Software defined radio based Global Navigation Satellite System real time spoofing detection and cancellation

Jean-Michel Friedt JMFRIEDT@FEMTO-ST.FR
FEMTO-ST Time & Frequency, 15 avenue des Montboucons, 25000 Besançon, France

David Rabus DAVID.RABUS@FEMTO-ST.FR
FEMTO-ST Time & Frequency, 15 avenue des Montboucons, 25000 Besançon, France

Gwenhael Goavec-Merou GWENHAEL.GOAVEC@FEMTO-ST.FR
FEMTO-ST Time & Frequency, 15 avenue des Montboucons, 25000 Besançon, France

Abstract

The software defined radio (SDR) global navigation satellite system decoder `gnss-sdr`, based on GNU Radio, is used to prototype computationally efficient global positioning system (GPS) anti-spoofing and anti-jamming strategies using controlled radiation pattern antenna and null steering towards a unique interfering source. Experimental demonstration is achieved on GPS L1 when running `gnss-sdr` on a Raspberry Pi4 single board computer fetching radiofrequency datastreams from an Ettus Research B210 SDR frontend.

1. Introduction

Global Navigation Satellite Systems (GNSS) in general, and the American Global Positioning System (GPS) in particular, has become ubiquitous to most daily activities including positioning, navigation and timing (PNT) with major economic and financial losses if such services are denied or, worse, spoofed (Economics, 2017). While the former jamming attack is trivially detected with a loss of service, the latter is more subtle since the user is fed erroneous, seemingly genuine, streams of information (Ioannides et al., 2016).

While GNSS constellations and modulation schemes are quickly evolving, the focus of the present work is the GPS L1 (1575.42 MHz) band. Initially deployed at the end of the 1970s, this technology is well documented and requires sufficiently low bandwidth to be readily addressed with low-power, narrow bandwidth (2.2 MS/s) Software Defined Radio frontends. Furthermore, the spectrum spreading Binary Phase Shift Keying (BPSK) is cancelled using codeless processing techniques, allowing for classical Continuous Wave (CW) direction of arrival (DoA) computations to be applied once the raw radiofrequency signal has been squared to remove the modulation. The unique attribute of SDR with respect to hardware GNSS receivers is hence the access to the raw radiofrequency signal before any processing has been applied, and hence the ability to analyze the physical properties of the incoming wave. While hardware receivers can always be spoofed with well enough designed synthetic signals (Psiaki & Humphreys, 2016) , the physics of DoA analysis cannot be simulated with a single attack source, while distributed attacks with multiple synchronized emitters is beyond the scope of this investigation.

Once spoofing has been detected (Schmidt et al., 2018) by analyzing the physical properties of the electromagnetic waves associated with each satellite transmission, genuine signal recovery is desirable. Cancelling spoofing signals using low complexity processing techniques (Daneshmand et al., 2012) is also applicable to jamming in order to recover the genuine signal, despite more complex signal processing due to the lack of structure of the BPSK modulated signal in the case of jamming.

While demonstrating the concepts in a post-processing analysis, real time jamming and spoofing cancellation is desirable to demonstrate recovery of the original functionality of GNSS constellations. A free, opensource implementation of GNSS decoders processing SDR datastreams is provided with `gnss-sdr` (`https://gnss-sdr.org/`, 2020), a set of processing blocks running on top of GNU Radio. Thus, developers familiar with the GNU Radio Out of Tree (OOT) module structure will find the architecture of `gnss-sdr` intuitive, with each processing block made of the constructor and the work function fed multiple datastreams and in our case acting as a single output stream, namely the processed output of multiple an-

Proceedings of the 1st GNU Radio Conference, Copyright 2016 by the author(s).

tenna inputs. Our modifications in `gnss-sdr` focus on `src/algorithms/signal_source/adapters/` and specifically the File and UHD sources to accept input streams from multiple antennas (File for simulating acquisitions from multiple antennas for post-processing, and UHD to collect data from both inputs of the B210), and the addition of processing blocks in `src/algorithms/signal_source/libs` for spoofing detection and cancellation as well as jamming cancellation. All contributions are available at `https://github.com/oscimp/gnss-sdr`, the fork of the original project running on the latest revision, namely 0.13 at the date of writing this manuscript.

A general presentation of our understanding of the `gnss-sdr` structure and means of adding a custom processing block is summarized at https://github.com/oscimp/gnss-sdr-custom. As is familiar to OOT GNU Radio developers, the processing block constructor is instanciated as `my_block=gnss_custom_make();` with the object being defined as `boost::shared_ptr<gr::block> my_block;`, and connected to a signal source, for example the UHD output, with `top_block->connect(uhd_source_, 0, my_block, 0);`. The constructor defines the object initialization `boost::shared_ptr<custom_Gnss> gnss_custom_make()` and the main processing function `int custom_Gnss::work(int noutput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items) { ... return noutput_items;}` is fed the I/Q stream array `input_items` to be processed with `noutput_items` elements, generating the output stream `output_items`, where "..." represents the digital signal processing operations. The only originality of the block hierarchy in `gnss-sdr` is that the last block in a processing chain must be defined so the next step can be linked to it: the function `gr::basic_block_sptr UhdSignalSource::get_right_block(int RF_channel)` returns this last processing block of the Signal Source with `return my_block;`.

The paper is organized as follows: a brief reminder of SDR GPS L1 spoofing is given to demonstrate the ability to spoof commercial, off the shelf civilian single frequency GPS receivers with readily available hardware (Analog Devices Inc. PlutoSDR) (Huang & Yang, 2015; Zeng et al., 2018). Having demonstrated GPS spoofing, we demonstrate spoofing detection by analyzing the phase of the despread BPSK carriers. Having identified spoofing, we cancel the signal from the direction of arrival identified with phase analysis and this null steering allows for recovering the genuine constellation signal. Finally, least-square weight calculation allows for jamming source DoA identification and again, null steering allows for jamming cancellation and recovery of the genuine constellation signals. All demonstrations are performed with the dual coherent channel AD9361 radiofrequency frontend fitted in the Ettus Research B210 SDR platform, with software either running on low-grade laptop computers (Panasonic CF-19 fitted with 12 GB RAM) but focusing on computational efficiency to execute on the single board computer Raspberry Pi 4 running a custom flavor of GNU/Linux compiled using Buildroot. The benefit of the custom compilation framework and the speed improvement by using the specific acceleration features of the processor not used by general purpose binary GNU/Linux distributions is emphasized in the last section.

Throughout this presentation, the genuine location should be considered as the location of the laboratory these experiments were run in, namely in Besançon, France located around 47.241_5 °N, 5.987_2 °E whereas the spoofed location was located in Western France in Brittany at 48.362_1 °N, 4.822_3 ° W in the English Channel, selected to be in view of a similar GPS constellation yet clearly off the real position.

2. Spoofing

A pre-requisite to spoofing detection and cancellation is demonstrating spoofing capability. While SDR implementations of GPS L1 spoofing is readily available, including using the AD9363 radiofrequency frontend fitted in the PlutoSDR evaluation board at https://github.com/Mictronics/pluto-gps-sim, we have observed poor performance on commercial GPS receivers such as those fitted in car navigation systems.

We have attributed this poor performance to the low quality Temperature Compensated Crystal Oscillator (TCXO) clocking the PlutoSDR with respect to the expected oscillator behavior of the atomic clocks on board GPS satellites (G. Goavec-Merou, 2019). While the hydrogen maser or rubidium microwave atomic transition allow for long term oscillator correction to keep the frequency accurate, short term ($<$1 h) atomic clock stability and phase noise is solely driven by the quartz oscillator generating the microwave signal from which all other clock signals are derived. Typical atomic clock local quartz oscillators are Oven Controlled Crystal Oscillators (OCXO) such as the Hewlett Packard 10811-60111 (Fig. 1). Not only are such oscillators available from refurbished instruments as second hand, but these older oscillators have completed their aging process and finished drifting over time and are hence well suited to mimicking the short to medium term stability of atomic clocks.

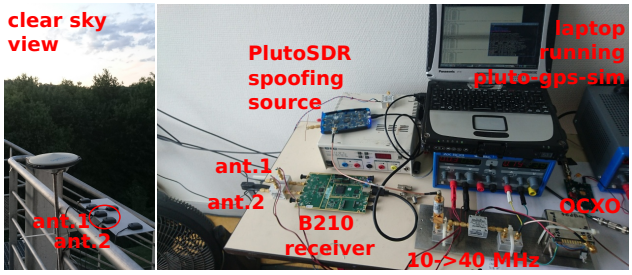

Figure 1. Experimental setup. The two L1-patch antennas on the left used for spoofing detection are separated by 10 cm of half-wavelength of the GPS signal. The distance from spoofing source to the outdoor antennas is about 3 m.

The AD9361 in the PlutoSDR is clocked with a 40 MHz reference while most OCXO output a 10 MHz signal. We have observed the phase noise of the AD9361 output to be degraded if the reference oscillator clocking this radiofrequency frontend is below 30 MHz. Hence, a double frequency-doubler circuit (MiniCircuits MK-2) is included between the OCXO output and the AD9361 clock input, with low noise ZFL-1000LN+ amplifiers bringing the signal power high enough to drive the non-linear frequency doublers. Thanks to this clocking scheme, all tested civilian single-channel L1 GPS receivers have been spoofed to a location a few hundred kilometers away from the genuine location (Fig. 2), with the receiver exposed to a clear sky view of the GPS constellation, with best results achieved when the ephemeris of the current constellation geometry is used after downloading the most recent configuration file from one of the International GNSS Services as described at https://kb.igs.org/hc/en-us/articles/202054393-IGS-FTP-Sites.

3. Spoofing detection

Having demonstrated the spoofing capability, spoofing detection is addressed by analyzing the direction of arrival of the GPS L1 signal. The assumption will be a single spoofing source, considering that a distributed attack requires multiple coherent emitters, a feat beyond most attackers. Since the physical property of the electromagnetic waves received by the B210 dual channel receiver two antennas connected to both inputs is investigated, the spectrum spreading BPSK modulation bringing the GPS signal below thermal noise power will be cancelled in the classical codeless analysis processing. Indeed, a BPSK signal switches the carrier phase to 0 or π depending on the bit state: the two information carried by the carrier are the space vehicle pseudo-random code identifier (PRN) at a rate of 1.023 Mb/s, and the navigation message at a rate of 50 b/s. These two informations are XORed so that applying

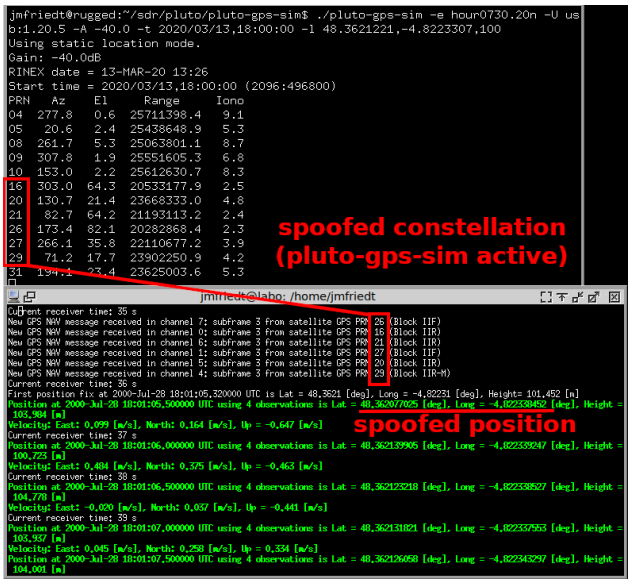

Figure 2. Spoofing demonstration. The displayed location at 48.3°N, 4.8°W, in the English Channel, is the spoofed location about 800 km west of Besançon (France) where the receiver is located.

the known PRN associated with each space vehicle allows for recovering the navigation message transmitted by each satellite and hence, using trilateration, locate the receiver position. Squaring the BPSK signal doubles the argument and the modulation becomes 0 or $2\pi = 0$ so that the spectrum spreading modulation has been removed. Each satellite is still identified by the different Doppler shift introduced by the different elevation and azimuth of each satellite: even a spoofer must introduce such frequency shifts on the signal simulating each satellite transmission to provide a credible spoofing signal. Having now a multitude of peaks in the Fourier transform of the squared signal, each located at double the Doppler shift (due to squaring the signal and hence doubling the argument), an analysis of the phase difference between both antennas is achieved by considering either the product of the Fourier transform of the squared signal received from one antenna with the complex conjugate of the Fourier transform of the squared signal received by the second antenna, or the ratio of these Fourier transforms. In both cases, the subtraction of the arguments of both Fourier transforms will cancel the common Doppler shift term and only the geometric phase difference representative of the path difference for the plane wave to reach both antennas is left.

Spoofing is thus detected if all geometric phases associated with the direction of arrival of the unique spoofing source is identified for all Doppler shifts associated with each satellite. Such a condition is physically impossible since dif-

ferent Doppler shifts from spaceborn signals is necessarily associated with different positions in the sky. On the other hand, a high standard deviation on the detected geometric phase is the signature of a genuine constellation signal: this threshold on the geometric phase standard deviation triggers the spoofing cancellation algorithm implemented in our fork of the `gnss-sdr` software available at https://github.com/oscimp/gnss-sdr. The user is informed of spoofing when a `/!\` symbol is appended to the messages provided by the GNSS decoding software.

4. Spoofing cancellation

Controlled Radiation Pattern Antenna (CRPA) processing aims at identifying the spoofing signal properties on an antenna array and cancel this contribution to recover the genuine signal, which is equivalent to electronic null steering by processing the I/Q stream recovered by both antennas. Since spoofing signal cancellation is performed after collecting the raw signal from each antenna (as opposed to tuning the phase of an analog phase shifter and cancelling the spoofing signal prior to data acquisition), this approach is only limited by the SDR frontend dynamic range: while typical GPS receivers will process a single or 2-bit encoded I/Q signals, CRPA requires high resolution and hence high communication bandwidth radiofrequency frontends to still encode the genuine GPS signal below the stronger spoofing signal.

While the product of Fourier transforms is used to readily identify the Doppler shift associated with each satellite since power accumulates, identifying the ratio of the spoofing signal on one antenna to the other antenna requires computing the ratio of the Fourier transforms (inverse filtering). Thus, computing

$$\frac{FFT(s_1^2)}{FFT(s_2^2)} = \frac{A_1^2}{A_2^2}\exp(j(2\delta\omega_1^{(j)}t - 2\delta\omega_2^{(j)}t + 2\varphi_{PRN1}^{(j)} - 2\varphi_{PRN2}^{(j)} + 2\varphi_1^{(j)} - 2\varphi_2^{(j)})) \quad (1)$$

with $\delta\omega_i^{(j)}$ the Doppler shift of satellite j on antenna i, $\varphi_{PRNi}^{(j)}$ the BPSK message transmitted by satellite j as seen by antenna i, and $\varphi_i^{(j)}$ the geometric phase associated with satellite j at antenna i. Since the same satellite signal is detected by both antennas, the term $2\delta\omega_1^{(j)} - 2\delta\omega_2^{(j)}$ cancels and since the BPSK modulation vanishes on the squared signals, the terms $2\varphi_{PRN1}^{(j)} - 2\varphi_{PRN2}^{(j)}$ also vanish, only leaving

$$\frac{FFT(s_1^2)}{FFT(s_2^2)} = \frac{A_1^2}{A_2^2}\exp(2j(\varphi_1^{(j)} - \varphi_2^{(j)}))$$

twice the geometric phase between the two antennas so that the weight α to cancel the spoofing signal detected on antenna 2 from antenna 1 and generate a signal s cleaned (Fig. 3) from spoofing interference $s(t) = s_1(t) - \alpha s_2(t)$ is

$$\alpha = \sqrt{\frac{FFT(s_1^2)}{FFT(s_2^2)}}$$

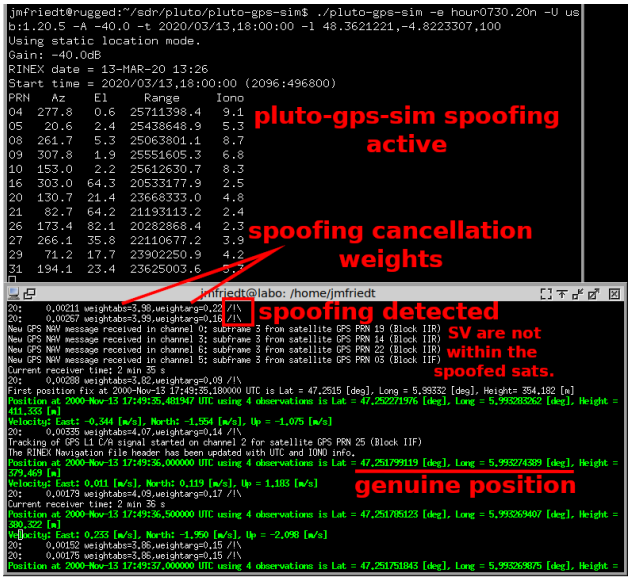

Figure 3. Spoofing detection and cancellation demonstration.

5. Jamming cancellation

Null steering in the occurrence of spoofing benefits from the assumption of BPSK modulation of the interfering signal and the ability to analyze a continuous wave carrier after squaring the received radiofrequency signal. Such an assumption is not valid in the general case of jamming in which, for example, a voltage controlled oscillator is swept at high rate over the frequency band in which GPS is emitting, preventing the genuine signal analysis (GPS jammer): this jammer sweeps at 300 kHz a microwave voltage controlled oscillator tuning voltage with a triangle wave, emitting 10 mW in the 1500 to 1600 MHz range. During our experiments, the antenna of this jammer was removed to reduce the range of the service denial to the power radiated by the printed circuit board traces to a few meters around the device. The same problem of identifying the weight of the jamming signal on one antenna to cancel its contribution from the second antenna is addressed as before, but this time as a least square error optimization problem. The general solution of this over-constrained problem is to consider the pseudo-inverse of s_1 applied to s_2 to identify α and thus cancel the jamming signal. The computation of the pseudo-inverse is however a computationally intensive operation hardly compatible with execution on an embed-

ded single board computer such as the Raspberry Pi 4 for real time streaming data processing.

The first approach implemented in our fork of `gnss-sdr` is to apply a stochastic gradient descent (Friedt et al., 2019) in which α is iteratively identified from the successive I/Q data collected on both antennas. While this method has been demonstrated as efficient on static targets, it appears unstable when the receiver or jammer are moving one with respect to the other.

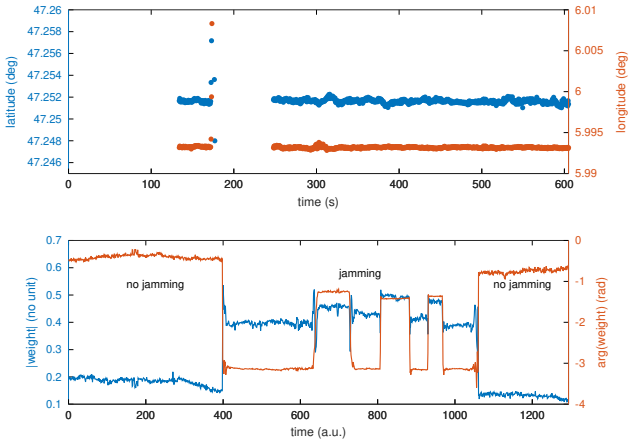

Figure 4. Jamming cancellation using inverse filtering (ratio of Fourier transforms of both channels) for weight estimate. The receiving antenna array is rotated by 90° multiple times to demonstrate the stability of the processing even when orientation of the receiving array is changing relative to the direction of arrival of the jamming plane wave.

An alternative approach for finding α as discussed in the spoofing identification section is to consider the ratio of the Fourier transforms of the signals, this time without preliminary squaring (Fig. 4). Indeed under the assumption that the jamming signal is above thermal noise, both channels will detect the same jammer contribution. The weight of the relative contribution is identified from the inverse filtering computed as the ratio of the Fourier transforms, acting similarly to a cross-correlation (product of the Fourier transform of the signal from one antenna with the complex conjugate of the Fourier transform of the signal collected by the second antenna), but taking the ratio of the magnitude rather than the product. This weight estimate has appeared robust to jamming source motion and to provide the best performance to recover the genuine signal by null-steering.

In the experiment depicted in Fig. 4, the genuine position is acquired in the first 120 seconds of cold-starting `gnss-sdr`. When the jamming signal is emitted, the position solution diverges and is lost: another 60 seconds are needed to acquire again the constellation signal and track the position. However once this position has been found, rotating the receiving antenna array by 90° as seen on the phase of the bottom chart (dates 610 and 720 for slow rotations, 800, 900, 950 and 980 for rotations as fast as possible from a radio controlled servo motor) does not induce tracking loss. Switching off the jammer (date 1050) also keeps the tracking loop locked. No significant deviation on the position (top chart) is visible whether jamming cancellation is active or the genuine signal is acquired. In this example, the jamming cancellation algorithm is activated on a threshold condition on the weight magnitude (bottom graph, blue curve clearly exhibiting the rise of the magnitude of the weight when jamming is activated at date 180 and deactivated at date 1050).

6. Embedded platform

An attractive development of GNU Radio and hence `gnss-sdr` is the ability to run on compact, single board computers such as the Raspberry Pi 4. The computational load for acquiring 12-bit complex data at 1.123 MS/s since we increase by 10% the theoretical signal bandwidth, and running anti-spoofing or anti-jamming algorithms for real time decoding of the genuine signal remains high. The 1.5 GHz Raspberry Pi 4 quad-core processor must be run at full speed (`ondemand` or `performance` power consumption settings) and all optimizations of the processor must be used, including the Single Instruction Multiple Data (SIMD) extension heavily used by the Vector Optimized Library of Kernels (VOLK) library (Fig. 5). Such functionalities are not accessible with general purpose binary distributions such as the Raspbian distribution (now called Raspberry Pi OS) targeting all flavors of the single board computer and hence lacking optimization. As a demonstration of this statement, table 1 exhibits a few of the VOLK function execution time calibrated with `volk_config`.

Speed improvements of factors of 3 (`volk_16ic_deinterleave_real_8i`) to 6 (`volk_16i_convert_8i`) are observed when using NEON SIMD instructions with respect to the generic implementation using general purpose instructions, and generic implementations exhibit equivalent execution speed on both operating systems. The benefit of compiling a custom Buildroot image optimized for a given processor is thus clear as opposed to using a generic binary image.

7. Conclusion

We demonstrate the implementation of Controlled Radiation Pattern Antenna (CRPA) using Software Defined Radio (SDR) implementation of Global Navigation Satellite System (GNSS) decoders as provided by `gnss-sdr`, a su-

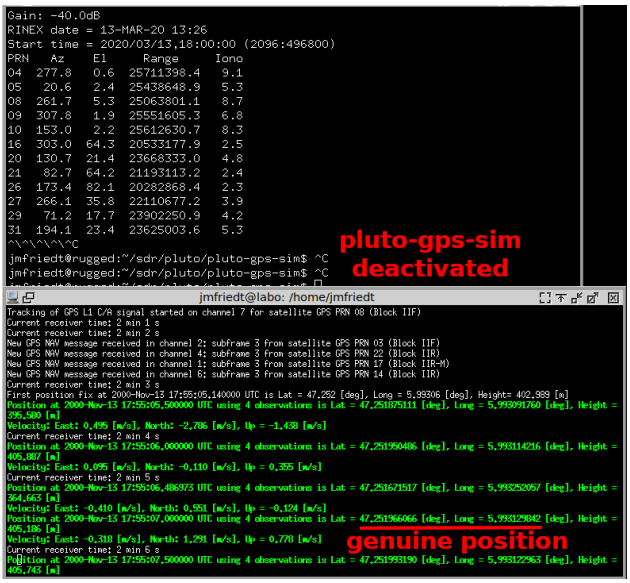

Figure 5. Genuine position measurement with `gnss-sdr` running on the Raspberry Pi 4 single board computer.

perset to the GNU Radio framework. Doing so, real time recovery of the genuine signal is achieved under spoofing and jamming conditions that would prevent locating the user at the right position: the former attack is not detected the leads to the wrong positioning of the receiver, while the latter leads to a loss of service and is readily detected. Having focused on the narrowband (1.023 MS/s complex datasteam) GPS L1 BPSK-modulated signal, further investigations will now focus on the broadband (L5/E5) lower frequency signal with the more complex BOC modulation requiring more complex analysis to recover the carrier properties. All software is available at https://github.com/oscimp/gnss-sdr for the experiment to be reproduced.

Acknowledgements

We acknowledge fruitful discussions with W. Feng (National Laboratory of Radar Signal Processing, Xidian University, Xi'an, China). This work is supported by the FAST-LAB joint laboratory between the FEMTO-ST Institute (Besançon, France) and the company Gorgy Timing (La Mure, France). Technical support from the Oscillator Instability Measurement Platform (OscIMP) and the FIRST-TF grants from the French National Research Agency (ANR) is acknowledged.

References

Daneshmand, Saeed, Jafarnia-Jahromi, Ali, Broumandan, Ali, and Lachapelle, Gérard. A low-complexity GPS anti-spoofing method using a multi-antenna array. In *Proc. 25th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2012)*, pp. 1233–1243, 2012.

Economics, London. The economic impact on the UK of a disruption to GNSS. *Showcase Final Report, UK Space Agency*, pp. 3–5, 2017. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/619544/17.3254_Economic_impact_to_UK_of_a_disruption_to_GNSS_-_Full_Report.pdf.

Friedt, J.-M., Feng, W., Chrétien, S., Goavec-Merou, G., and Sato, M. Passive radar for measuring passive sensors: direct signal interference suppression on FPGA using orthogonal matching pursuit and stochastic gradient descent. In *Multimodal Sensing: Technologies and Applications*, volume 11059, pp. 1105906. International Society for Optics and Photonics, 2019.

G. Goavec-Merou, J.-M Friedt, F. Meyer. Spoofing GPS. In *FOSDEM 2019 (Free Software Radio devroom)*, 2019. https://archive.fosdem.org/2019/schedule/event/sdr_gps/.

GPS jammer. https://www.amazon.fr/IrahdBowen-Bloqueur-Bouclier-Brouilleur-Disjoncte dp/B07KSC5LLD.

Huang, Ling and Yang, Qing. Low-cost GPS simulator GPS spoofing by SDR. In *Proc. DEFCON*, 2015.

Ioannides, Rigas Themistoklis, Pany, Thomas, and Gibbons, Glen. Known vulnerabilities of global navigation satellite systems, status, and potential mitigation techniques. *Proceedings of the IEEE*, 104(6):1174–1194, 2016.

Psiaki, Mark L and Humphreys, Todd E. GNSS spoofing and detection. *Proceedings of the IEEE*, 104(6):1258–1270, 2016.

Schmidt, Erick, Ruble, Zachary, Akopian, David, and Pack, Daniel J. Software-defined radio GNSS instrumentation for spoofing mitigation: A review and a case study. *IEEE Transactions on Instrumentation and Measurement*, 68 (8):2768–2784, 2018.

https://gnss-sdr.org/. GNSS-SDR. *Access online*, 2020.

Zeng, Kexiong Curtis, Liu, Shinan, Shu, Yuanchao, Wang, Dong, Li, Haoyu, Dou, Yanzhi, Wang, Gang, and Yang, Yaling. All your GPS are belong to us: Towards stealthy manipulation of road navigation systems. In *27th USENIX Security Symposium*, pp. 1527–1544, 2018.

Table 1. Execution duration of VOLK functions as calibrated by `volk_profile`, when running on a Raspberry Pi 4 single board computer executing a custom Buildroot generated image with the processing in powersave mode (600 MHz), same with the processor in performance mode (1500 MHz), and running a Raspberry Pi OS with the 64 bit kernel version with the processor on demand mode (1500 MHz when running the benchmark program).

Buildroot, powersave	Buildroot, performance	Raspbian, ondemand
volk_64u_popcntpuppet_64uu	**volk_64u_popcntpuppet_64u**	**volk_64u_popcntpuppet_64u**
generic completed in 7103.62 ms	generic completed in 3089.73 ms	no architectures to test
neon completed in 4038.24 ms	neon completed in 1897.77 ms	
Best aligned arch: neon	Best aligned arch: neon	
Best unaligned arch: neon	Best unaligned arch: neon	
volk_64u_popcntpuppet_64u	**volk_64u_popcntpuppet_64u**	**volk_64u_popcntpuppet_64u**
generic completed in 7154.26 ms	redgeneric completed in 3157.41 ms	no architectures to test
neon completed in 4106.08 ms	neon completed in 2081.84 ms	
Best aligned arch: neon	Best aligned arch: neon	
Best unaligned arch: neon	Best unaligned arch: neon	
volk_16ic_deinterleave_real_8i	**volk_16ic_deinterleave_real_8i**	**volk_16ic_deinterleave_real_8i**
generic completed in 1745.19 ms	generic completed in 697.845 ms	generic completed in 420.678ms
neon completed in 254.155 ms	neon completed in 105.462 ms	u_orc completed in 391.035ms
Best aligned arch: neon	Best aligned arch: neon	Best aligned arch: u_orc
Best unaligned arch: neon	Best unaligned arch: neon	Best unaligned arch: u_orc
volk_16ic_s32f_deinterleave_32f_x2	**volk_16ic_s32f_deinterleave_32f_x2**	**volk_16ic_s32f_deinterleave_32f_x2**
generic completed in 2258.27 ms	generic completed in 2185.24 ms	generic completed in 2211.99ms
neon completed in 1274.83 ms	neon completed in 728.173 ms	u_orc completed in 4766.13ms
Best aligned arch: neon	Best aligned arch: neon	Best aligned arch: generic
Best unaligned arch: neon	Best unaligned arch: neon	Best unaligned arch: generic
volk_16i_s32f_convert_32f	**volk_16i_s32f_convert_32f**	**volk_16i_s32f_convert_32f**
generic completed in 2181 ms	generic completed in 870.3 ms	generic completed in 749.928ms
neon completed in 697.446 ms	neon completed in 310.137 ms	a_generic completed in 750.233ms
a_generic completed in 2181.02 ms	a_generic completed in 870.304 ms	Best aligned arch: generic
Best aligned arch: neon	Best aligned arch: neon	Best unaligned arch: generic
Best unaligned arch: neon	Best unaligned arch: neon	
volk_16i_convert_8i	**volk_16i_convert_8i**	**volk_16i_convert_8i**
generic completed in 1745.56 ms	generic completed in 696.289 ms	generic completed in 457.922ms
neon completed in 134.038 ms	neon completed in 75.7975 ms	a_generic completed in 458.445ms
a_generic completed in 1745.59 ms	a_generic completed in 696.28 ms	Best aligned arch: generic
Best aligned arch: neon	Best aligned arch: neon	Best unaligned arch: generic
volk_32f_cos_32f	**volk_32f_cos_32f**	**volk_32f_cos_32f**
generic_fast completed in 51036.2 ms	generic_fast completed in 19325.9 ms	generic_fast completed in 22240.9ms
generic completed in 13673.1 ms	generic completed in 4678.62 ms	generic completed in 5470.72ms
Best aligned arch: generic	Best aligned arch: generic	Best aligned arch: generic
Best unaligned arch: generic	Best unaligned arch: generic	Best unaligned arch: generic