Università
della
Svizzera
italiana

Faculty
of
Informatics

Bachelor Thesis

May 19, 2019

Implementing GPS Spoofing Detection in a Software Receiver

Robert Jans

Abstract

Global Navigation Satellite System (GNSS) receivers are nowadays used in many applications, ranging from personal navigation devices to spacecraft navigation and weapon guidance. Therefore, reliable GNSS positioning is fundamental and failure to provide accurate positioning could cause catastrophic effects threatening vital infrastructures and even causing loss of human lives. There may be different reasons for unreliable GNSS signals, among which malicious attacks. A preliminary study was performed at USI on detection of spoofing attacks to *Global positioning system* (GPS) signals based on machine learning methods. These methods can detect spoofing attacks, and even classify them in three categories, with very high accuracy. The goal of this project is to implement one of these detection methods and integrate it into an existing open source *Software Defined Receiver*(SDR), namely GNSS-SDR.

Keywords: GNSS; GPS; Spoofing Detection

Advisor:
Prof. Miroslaw Malek
Co-advisor:
Dr. Alberto Ferrante

Approved by the advisor on Date: Prof.

Contents

1 Introduction

GNSS systems are able to determine a receiver's position by comparing time signals emitted by satellites. Basically from the time differences the distance of a satellite can be calculated, then the position is computed using triangulation of the distances to several satellites. A detailed description of the inner workings of GNSS systems goes beyond the scope of this project, but useful information can be found e.g. on Wikipedia.

In real-world application there are two fundamentally distinct kinds of GNSS receivers: hardware receivers and software defined receivers. Both give as output an estimate of the receiver's position in terms of longitude, latitude and height (in some cases also velocity and acceleration are determined), in addition to some metadata including timestamps; the difference lies in the implementation: in the former kind the algorithms needed for extrapolating the positioning data from the raw signals are built into the hardware of an embedded device, while in the latter kind the raw signals are given as input to a software program which performs the post-processing.

This project focuses on a software defined receiver for the main reason that it is easily modifiable, so the spoofing detection feature can be implemented by writing software to be integrated into the existing program. The choice of using GNSS-SDR is based on the the fact that it is open-source and widely used for research purposes.

The final result is a modified version of GNSS-SDR which adds a warning message to its output in case a spoofing attack is detected.

2 Background

The project builds upon research conducted at the *Advanced Learning and Research Institute* (ALaRI), Faculty of Informatics, Università della Svizzera italiana (USI). The main reference and staring point of my work is the report "GAD: Machine Learning GNSS Attack Detection" proposed in 2017 by Dr. Alberto Ferrante.

The GAD report describes several methods for spoofing detection based on classifiers which take as input features some properties of the raw signals as well as some of the processed data.

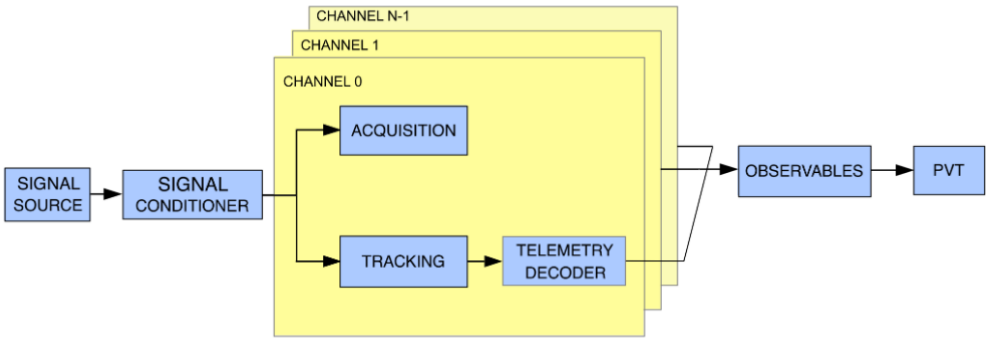
Due to the limitations in development time, this project considers only one of them: The J48 decision tree which gave best results on general attack detection. Future work may include the integration of other classifiers in order to distinguish the different attack types.

J48 is a general purpose decision tree algorithm implemented in Java and included in the WEKA project. J48 is an implementation of the C4.5 algorithm developed in 1993 by Ross Quinlan [2], who published a book about it in 1993 . A review of the book consisting in a description of the algorithm was written by Steven L. salzberg [3].

Much of my implementation is inspired by a comparative study published by H. M. E. Hssina, which describes in detail the algorithm and compares it with its predecessor ID3 [1].

2.1 High-level structure of GNSS-SDR

GNSS-SDR is a software written in C++ which is able to process several channels of GNSS data in parallel. It has a modular structure which makes it highly configurable (e.g. for using different signal sources). At a high level the program can be described as a sequence of processing blocks implementing a data flow architecture.



For each block there are several implementations; which implementation to use at which processing stage can be defined in a configuration file. For instance as signal source implementations there are available modules designed for specific hardware front-ends as well as file signal sources (For this project only file signal sources are used, but as a further work it could be considered to test the system with real time signals and spoofing devices).

2.2 Retrieving the features for the classifier

In GNSS-SDR after the acquisition stage, per-channel information is stored and updated in an object of a class called Gnss_Synchro and propagated through the blocks up to the PVT, where the output is generated. The original version of Gnss_Synchro already contains some features needed, or information based on which the features can be computed. An important exception is the amplitude of the raw signal, which is available at the acquisition stage but not stored in the Gnss_Synchro. the strategy to deal with this issue is to add a field to the mentioned class in order to store and propagate the missing information.

With this setting all features needed for the classifier can be retrieved from the Gnss_Synchro objects at the PVT stage. More on this will follow in the section dedicated to the implementation.

2.3 The C4.5 decision tree algorithm

2.3.1 About decision trees

Decision trees are popular tools in machine learning and data mining, mainly due to their simplicity and efficiency: though being structurally simple and not not excessively expensive in terms of computational time, they often return good results on data based classification problems. The basic concept is to start with a root node containing the entire training set and splitting it on some criteria. the process then is carried out recursively on the child nodes until some base case is reached; at this point a lea node representing a decision is created.

The C4.5 algorithm uses the information theoretical concept of entropy as splitting criterion: the set is split based on the feature that provides the greatest amount of information about the set with respect to the classification.

2.3.2 Concepts from information theory

C4.5 uses the concepts of *Entropy* and *Gain*, which are defined as follows:

Entropy given a probability distribution P and a sample S of size n

$$H(P) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

Gain With D being the the set to split and D_1, D_2 the candidate subsets, the gain is the difference in of the entropy of D and the weighted sum of the entropies of D_1 and D_2:

$$G(D, D_1, D_2) = H(D) - H(D_1) \cdot \frac{|D_1|}{|D|} - H(D_2) \cdot \frac{|D_2|}{|D|}$$

In the C4.5 the gains must be calculated for splits on all possible attribute-value pairs, the pair which leads to the highest gain is selected as decision condition for the given node.

2.3.3 Steps of the algorithm

base case 1 all elements of the set belong to the same class: create a leaf with the given class as decision result.

base case 2 the gain is zero or below a defined threshold: transform the parent node into a leaf, with the majority class as decision value.

base case 3 (optional) the tree has reached a defined maximum depth: create a leaf node with the majority class as decision value.

base case 4 (optional) the set is smaller than a defined minimum size: create a leaf node as above.

recursive case iterate over all attributes, for each attribute iterate over the values. for each attribute-value pair compute the gain and choose the best one. Apply the procedure to the two subsets obtained by splitting on the selected attribute-value pair.

Every time an attributeis selected remove it from the attributes on which to test in the child nodes.

2.4 The TEXBAT data set

the *Texas Spoofing Test Battery* TEXBAT is a set of recorded spoofed and non spoofed GNSS traces which is freely available at http://radionavlab.ae.utexas.edu/datastore/texbat/. The data used for training, valdating and testing of this project's implementations is taken from parts of TEXBAT.

Researchers at the ALaRI institute have prepared csv files containing records from TEXBAT labeled as "spoofed" or "clean", which is the basis for the supervised learning procedure.

3 Approach

3.1 General outline

The main part of my work is extracting the required features from GNSS-SDR, building a special purpose C++ version of the algorithm, adapted to the specific features used in the ALaRI research, and integrate it into GNSS-SDR.

In order to train the classifier, ALaRI provided a labeled data-set based on *TEXBAT* for supervised learning. In order to achieve a light-weight implementation capable of running on embedded devices with limited resources, the idea is to build the decision tree offline and encode it in a file, from which a minimal tree (containing only the decision attribute and threshold value for each node) can be loaded to run in real-time.

3.2 Feature retrieval

For using a classifier, at some point in the data flow of GNSS-SDR all the required features need to be assembled and made available to the classifier. In the case treated by this project (corresponding to the tests made at ALaRI using J48 with16 features), the following features are needed:

- minimum Doppler
- average Doppler
- standard deviation of Doppler
- minimum number of valid satellites
- average number of valid satellites
- standard deviation of the number of valid satellites
- maximum number of satellites changed
- standard deviation of the number of satellites changed
- minimum signal to noise ratio
- maximum signal to noise ratio
- average signal to noise ratio
- standard deviation of signal to noise ratio
- minimum pseudorange
- average pseudorange
- standard deviation of pseudorange
- maximum carrier phase

At the PVT processing stage some of these features are available in the incoming Gnss_Synchro objects, others are available in the PVT output, yet others can be computed either by combining available values or keeping track of values over time (for minima, maxima, averages and standard deviations). Details are described in the implementation section.

3.3 Building the C4.5 decision tree

A separate program takes as input the csv containing the labeled data and outputs a file containing an encoded form of the decision tree. The program must be able to run independently from GNSS-SDR since the processing time can be large given large input files.

3.4 Create and integrate the classifier

A program representing the decision tree must be built given a file with the encoded tree. This program must use the features extracted at run-time and provide a classification decision. It needs to interact directly with GNSS-SDR in order to include warnings in the outputs of the PVT.

3.5 Validation and testing

in order to tune the hyper-parameters of the tree-generating program, experiments need to be done on a validation set. Performance assessments are to be done on a separate testing data set. Luckily the available data-set is large enough to allow splitting it into disjoint subsets, leaving the training set large enough.

4 Implementation

WORK IN PROGRESS

5 Evaluation

5.1 Results

The experimental result goes here ... [1].

TO BE DONE

[1] https://www.usi.ch

6 Future work

7 Summary

Future works goes here.

References

[1] H. E. M. E. Badr HSSINA, Abdelkarim MERBOUHA. A comparative study of decision tree id3 and c4.5.

[2] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[3] S. L. SALZBERG. Book review: C4.5: by j. ross quinlan.inc., 1993.