, it checks if the distance between the positions is less than a configured error threshold (`d_pos_error_threshold`). It then returns a Boolean value indicating whether this is the case.

Here is flow chart of code:

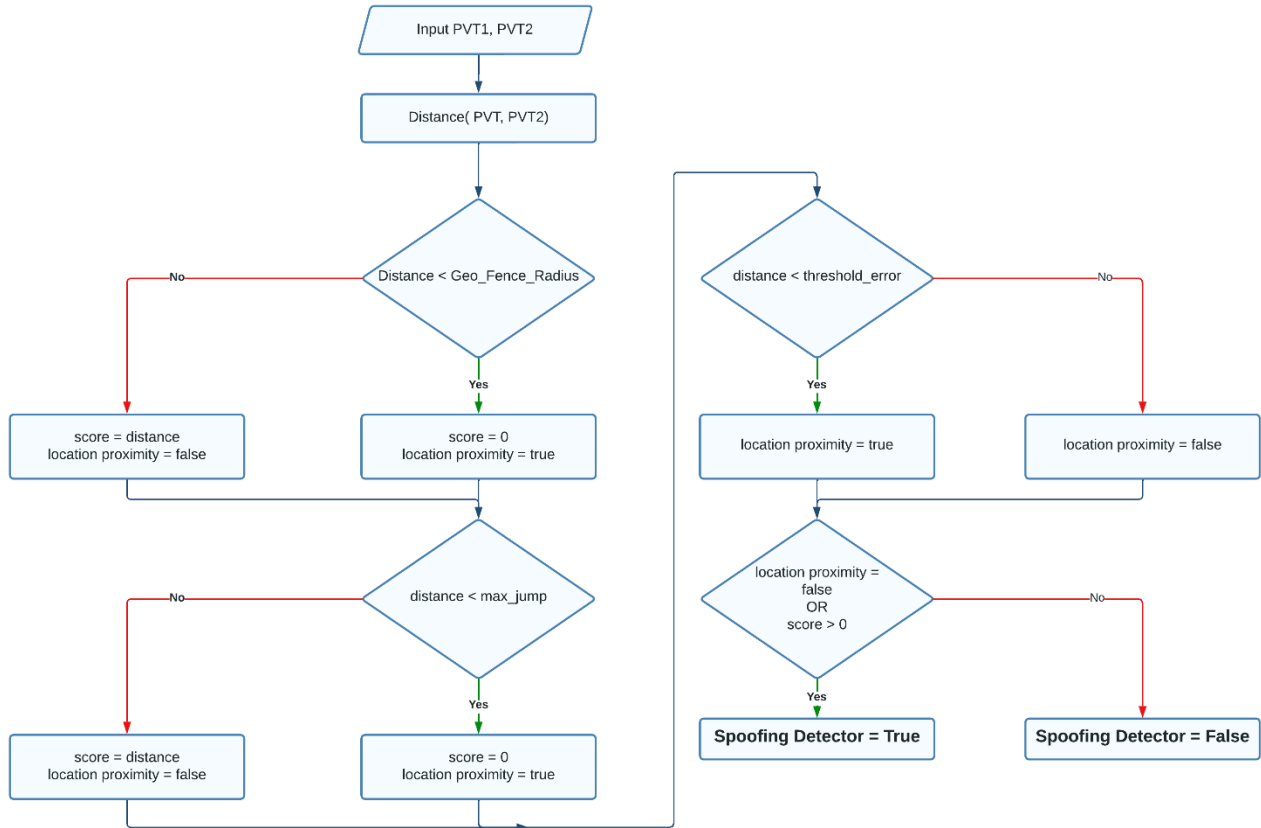


Figure 20 Static Position Check flow chart

### 3.2.3.2 Position Jump Check

The position jump check is a type of GNSS anti-spoofing technique that detects sudden changes in the reported position of a GPS receiver, which could be indicative of a spoofing attack.

In normal operation, we expect the reported position of the GPS receiver to change gradually as the receiver moves or stay approximately constant if the receiver is stationary. However, in a spoofing attack, the attacker might attempt to manipulate the GNSS signals to make the receiver think it's at a different location, causing a "jump" in the reported position.

By monitoring for these jumps and comparing them with a threshold (which could be based on the maximum speed of the receiver, for example), we can detect possible spoofing incidents.

However, this technique is not foolproof, as there can be legitimate reasons for a position jump (e.g., if the receiver loses lock on the satellites and then reacquires them). Therefore, the algorithm also includes additional checks to reduce the likelihood of false positives.

1. The working of code is as follows:



It starts by checking if this is the first record (`d_first_record`). If it is, the function updates the old position, checks the current spoofing score (with `get_spoof_score()`), and if it is 0 (indicating no detected spoofing), it sets the last known good position (`lkg_pvt`) to the current coordinates. The function then sets '`d_first_record`' to false and returns false.

2. If `d_first_record` is false, the function checks if `d_score.position_jump_score` is greater than 0. If it is, it validates the proximity between the last known good position and the new position. If they are close enough, it resets the position jump check.

3. If `d_score.position_jump_score` is 0, the function validates the proximity between the old position and the new position. If they are not close enough, it then validates the proximity between the last known good position and the new position.

4. If the new position is not close to either the old position or the last known good position, the function checks the propagated position. If the check fails, it sets `is_spoofing` to true. If `is_spoofing` is true, it logs the spoof score, updates the last known good position to the old position (if `d_update_lkgp` is true), and returns true.

5. If the new position is close to the old position, the function updates the last known good position to the new position.

6. Finally, if none of the previous conditions are met, the function returns false.

Here is flow chart of code.

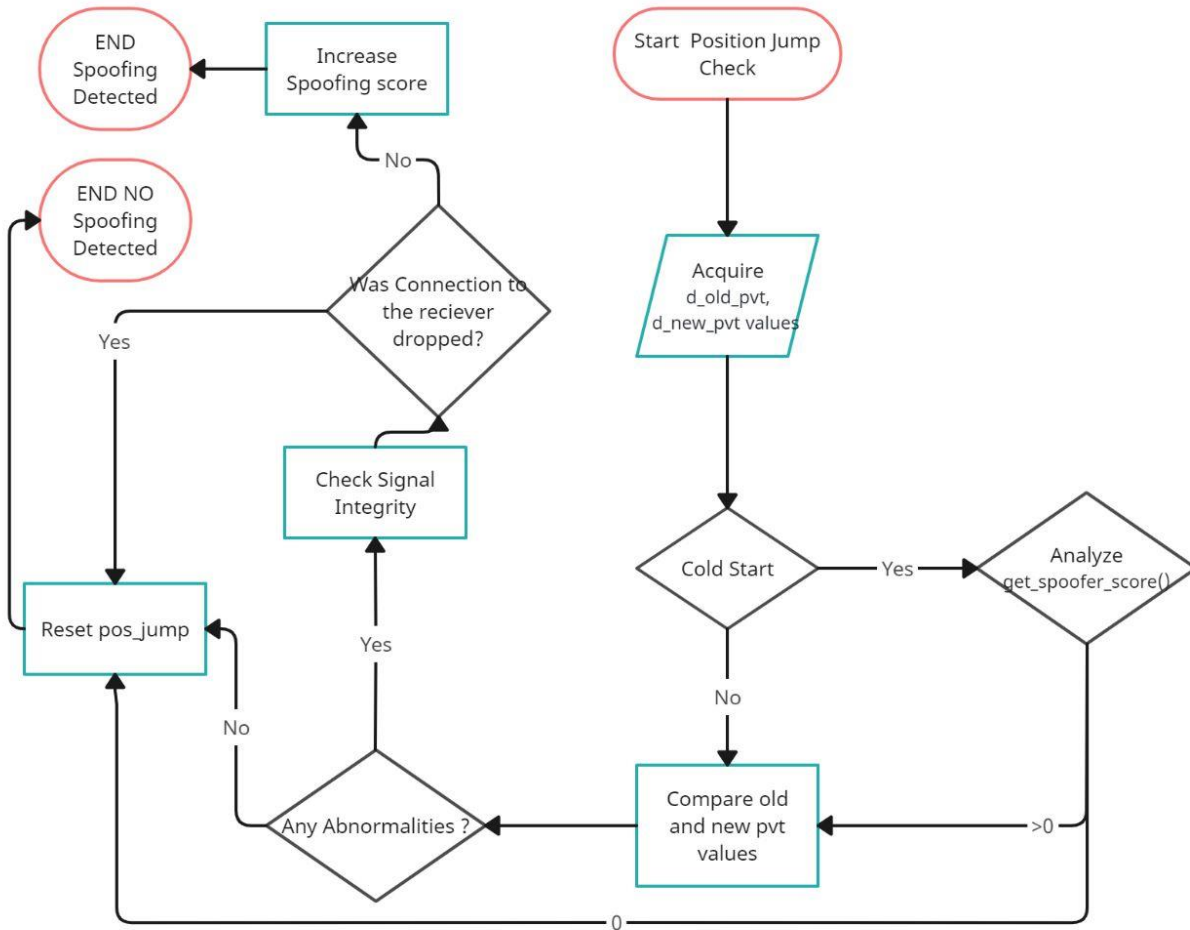


Figure 21 Position jump Test Flow Chart

### 3.2.4 Velocity Consistency Check

*check\_propagated\_pos()*

This check is defined in line 227 of header file *spoofing\_detector.h* present at path

*Gnss\_sdr/src/algorithms/libs/*

This check is implemented in line 275 of cc file *spoofing\_detector.cc* present at path

*Gnss\_sdr/src/algorithms/libs/*

And this function is called in line 149, 249 of file *spoofing\_detector.cc* present at path

*Gnss\_sdr/src/algorithms/libs/*

*check\_velocity\_consistency()*

This check is defined in line 225 of header file *spoofing\_detector.h* present at path

*Gnss\_sdr/src/algorithms/libs/*

This check is implemented in line 144 of cc file *spoofing\_detector.cc* present at path.

*Gnss\_sdr/src/algorithms/libs/*

And this function is called in line 117 of file *spoofing\_detector.cc* present at path  
*Gnss\_sdr/src/algorithms/libs/*

The velocity consistency check is a GNSS anti-spoofing technique that checks the consistency between the reported velocity and the change in position. In normal operation, the change in position over a certain time interval should be consistent with the reported velocity. If a spoofing attack is attempting to manipulate the receiver's position without correctly emulating the corresponding velocity, this technique could potentially detect the discrepancy.

However, this technique also relies on the accuracy of the velocity and position measurements, which can be affected by various factors such as signal noise and atmospheric conditions. Therefore, it might need to be combined with other techniques to achieve reliable anti-spoofing protection.

Here is working of code:

The function `check_velocity_consistency()` is another method in the `SpoofingDetector` class. This method checks the consistency between the reported velocity and the change in position. It increments the `d_checked_velocity_pairs` count and if `check_propagated_pos()` returns false, it increments `d_velocity_error`.

Here's an explanation of how these functions work:

`check_propagated_pos()` function

1. `PvtSol temp_pvt;` This line creates a new `PvtSol` object to hold the expected position.
2. `double dt = (d_new_pvt.timestamp - d_old_pvt.timestamp) / 1000;` This line calculates the time elapsed between the old and new positions in seconds.
3. The next four lines calculate the conversion factors from degrees or radians to meters in the latitude and longitude directions.
4. `temp_pvt.lat`, `temp_pvt.lon`, and `temp_pvt.alt` are calculated based on the old position and velocity and the elapsed time.
5. `double distance_error = calculate_distance(&temp_pvt, &d_new_pvt);` This line calculates the distance error between the expected position (`temp_pvt`) and the new received position (`d_new_pvt`).
6. Then it logs the expected position, received position, and the error.
7. `return validate_location_proximity(&temp_pvt, &d_new_pvt, 3);` This line validates the proximity between the expected position and the new received position.

`check_velocity_consistency()` function

1. `++d_checked_velocity_pairs;` This line increments the count of checked velocity-position pairs.
2. `if(!check_propagated_pos()) { ++d_velocity_error; }` This line checks if the new received position matches the expected position. If it doesn't, it increments the velocity error count.
3. `d_score.velocity_check_score = d_velocity_error / d_checked_velocity_pairs;` This line calculates the velocity check score, which is the ratio of the velocity error count to the count of checked velocity-position pairs.
4. Then it logs the velocity error count and the count of checked velocity-position pairs.

### 3.2.5 Abnormal Position Check

This check is defined in line 228 of header file *spoofing\_detector.h* present at path

*Gnss\_sdr/src/algorithms/libs/*

This check is implemented in line 131 of cc file *spoofing\_detector.cc* present at path

*Gnss\_sdr/src/algorithms/libs/*

And this function is called in line 119 of file *spoofing\_detector.cc* present at path

*Gnss\_sdr/src/algorithms/libs/*

The abnormal position check is a GNSS anti-spoofing technique that checks for abnormalities in the new received position of the GPS receiver. In normal operation, the speed of the GPS receiver should be within certain limits. If a spoofing attack is attempting to manipulate the receiver's position to give an abnormally high or low speed, this technique could potentially detect the anomaly.

However, this technique also relies on the accuracy of the speed measurement, which can be affected by various factors such as signal noise and atmospheric conditions. Therefore, it might need to be combined with other techniques to achieve reliable anti-spoofing protection.

The values of speed and altitude can be adjusted in configuration file as following:

*SecureGNSS.min\_altitude = -10*

*SecureGNSS.max\_altitude = 20000*

*SecureGNSS.min\_ground\_speed = 0*

*SecureGNSS.max\_ground\_speed = 200*

The working of code is as follows:

1. `d_score.abnormal_position_score= 0`; This line resets the abnormal position score before each check.
2. `if (d_new_pvt.speed_over_ground < d_min_ground_speed)`  
`d_score.abnormal_position_score += 0.25`; This line checks if the speed of the GPS receiver over the ground is less than a minimum threshold `d_min_ground_speed`. If it is, it increases the abnormal position score by 0.25.
3. `if (d_new_pvt.speed_over_ground > d_max_ground_speed)`  
`d_score.abnormal_position_score += 0.25`; This line checks if the speed of the GPS receiver over the ground is greater than a maximum threshold `d_max_ground_speed`. If it is, it increases the abnormal position score by 0.25.
4. `DLOG(INFO) << "ABNORMAL_CHECK: " << d_score.abnormal_position_score`; This line logs the abnormal position score after the checks.

Here is slow chart of code:

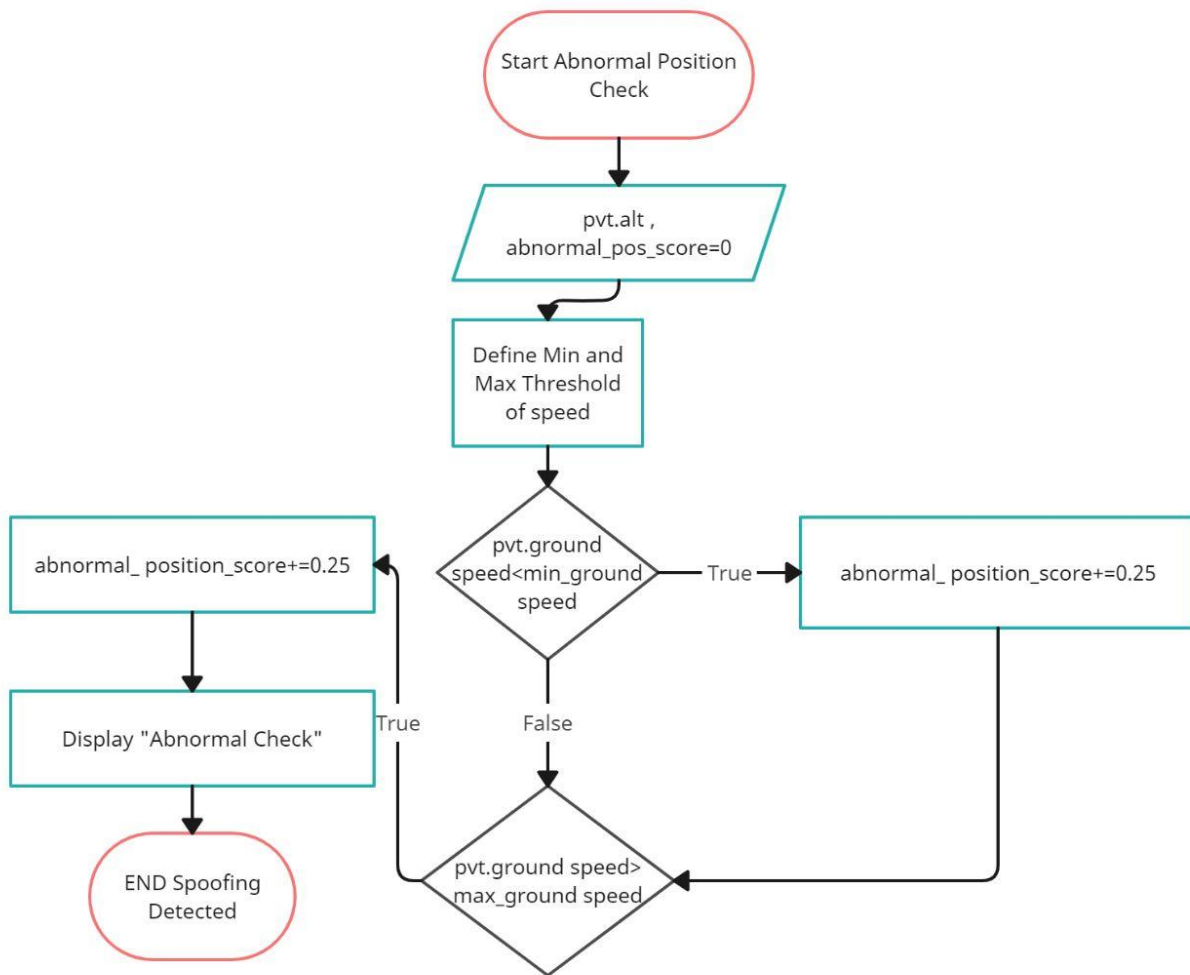


Figure 22 Abnormal Position Check FlowChart

### 3.2.6 Clock Offset Check

This check is defined in line 137 of header file *spoofing\_detector.h* present at path *Gnss\_sdr/src/algorithms/libs/*

This check is implemented in line 313 of cc file *spoofing\_detector.cc* present at path *Gnss\_sdr/src/algorithms/libs/*

And this function is called in line 2089 of file *rtklib\_pvt\_gs.cc* present at path *Gnss\_sdr/src/algorithms/PVT/gnuradio\_blocks/*

Clock offset check is an anti-spoofing technique that verifies the integrity of the GNSS receiver's clock. In a spoofing attack, the adversary might manipulate the signals to induce a clock offset or drift in the receiver's clock. By checking the clock offset and drift measurements against expected behaviors, this method could potentially detect the abnormal clock behaviors caused by spoofing.

The working of code is as follows:

The `check_clock_offset()` function in the `SpoofingDetector` class takes two arguments, `clk_offset` and `clk_drift`. It evaluates the potential existence of spoofing based on the measured GNSS receiver clock offset and drift. Here's a detailed walkthrough:

1. `offset.offset = clk_offset * 1e9`; This line scales the clock offset from seconds to nanoseconds.
2. `offset.drift = clk_drift * 1e-3`; This line scales the clock drift from parts per second to parts per thousand.
3. `offset.timestamp = SpoofingDetector::CurrentTime_nanoseconds()`; The current time in nanoseconds is recorded as the timestamp for the clock offset measurement.
4. The function then checks if the number of recorded clock offset measurements is less than a certain threshold `d_clk_offset_vector_size`. If so, it adds the new clock offset measurement to the vector and returns `false`, indicating no spoofing is detected.
5. If the number of recorded clock offset measurements is more than `d_clk_offset_vector_size`, it removes the oldest measurements until it reaches `d_clk_offset_vector_size`.
6. The function then calculates the mean and variance of the clock drift measurements.
7. It computes the predicted clock offset `offset_propd` based on the mean clock drift and the time difference between the latest two clock offset measurements.
8. The error between the expected and actual clock offset is calculated as `offsetError`.
9. This error is then added to another vector `d_clock_offset_errors_vector`, which also maintains a certain size `d_clk_offset_vector_size / 10`.

10. If the size of `d_clock_offset_errors_vector` exceeds `d_clk_offset_vector_size / 10`, the oldest errors are removed and the average of the remaining errors is calculated.

11. If the average error exceeds a threshold `d_clk_offset_error`, it's determined that spoofing is likely occurring.

12. The function then logs the calculated variables and returns the spoofing flag.

Here is flow chart of code:

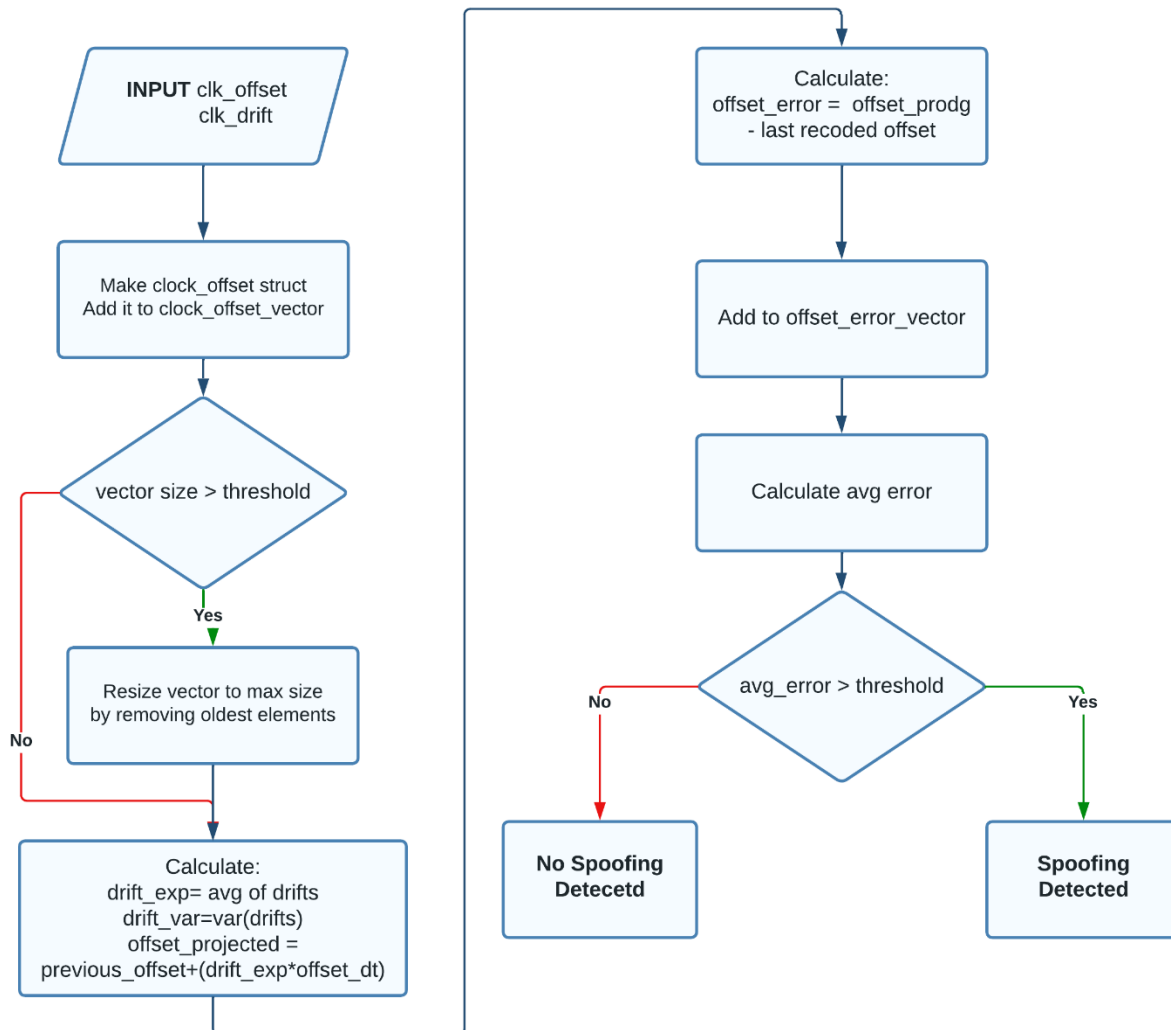


Figure 23 Clock Offset Check

### 3.2.7 Clock Jump Test

This check is defined in line 147 of header file `spoofing_detector.h` present at path `Gnss_sdr/src/algorithms/libs/`

This check is implemented in line 384 of cc file *spoofing\_detector.cc* present at path *Gnss\_sdr/src/algorithms/libs/*

And this function is called in line 531 of file *gps\_ll\_ca\_telemetry\_decoder\_gs.cc* present at path *Gnss\_sdr/src/algorithms/telemetry\_decoder/gnuradio\_blocks/*

The Clock Jump Check is a form of GNSS anti-spoofing that seeks to detect unexpected discontinuities (jumps) in the GNSS receiver's clock. These jumps can be caused by a spoofer sending signals that alter the perceived time, causing the receiver's clock to abruptly change. Continuously monitoring the receiver's clock for such jumps can thus provide a line of defense against this type of attack.

The working of code is as follows:

- If this is the first record (*d\_first\_record* is true), the function initializes the old and last known good (LKG) clock states, then turns off the *d\_first\_record* flag and returns false. This initial state setting helps prepare for the upcoming clock jump checks. It logs an informational message stating that the clock jump check has been enabled and then it sets *d\_first\_record* to false to avoid resetting the clock states in subsequent checks.
- If it is not the first record, the function proceeds to check for clock jumps by invoking *check\_clock\_jump()*. This function analyzes the receiver's clock to look for abrupt changes in clock value, which may signify a spoofing attempt.



Here is flow chart of code:

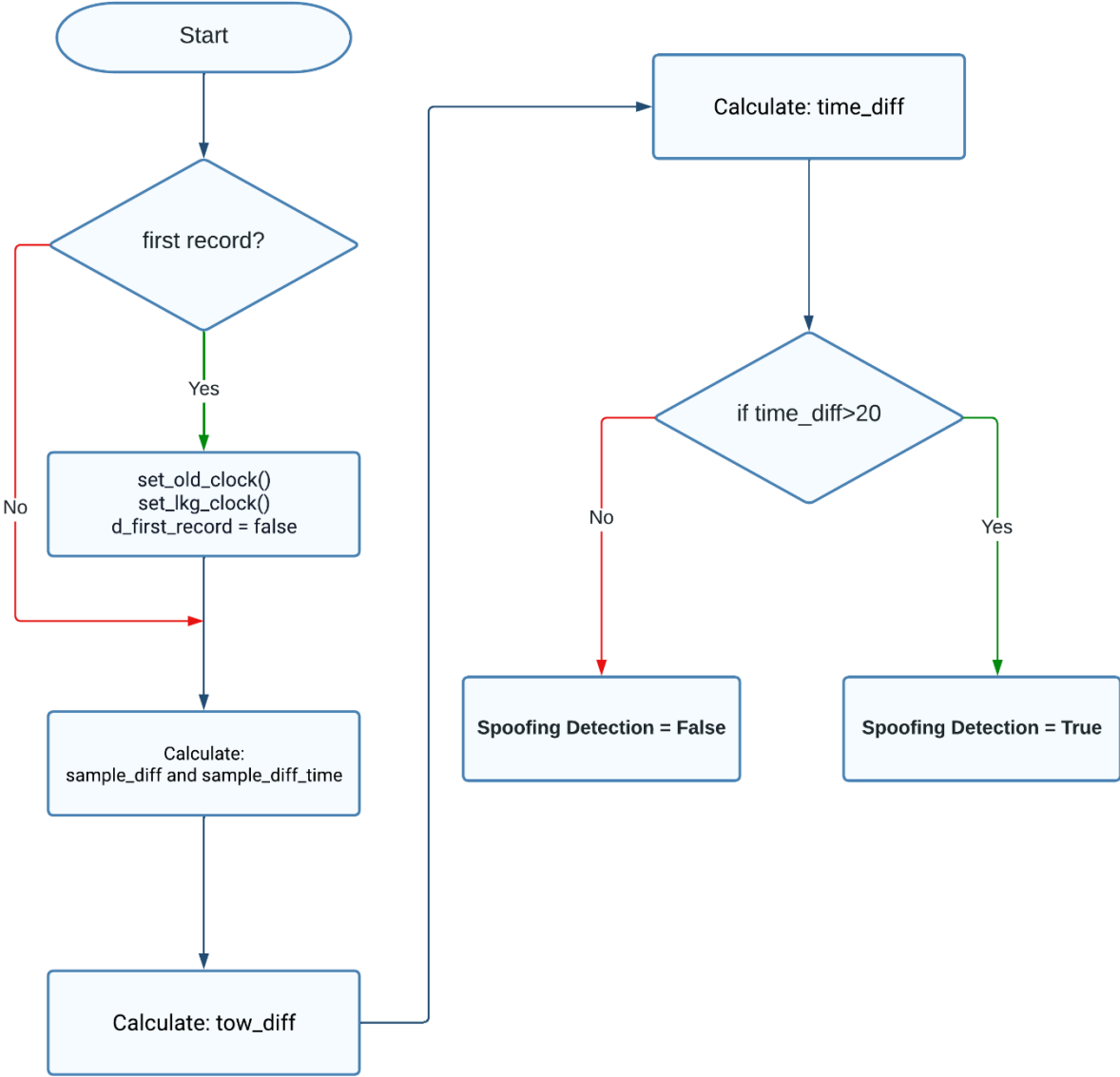


Figure 24 Clock Jump Check Test

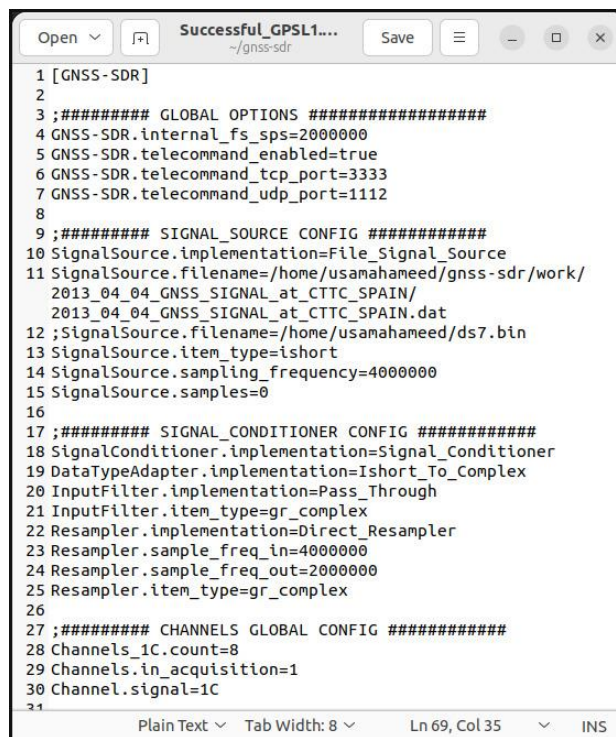
## Chapter 4: Results

This chapter presents the experimental results of the project, where the configuration file in the GNSS-SDR directory was implemented, and the receiver was controlled remotely using a Telecommand interface. Detailed analysis of the extracted Position, Velocity, Time (PVT) data and the subsequent spoofing report highlight the effectiveness of our anti-spoofing measures.

### 4.1 Acquisition of PVT Information

The extraction of Position, Velocity, Time (PVT) information was a central part of our experiment. PVT data served as the primary output from the GNSS-SDR receiver and played a crucial role in the evaluation of our anti-spoofing measures. This subsection delves into the process of acquiring PVT information.

The process was initiated by executing the configuration file found in the GNSS-SDR directory via terminal. The configuration file dictates the behavior of the GNSS-SDR receiver, setting various parameters for the software-defined radio to follow. The specific configurations used for our experiment are shown in the figure:



```
1 [GNSS-SDR]
2
3 ;##### GLOBAL OPTIONS #####
4 GNSS-SDR.internal_fs_sps=2000000
5 GNSS-SDR.telecommand_enabled=true
6 GNSS-SDR.telecommand_tcp_port=3333
7 GNSS-SDR.telecommand_udp_port=1112
8
9 ;##### SIGNAL_SOURCE CONFIG #####
10 SignalSource.implementation=File_Signal_Source
11 SignalSource.filename=/home/usamahameed/gnss-sdr/work/
  2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN/
  2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.dat
12 ;SignalSource.filename=/home/usamahameed/ds7.bin
13 SignalSource.item_type=ishort
14 SignalSource.sampling_frequency=4000000
15 SignalSource.samples=0
16
17 ;##### SIGNAL_CONDITIONER CONFIG #####
18 SignalConditioner.implementation=Signal_Conditioner
19 DataTypeAdapter.implementation=Ishort_To_Complex
20 InputFilter.implementation=Pass_Through
21 InputFilter.item_type=gr_complex
22 Resampler.implementation=Direct_Resampler
23 Resampler.sample_freq_in=4000000
24 Resampler.sample_freq_out=2000000
25 Resampler.item_type=gr_complex
26
27 ;##### CHANNELS GLOBAL CONFIG #####
28 Channels_1C.count=8
29 Channels.in_acquisition=1
30 Channel.signal=1C
31
```

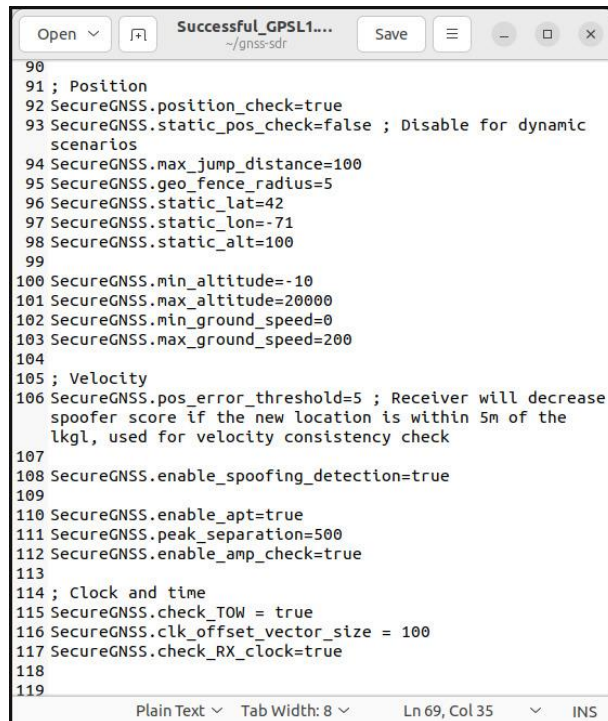
Figure 25 Configuration File part 1

```
Open Save
Successful_GPS1...
~/gnss-sdr
30 Channel.signal=1C
31
32 ;##### ACQUISITION GLOBAL CONFIG #####
33 Acquisition_1C.implementation=GPS_L1_CA_PCPS_Acquisition
34 Acquisition_1C.item_type=gr_complex
35 Acquisition_1C.coherent_integration_time_ms=1
36 Acquisition_1C.pfa=0.01
37 ;Acquisition_1C.pfa=0.01
38 Acquisition_1C.doppler_max=10000
39 Acquisition_1C.doppler_step=250
40 Acquisition_1C.dump=false
41 Acquisition_1C.dump_filename=./acq_dump.dat
42
43 SecureACQ.enable_aprt=true
44 SecureACQ.channels_per_sv=2
45
46 ;##### TRACKING GLOBAL CONFIG #####
47 Tracking_1C.implementation=GPS_L1_CA_DLL_PLL_Tracking
48 Tracking_1C.item_type=gr_complex
49 Tracking_1C.dump=true
50 Tracking_1C.dump_filename=epl_tracking_ch_
51 Tracking_1C pll_bw_hz=40.0;
52 Tracking_1C.dll_bw_hz=4.0;
53 Tracking_1C.order=3;
54 Tracking_1C.dump=false
55 Tracking_1C.dump_filename=./data/epl_tracking_c
56
57 ;##### TELEMETRY DECODER GPS CONFIG #####
58 TelemetryDecoder_1C.implementation=GPS_L1_CA_Telemetry_Dec
59 TelemetryDecoder_1C.dump=false
60 TelemetryDecoder_1C.security_checks=true
61
62
Plain Text Tab Width: 8 Ln 69, Col 35 INS
```

Figure 26 Configuration File part 2

```
Open Save
Successful_GPS1...
~/gnss-sdr
63 ;##### OBSERVABLES CONFIG #####
64 Observables.implementation=Hybrid_Observables
65 Observables.dump=false
66 Observables.dump_filename=./observables.dat
67
68
69 ;##### PVT CONFIG #####
70 PVT.implementation=RTKLIB_PVT
71 PVT.positioning_mode=PPP_Static ; options: Single,
Static, Kinematic, PPP_Static, PPP_Kinematic
72 PVT.iono_model=Broadcast ; options: OFF, Broadcast, SBAS,
Iono-Free-LC, Estimate_STEC, IONEX
73 PVT.trop_model=Saastamoinen ; options: OFF, Saastamoinen,
SBAS, Estimate_ZTD, Estimate_ZTD_Grad
74 PVT.output_rate_ms=1
75 PVT.display_rate_ms=1000
76 PVT.dump_filename=./PVT
77 PVT.nmea_dump_filename=./gnss_sdr_pvt.nmea;
78 PVT.flag_nmea_tty_port=false;
79 PVT.nmea_dump_devname=/dev/pts/4
80 PVT.flag_rtcn_server=false
81 PVT.flag_rtcn_tty_port=false
82 PVT.rtcn_dump_devname=/dev/pts/1
83 PVT.dump=false
84
85 PVT.security_checks=true
86 PVT.print_score=true
87
88 ;##### Secure PVT #####
89 SecureGNSS.dump_pvt_checks_results=true;
90
91 ; Position
Plain Text Tab Width: 8 Ln 69, Col 35 INS
```

Figure 27 Configuration File part 3

A screenshot of a text editor window titled "Successful\_GPS1..." with a file path of "~/gnss-sdr". The window contains a configuration file with the following content:

```
90
91 ; Position
92 SecureGNSS.position_check=true
93 SecureGNSS.static_pos_check=false ; Disable for dynamic
  scenarios
94 SecureGNSS.max_jump_distance=100
95 SecureGNSS.geo_fence_radius=5
96 SecureGNSS.static_lat=42
97 SecureGNSS.static_lon=-71
98 SecureGNSS.static_alt=100
99
100 SecureGNSS.min_altitude=-10
101 SecureGNSS.max_altitude=20000
102 SecureGNSS.min_ground_speed=0
103 SecureGNSS.max_ground_speed=200
104
105 ; Velocity
106 SecureGNSS.pos_error_threshold=5 ; Receiver will decrease
  spoofer score if the new location is within 5m of the
  lkgl, used for velocity consistency check
107
108 SecureGNSS.enable_spoofing_detection=true
109
110 SecureGNSS.enable_apt=true
111 SecureGNSS.peak_separation=500
112 SecureGNSS.enable_amp_check=true
113
114 ; Clock and time
115 SecureGNSS.check_TOW = true
116 SecureGNSS.clk_offset_vector_size = 100
117 SecureGNSS.check_RX_clock=true
118
119
```

The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 69, Col 35", and "INS".

Figure 28 Configuration File part 4

Once the configuration file was executed, the GNSS-SDR started decoding the satellite signals. The decoder then processed these signals to extract PVT information, i.e., the receiver's position (longitude and latitude), velocity, and time at each second. This high-resolution data capture was made possible by the sophisticated design of the GNSS-SDR, allowing us to monitor the receiver's performance in real-time.

The output data from this process contains rich information about the receiver's position, velocity, and the precise time of the data capture. The specific results, including the longitudinal and latitudinal coordinates, velocity measurements, and timestamped data, are presented in the figure:



```

GNSS-SDR program ended.
usamahameed@usamahameed-Latitude-E7250:~/gnss-sdr$ ./src/Utils/scripts
/gnss-sdr-harness.sh gnss-sdr -c Successful_GPSL1.conf
gnss-sdr -c Successful_GPSL1.conf
Initializing GNSS-SDR v0.0.14.git-gsoc_21_sd-f031e52a2 ... Please wait
.
Logging will be written at "/tmp"
Use gnss-sdr --log_dir=/path/to/log to change that.
RF Channels: 1
Requested number of threads exceeds processor count. Proceed with caution
Requested number of threads exceeds processor count. Proceed with caution
Requested number of threads exceeds processor count. Proceed with caution
Processing file /home/usamahameed/gnss-sdr/work/2013_04_04_GNSS_SIGNAL
_at_CTTT_SPAIN/2013_04_04_GNSS_SIGNAL_at_CTTT_SPAIN.dat, which contains
800000000 samples (1600000000 bytes)
GNSS signal recorded time to be processed: 99.999 [s]
TcpCmdInterface: Telecommand TCP interface listening on port 3333
Tracking of GPS L1 C/A signal started on channel 0 for satellite GPS PRN
01 (Block IIF)
Current receiver time: 1 s
Tracking of GPS L1 C/A signal started on channel 3 for satellite GPS PRN
11 (Block IIR)
Tracking of GPS L1 C/A signal started on channel 6 for satellite GPS PRN
20 (Block IIR)
Tracking of GPS L1 C/A signal started on channel 1 for satellite GPS PRN
32 (Block IIF)
Current receiver time: 2 s
Current receiver time: 3 s
Current receiver time: 4 s
Current receiver time: 5 s
Current receiver time: 6 s
Current receiver time: 7 s
Current receiver time: 8 s
Tracking of GPS L1 C/A signal started on channel 5 for satellite GPS PRN
08 (Block IIF)

```

Figure 29 Start of Decoding

```

Position at 2032-Nov-18 06:24:07.000000 UTC using 5 observations is Lat =
41.274839237 [deg], Long = 1.987693845 [deg], Height = 74.283 [m]
Velocity: East: -0.615 [m/s], North: 0.327 [m/s], Up = -1.359 [m/s]
Current receiver time: 1 min 8 s
New GPS NAV message received in channel 6: subframe 5 from satellite GPS
PRN 20 (Block IIR)
New GPS NAV message received in channel 0: subframe 5 from satellite GPS
PRN 01 (Block IIF)
New GPS NAV message received in channel 3: subframe 5 from satellite GPS
PRN 11 (Block IIR)
New GPS NAV message received in channel 1: subframe 5 from satellite GPS
PRN 32 (Block IIF)
New GPS NAV message received in channel 2: subframe 5 from satellite GPS
PRN 17 (Block IIR-M)
Position at 2032-Nov-18 06:24:08.000000 UTC using 5 observations is Lat =
41.274837899 [deg], Long = 1.987693711 [deg], Height = 74.587 [m]
Velocity: East: -0.557 [m/s], North: -0.761 [m/s], Up = 1.337 [m/s]
Current receiver time: 1 min 9 s
Current receiver time: 1 min 10 s
Position at 2032-Nov-18 06:24:09.000000 UTC using 5 observations is Lat =
41.274836254 [deg], Long = 1.987692736 [deg], Height = 75.001 [m]
Velocity: East: -0.117 [m/s], North: -0.555 [m/s], Up = 2.006 [m/s]
Current receiver time: 1 min 11 s
Position at 2032-Nov-18 06:24:10.000000 UTC using 5 observations is Lat =
41.274833784 [deg], Long = 1.987692175 [deg], Height = 75.101 [m]
Velocity: East: -0.994 [m/s], North: -1.545 [m/s], Up = 1.862 [m/s]
Current receiver time: 1 min 12 s
Current receiver time: 1 min 13 s
Position at 2032-Nov-18 06:24:11.000000 UTC using 5 observations is Lat =
41.274833440 [deg], Long = 1.987691630 [deg], Height = 75.012 [m]
Velocity: East: 0.249 [m/s], North: -0.032 [m/s], Up = -0.224 [m/s]
Tracking of GPS L1 C/A signal started on channel 7 for satellite GPS PRN
14 (Block Decommissioned)
Current receiver time: 1 min 14 s
New GPS NAV message received in channel 6: subframe 1 from satellite GPS
PRN 20 (Block IIR)
New GPS NAV message received in channel 1: subframe 1 from satellite GPS
PRN 32 (Block IIF)

```

Figure 30 PVT Solution of GPS L1 data

This data was the foundation of our subsequent analyses, allowing us to assess the receiver's accuracy and the effectiveness of our anti-spoofing measures. By monitoring changes in the position, velocity, and time, we could identify potential instances of spoofing and validate the performance of our GNSS SDR setup.

#### **4.2 Generation and Analysis of Spoofing Report**

After the acquisition of PVT information, the next pivotal part of the experiment involved generating a spoofing report. This report served as a comprehensive overview of the system's response to potential spoofing threats, providing a detailed breakdown of the anti-spoofing measures' effectiveness.

The spoofing report was obtained through the Telecommand interface. This interface, connected via Transmission Control Protocol (TCP), allows for the remote control and monitoring of the GNSS-SDR receiver. To access Telecommand, the command “**telnet localhost 3333**” was executed. Once connected, the system was primed to receive and respond to the Telecommand inputs.

The spoofing report was then generated by issuing the command ‘**spoofer\_status**’ in the Telecommand interface. This command prompted the system to compile a report detailing the results of various spoofing checks. A visual representation of this report can be seen in the corresponding figure:

```

Command 'qstatus' from deb gnss-sdr (0.1.0-1sg-1ubuntu1)
See 'snap info <snapname>' for additional versions.
usamahameed@usamahameed-Latitude-E7250:~/gnss-sdr$ telnet localhost 33
33
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
spoofer_status
=====
-----
+++++++ GNSS Anti Spoofing Using GNSS SDR ++++++
Project by NESCOM
Supervised by: Dr Fahad Mumtaz Malik
Group Members:
  Usama Hameed (288160)
  Umer Bukhari (289931)
  Hassan Tariq (32167)
  Mohsin Ali (295858)
-----
===== Spoofing Detection Report =====
Position jump      :POSITION JUMP NOT DETECTED
Velocity check     :VELOCITY CONSISTENT
Static position check :STATIC POSITION CONSISTENT
Clock offset       :SPOOFING NOT DETECTED
Auxiliary peak tracking :SPOOFING NOT DETECTED
Power level Attack :SPOOFING NOT DETECTED
Time jump          :SPOOFING NOT DETECTED
-----
===== End Report =====
Connection closed by foreign host.

```

Figure 31 Spoofing Detection Report

The spoofing report covers several important tests:

**Position Jump Test:** This test checks for sudden, improbable jumps in the receiver's position, which could indicate a spoofing attack.

**Velocity Check:** This test ensures that the velocity measurements are consistent with the receiver's actual motion, helping detect potential spoofing attempts that cause artificial motion detections.

**Static Position Check:** When the receiver is known to be stationary, this check validates whether the reported position remains constant.

**Clock Offset Test:** This test examines the receiver's internal clock for any inconsistencies that may indicate a time spoofing attack.

**Auxiliary Peak Tracking Test:** This test looks for secondary peaks in the signal correlation, which can be a sign of multiple spoofing signals.

**Power Level Attack Check:** This check monitors for sudden increases in the signal power level, which may signify a power-level spoofing attempt.

**Time Jump Test:** This test monitors for sudden jumps in time, which can indicate a spoofing attack.

Each of these tests provides valuable insights into the system's robustness against various forms of spoofing attacks. Through a detailed analysis of the spoofing report, we can evaluate the effectiveness of our anti-spoofing measures implemented in the GNSS-SDR receiver. The following sections will discuss the interpretation of these results and their implications.



## **Chapter 5: CONCLUSIONS AND FUTURE WORK**

### **5.1 Conclusion**

This research has engaged in the study, analysis, and implementation of anti-spoofing techniques for GNSS using GNSS-SDR. It has effectively provided a comprehensive understanding of the operations of GNSS, GNSS SDR, and various spoofing techniques that could pose serious threats to the effective functionality of the GNSS system.

Through this study, we have realized the importance of secure GNSS signals in various fields, including military, aviation, telecommunications, and more. This research has examined the relevance of both hardware-based and software-based GNSS receivers. More notably, the significance of GNSS-SDR in the creation of flexible and cost-effective receivers that can easily be manipulated to counter spoofing threats has been explored.

The study delved into various GNSS spoofing techniques, namely Replay Spoofing Attack Strategies (RSA), Forgery Spoofing Attack Strategies (FSA), Estimation Spoofing Attack Strategies (ESA), and Advanced Spoofing Attack Strategies (ASA). The evolution of methodologies has progressed alongside technological advancements, resulting in increased sophistication and a heightened emphasis on the imperative nature of anti-spoofing techniques.

The discourse delved into various anti-spoofing techniques, including Doppler Shift-Based Methods, Consistency Check-Based Methods, Signal Parameter Statistics Analysis-Based Methods, Arrival Time and Arrival Time Difference-Based Methods, and Residual Signal Detection-Based Methods. Each of the aforementioned techniques exhibited distinctive benefits and presented promise in mitigating the hazard of Global Navigation Satellite System (GNSS) spoofing.

The empirical component of the study has additionally afforded the chance to apply various anti-spoofing methodologies. The efficacy of each method was evaluated and assessed for their ability to detect and mitigate spoofing threats. Various techniques, including Power Level Check, Carrier-to-Noise Density Ratio (C/No) Check, Position Consistency Check, Velocity Consistency Check, Abnormal Position Check, Clock Offset Check, and Clock Jump Test, have demonstrated encouraging outcomes in mitigating diverse types of spoofing attacks.

This work, therefore, conclusively demonstrates the potential of GNSS-SDR as an effective tool for countering spoofing threats. Through the successful implementation and testing of various anti-spoofing techniques, the resilience of GNSS-SDR against different spoofing methods is evident.

### **5.2 Future Work**

This research opens several paths for future investigations.

Improvement of the existing anti-spoofing techniques is one key area to be explored. While the current methods show promising results, refinement through sophisticated algorithms or machine learning approaches could enhance their detection and mitigation capabilities.

Secondly, as technology continues to advance, newer and possibly more complex spoofing techniques may emerge. It's crucial to stay abreast of these advancements and adapt our anti-spoofing techniques accordingly. This calls for ongoing research and development in the realm of GNSS spoofing.

An exciting avenue for future work is the investigation of anti-spoofing techniques in multi-frequency receivers. The additional frequencies could enhance the resilience against spoofing and provide a more robust positioning solution.

Lastly, and significantly, the integration of GNSS anti-spoofing techniques on Field Programmable Gate Arrays (FPGA) should be explored. The real-time operation and parallel processing capabilities of FPGA could provide a more efficient implementation of these techniques. FPGAs can handle multiple processing tasks concurrently, thereby potentially increasing the system's ability to detect and respond to spoofing attacks swiftly. Moreover, the flexible nature of FPGA design allows for easy updates and improvements as new spoofing techniques arise.

In conclusion, this study has highlighted several key aspects of GNSS spoofing and its countermeasures, yet many dimensions remain to be explored. Future research endeavors should strive to enhance the security and reliability of GNSS, ultimately fostering the creation of more secure GNSS systems for the benefit of all users.

## REFERENCES

- [1] L. Scott, "Anti-spoofing & authenticated signal architectures for civil navigation systems," in Proc. 16th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GPS/GNSS), Portland, OR, USA, Sep. 2003, pp. 1543–1552.
- [2] L. Huang, Z. Lv, and F. Wang, "Spoofing pattern research on GNSS Receivers," J. Astronaut., vol. 33, no. 7, pp. 884–890, Jul. 2012.
- [3] K. Wesson, M. Rothlisberger, and T. Humphreys, "Practical cryptographic civil GPS signal authentication," Navigation, vol. 59, no. 3, pp. 177–193, Sep. 2012.
- [4] T. E. Humphreys, "Detection strategy for cryptographic GNSS antispoofing," IEEE Trans. Aerosp. Electron. Syst., vol. 49, no. 2, pp. 1073–1090, Apr. 2013.
- [5] J. T. Curran and C. O'Driscoll, "Message authentication, channel coding & anti-spoofing," in Proc. 29th Int. Tech. Meeting Satell. Division The Inst. Navigat. (ION GNSS), Portland, OR, Sep. 2016, pp. 2948–2959.
- [6] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," Proc. IEEE, vol. 104, no. 6, pp. 1258–1270, Jun. 2016
- [7] A. Jovanovic, C. Botteron, and P.-A. Fariné, "Multi-test detection and protection algorithm against spoofing attacks on GNSS receivers," in Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS), Monterey, CA, USA, May 2014, pp. 1258–1271
- [8] W. Qi, Y. Zhang, and X. Liu, "A GNSS anti-spoofing technology based on Doppler shift in vehicle networking," in Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC), Paphos, Cyprus, Sep. 2016, pp. 725–729
- [9] D. Yuan, H. Li, F. Wang, and M. Lu, "A GNSS acquisition method with the capability of spoofing detection and mitigation," Chin. J. Electron., vol. 27, no. 1, pp. 213–222, Jan. 2018
- [10] J. Tu, X. Zhan, M. Chen, H. Gao, and Y. Chen, "GNSS intermediate spoofing detection via dual-peak in frequency domain and relative velocity residuals," IET Radar, Sonar Navigat., vol. 14, no. 3, pp. 439–447, Mar. 2020
- [11] L. Yao, Z. Geng, Y. Su, and J. Nie, "The characteristics of single antenna repeater jamming coordinates mappings," GNSS World China, vol. 40, no. 5, pp. 19–24, Oct. 2015
- [12] A. Broumandan, A. Jafarnia-Jahromi, S. Daneshmand, and G. Lachapelle, "Overview of spatial processing approaches for GNSS structural interference detection and mitigation," Proc. IEEE, vol. 104, no. 6, pp. 1246–1257, Jun. 2016
- [13] K. D. Wesson, J. N. Gross, T. E. Humphreys, and B. L. Evans, "GNSS signal authentication via power and distortion monitoring," IEEE Trans. Aerosp. Electron. Syst., vol. 54, no. 2, pp. 739–754, Apr. 2018
- [14] A. Broumandan, R. Siddakatte, and G. Lachapelle, "Feature paper: An approach to detect GNSS spoofing," IEEE Aerosp. Electron. Syst. Mag., vol. 32, no. 8, pp. 64–75, Jun. 2017.
- [15] D. Borio, "PANOV tests and their application to GNSS spoofing detection," IEEE Trans. Aerosp. Electron. Syst., vol. 49, no. 1, pp. 381–394, Jan. 2013
- [16] D. Borio and C. Gioia, "A sum-of-squares approach to GNSS spoofing detection," IEEE Trans. Aerosp. Electron. Syst., vol. 52, no. 4, pp. 1756–1768, Aug. 2016
- [17] P. Y. Hwang and G. A. McGraw, "Receiver autonomous signal authentication (RASA) based on clock stability analysis," in Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS), Monterey, CA, USA, May 2014, pp. 270–281.
- [18] D. Yuan, H. Li, and M. Lu, "A method for GNSS spoofing detection based on sequential probability ratio test," in Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS), Monterey, CA, USA, May 2014, pp. 351–358.
- [19] F. Wang, H. Li, and M. Lu, "ARPSO-MLE based GNSS anti-spoofing method," in Proc. IEEE Int. Conf. Signal Process., Commun. Comput. (ICSPCC), Ningbo, China, Sep. 2015
- [20] J. N. Gross, C. Kilic, and T. E. Humphreys, "Maximum-likelihood powerdistortion monitoring for GNSS signal authentication," IEEE Trans. Aerosp. Electron. Syst., vol. 55, no. 1, pp. 1–6, Feb. 2019.
- [21] Z. Zhang, X. Zhan, and Y. Zhang, "GNSS spoofing localization based on differential code phase," in Proc. Forum Cooperat. Positioning Service (CPGPS), Harbin, China, May 2017, pp. 338–344.
- [22] K. Ali, E. G. Manfredini, and F. Dovis, "Vestigial signal defense through signal quality monitoring techniques based on joint use of two metrics," in Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS), Monterey, CA, USA, May 2014, pp. 1240–1247
- [23] W. Yimin, L. Hong, and L. Mingquan, "Spoofing profile estimation based GNSS spoofing identification method for tightly coupled MEMS INS/GNSS integrated navigation system," IET Radar, Sonar Navigat., vol. 14, no. 2, pp. 216–225, Feb. 2020.
- [24] E. G. Manfredini, F. Dovis, and B. Motella, "Validation of a signal quality monitoring technique over a set of spoofed scenarios," in Proc. 7th ESA Workshop Satell. Navigat. Technol. Eur. Workshop GNSS Signals Signal Process. (NAVITEC), Noordwijk, The Netherlands, Dec. 2014, pp. 1–7
- [25] A. J. Jahromi, A. Broumandan, S. Daneshmand, G. Lachapelle, and R. T. Ioannides, "Galileo signal authenticity verification using signal quality monitoring methods," in Proc. Int. Conf. Localization GNSS (ICLGNSS), Barcelona, Spain, Jun. 2016, pp. 1–8.
- [26] A. Broumandan, A. Jafarnia-Jahromi, G. Lachapelle, and R. T. Ioannides, "An approach to discriminate GNSS spoofing from multipath fading," in Proc. 8th ESA Workshop Satell. Navigat. Technol. Eur. Workshop GNSS Signals Signal Process. (NAVITEC), Noordwijk, The Netherlands, Dec. 2016, pp. 1–10.

- [27] P. Risbud, N. Gatsis, and A. Taha, "Vulnerability analysis of smart grids to GPS spoofing," IEEE Trans. Smart Grid, vol. 10, no. 4, pp. 3535–3548, Jul. 2019
- [28] T.-H. Kim, C. S. Sin, S. Lee, and J. H. Kim, "Analysis of effect of antispoofing signal for mitigating to spoofing in GPS II signal," in Proc. 13th Int. Conf. Control, Autom. Syst. (ICCAS ), Gwangju, South Korea, Oct. 2013, pp. 523–526
- [29] M. Berardo, E. G. Manfredini, F. Dovis, and L. L. Presti, "A spoofing mitigation technique for dynamic applications," in Proc. 8th ESA Workshop Satell. Navigat. Technol. Eur. Workshop GNSS Signals Signal Process. (NAVITEC), Noordwijk, The Netherlands, Dec. 2016, pp. 1–7.
- [30] L. Zhao, Z. Miao, B. Zhang, B. Liu, G. Li, and X. Zhou, "A novel spoofing attack detection method in satellite navigation tracking phase," J. Astronaut., vol. 36, no. 10, pp. 1172–1177, Oct. 2015
- [31] S. Bhamidipati, T. Y. Mina, and G. X. Gao, "GPS time authentication against spoofing via a network of receivers for power systems," in Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS), Monterey, CA, USA, Apr. 2018, pp. 1485–1491
- [32] J. T. Curran and C. O'Driscoll, "Message authentication, channel coding & anti-spoofing," in Proc. 29th Int. Tech. Meeting Satell. Division The Inst. Navigat. (ION GNSS), Portland, OR, Sep. 2016, pp. 2948–2959
- [33] J. Arribas, *GNSS Array-based Acquisition: Theory and Implementation*, PhD Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, June 2012.
- [34] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, S. H. Jensen, *A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach, 1st edition*, Boston: Birkhäuser, November 2006
- [35] A. Broumandan, A. Jafarnia-Jahromi, S. Daneshmand, and G. Lachapelle, "Overview of spatial processing approaches for GNSS structural interference detection and mitigation," Proc. IEEE, vol. 104, no. 6, pp. 1246–1257, Jun. 2016.
- [36] L. Yao, Z. Geng, Y. Su, and J. Nie, "The characteristics of single antenna repeater jamming coordinates mappings," GNSS World China, vol. 40, no. 5, pp. 19–24, Oct. 2015