Real-Time Spoofing Detection Using Correlation Between Two Civil GPS Receiver

Brady W. O'Hanlon, Mark L. Psiaki, *Cornell University, Ithaca, NY*
Todd E. Humphreys, Jahshan A. Bhatti, *The University of Texas at Austin, Austin, TX*

BIOGRAPHY

Brady W. O'Hanlon is a Ph.D. candidate in the School of Electrical and Computer Engineering at Cornell University. He received both his M.S. and B.S. in Electrical and Computer Engineering from Cornell University. His interests are in the areas of GNSS technology and applications, GNSS security, and space weather.

Mark L. Psiaki is a Professor in the Sibley School of Mechanical and Aerospace Engineering. He received a B.A. in Physics and M.A. and Ph.D. degrees in Mechanical and Aerospace Engineering from Princeton University. His research interests are in the areas of GNSS technology, applications, and integrity, spacecraft attitude and orbit determination, and general estimation, filtering, and detection.

Todd E. Humphreys is an assistant professor in the department of Aerospace Engineering and Engineering Mechanics at the University of Texas at Austin and Director of the UT Radionavigation Laboratory. He received a B.S. and M.S. in Electrical and Computer Engineering from Utah State University and a Ph.D. in Aerospace Engineering from Cornell University. His research interests are in estimation and filtering, GNSS technology, GNSS-based study of the ionosphere and neutral atmosphere, and GNSS security and integrity.

Jahshan A. Bhatti is pursuing a Ph.D. in the Department of Aerospace Engineering and Engineering Mechanics at the University of Texas at Austin, where he also received his M.S. and B.S degrees. He is a member of the UT Radionavigation Laboratory. His research interests are in the development of small satellites, software-defined radio applications, space weather, and GNSS security and integrity.

ABSTRACT

A real-time method for detecting GPS spoofing in a narrow-bandwidth civilian GPS receiver has been implemented and tested, both in the absence of and in the presence of spoofing. The system was implemented as a software-defined radio system on a personal computer, using a pair of narrow-bandwidth radio front-ends that were geographically separated, with data transmitted between the two over the Internet.

The presence of a spoofing signal is determined by mixing and accumulating the base-band quadrature channel samples from the two receivers, with the aim of cross-correlating the P(Y) code that should be present in both signals in the absence of spoofing.

Several spoofing attacks were successfully detected in real-time.

I. INTRODUCTION

As the reliance of the civilian community on GPS signals for timing and positioning in mission-critical applications grows, so does the vulnerability to and potential cost of an attack via signal spoofing. GPS signal spoofing is a type of attack whereby a GPS receiver is fooled into tracking counterfeit signals, generally with the intention of misleading the receiver with regards to position, time, or both. In 2001 the U.S. Department of Transportation warned of the vulnerability of civilian GPS receivers to attacks such as spoofing[1], and such attacks have since been demonstrated by a variety of parties[2,3].

Given the potential damage a successful spoofing attack could inflict, detecting this kind of attack is of paramount importance. The spoofing detection method implemented here was proposed by Lo et al.[4], and is based on the presumed security of the encrypted P(Y) code. This method has previously been successfully implemented using two narrow-bandwidth civil receiver operating in a post-processing mode (i.e., not done in real-time, and using a software receiver written in MATLAB)[5,6]. This paper seeks to examine the efficacy of this method in the context of a narrow-bandwidth real-time software receiver using only those components that would normally be used in a civilian receiver (that is, using only a patch antenna rather than a high-gain antenna and no additional timing hardware).

The Lo method assumes that a spoofer can only spoof the C/A code, and in doing so thereby changes the relationship between the C/A code and P(Y) for a particular spoofed signal. Let us assume that there exists a "reference" receiver that is trusted (that is, the signals are believed genuine), and a "user equipment" receiver which may or may not be under a spoofing attack. If one can isolate that portion of the signal that should contain the P(Y) code from the reference receiver, a cross-correlation of this data and similar data from the user equipment receiver can be carried out. Only if the user equipment receiver is not being spoofed should there be a large correlation value due to the cross-correlation of the P(Y) code from both sets of data. Properly executing this cross-correlation requires isolating the portion of the signal that should contain the P(Y) code and temporal alignment of the two data streams.

A good discussion on the probable efficacy of several other spoofing detection methods can be found in the paper by Humphreys.

Section II of this paper contains a description of the hardware used in this work. Section III is an overview of the software used in implementing this spoofing detection method including the algorithm in general terms, derivation and analysis of the spoofing detection statistic threshold, and peculiarities that are necessary due to the particular software receiver used. Section VI contains initial results from testing the algorithm under a variety of conditions including when the UE receiver is being subjected to a spoofing attack. Section V contains a discussion of the results, including the possibility of spoofing of the reference receiver, and Section VI contains conclusions.

II. HARDWARE OVERVIEW

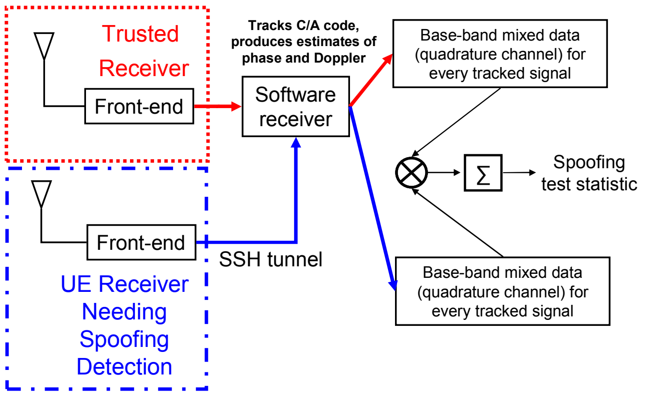
Trusted
Receiver
Front-end
Tracks C/A code, produces estimates of phase and Doppler
Software receiver
Base-band mixed data (quadrature channel) for every tracked signal
Σ
Spoofing test statistic
Front-end
SSH tunnel
Base-band mixed data (quadrature channel) for every tracked signal
UE Receiver Needing Spoofing Detection

Fig. 1. Spoofing Detection System Architecture

Data for this experiment was collected using custom designed radio-frequency front-ends (RFEs) paired with data acquisition units. The RFEs used have intermediate frequency filters with a bandwidth of 2.5 MHz, and produce 2-bit quantized data at a sampling frequency of approximately 5.7 MHz. The quantized data is recorded to a personal computer using a data acquisition peripheral and then transmitted over the internet from the user equipment receiver to the reference receiver, where all processing is done. This system is symmetric in the sense that it does not matter if the spoofing detection is done at the reference or UE receiver; it was done this way for convenience, though it is likely more scalable if processing is done at the UE receiver. It is important to note that this data should be transmitted in some secure manner to ensure the avoidance of man-in-the-middle type attacks where the data is intercepted by a third party and tampered with. In this implementation, a secure shell tunnel was used, which adds somewhat to the computational burden. As the sampling rate is only 5.7 MHz and the data are sampled with 2-bit quantization, this means the data link between the reference and user equipment receivers need only support rates of 11.4 megabits per second, which is well within the capabilities of standard internet connectivity. A block diagram of the system architecture is shown in Fig. 1.

In order to perform a realistic spoofing attack without broadcasting any signals over the air, the receiver-spoofer was connected via an RF combiner to the UE receiver as shown in Fig. 2. Although in a real attack the spoofing signals would be transmitted wirelessly to the UE receiver, the setup-up used presents a very similar signal from the point of view of the UE receiver.

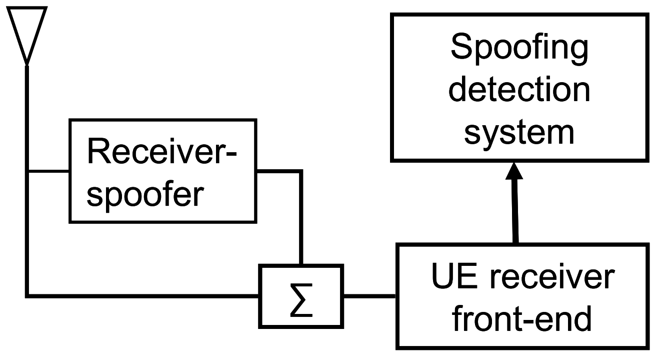

Fig. 2. Receiver/Spoofer Connection.

All processing was done on a personal computer with a quad-core Intel i7 930 CPU, and only standard hemispherical patch antennas were used at both the reference and user equipment receivers. The current implementation can process data with 10 signals common to both receivers at approximately 3 times faster than real-time, implying up to 30 common channels can be processed in real-time.

III. SOFTWARE OVERVIEW

In this section we will examine the general theory behind the spoofing detection method implemented here, how the spoofing detection statistic threshold was calculated, and general algorithms required due to the particular software receiver that was used.

A. Spoofing detection algorithm

In this implementation of the Lo spoofing detection method, temporal alignment of the two data streams is the first step taken. Rather than time-stamping the data streams using additional equipment, the embedded navigation message data is used. To do this, both data streams are tracked until the time of week (TOW) has been decoded in both. Using this information, the latency between the two streams can be determined. Whichever stream lags is then tracked while the other stream is buffered, until the receiver is processing the exact same C/A code period on both data streams. The estimated start time of the n^{th} C/A code period for the reference receiver is defined as $t_{ref}(n)$. Similarly, the estimated start time of the n^{th} C/A code period for the user equipment receiver is defined as $t_{ue}(n)$. Given that we know $t_{ref}(n)$ and $t_{ue}(n)$ from the normal, continuous tracking of the C/A signal, and given that any group delay between the C/A and P(Y) codes is determined by the transmitter and common to both the reference and UE receivers, the P(Y) code phase in the reference receiver data stream at $t_{ref}(n)$ should be the same as the P(Y) code phase in the UE receiver data stream at time $t_{ue}(n)$. The effects of different multipath errors on the C/A and P(Y) codes are assumed negligible due to the low-multipath environment of the antennas used here, although in general this may not be true.

Due to the low sampling rate (5.7 MHz) of the receivers used in this work as compared to the chipping rate of the P(Y) code (10.23 MHz) there is the question of sub-sample alignment as well as the coarse alignment described above. That is, to achieve a large cross-correlation value, we must temporally align the P(Y) codes to within a fraction of a chip. The P(Y) chip period of about 97 ns and the data sampling period of 175 ns means that if alignment is done only to the nearest sample it could be off by as much as ½ sample, or 0.9 chips, leading to significant correlation loss. However, as the sampling period is not a multiple of the P(Y) code chipping period, this error will vary over the course of each accumulation, sometimes being close to zero, and having a mean value 0.45 chips. The computational resources that would be required to interpolate the data on a sample-by-sample basis to the estimated start times of the P(Y) code chips was deemed prohibitively expensive, so instead we have elected to simply choose the sample nearest the estimated P(Y) code chip start time, updating it every millisecond based on the estimated C/A code start time.

The GPS C/A and P(Y) codes are both transmitted on the L1 frequency, with the C/A code being transmitted ninety degrees out of phase with respect to the P(Y) code. As the P(Y) code is encrypted and generally unavailable to civilians, it is necessary to track the C/A code in such a way that the phase is known.

A phase-locked loop (PLL) is used to accurately measure the phase of the desired C/A code signal. The PLL discriminator requires mixing of the signal with both an in-phase and a quadrature carrier replica. The PLL is formulated to steer the carrier such that the C/A code power lies entirely in the in-phase channel after carrier wipe-off. As the P(Y) code is in quadrature with the C/A code, all that is required to isolate the portion of the signal that should contain the P(Y) code is to save a replica of the data after mixing with the quadrature carrier replica.

Once the portion of the signal containing P(Y) code has been isolated in both the reference receiver and the user equipment receiver, it only remains to multiply the two data streams on a sample-by-sample basis and accumulate the result. If the receiver is not being spoofed, one is essentially computing the autocorrelation of the P(Y) code as modified by the receiver front-end and with the inclusion of noise.

B. Detection Threshold Calculation and Analysis

It is necessary to analyze the cross-correlation spoofing detection statistic in order to determine how much integration time would be required in order to achieve a reasonably small probability of false alarm and, at the same time, a reasonably large probability of detecting an actual spoofing attack. This analysis is particularly important given the unusual approach used here, one which relies on a heavily filtered version of the P(Y) code that retains only the central 2.5 MHz of its 20 MHz bandwidth.

A full treatment of the spoofing detection statistic derivation is not provided here; the reader is referred to Refs. 5 and 6 for thorough coverage of this topic. An abbreviated discussion is provided in the interest of highlighting some of the challenges addressed in making this system operate in real-time.

The un-normalized spoofing detection statistic is defined as:

$$\gamma_u = \sum_{i=0}^{M} y_{qai} y_{qbi} \tag{1}$$

where y_{qai} is the quadrature base-band-mixed signal from Receiver A that is sampled at time t_i, and y_{qbi} is the quadrature base-band-mixed signal from Receiver B sampled at the same time. The number of samples summed together to produce the spoofing detection statistic is M, and i indicates the index within the summation period. The quadrature base-band mixed signal from Receiver A can be modeled as:

$$y_{qai} = 0.5 A_{pa} P_{Yf}(t_i) + n_{qai} \tag{2}$$

where A_{pa} is the P(Y) code amplitude at receiver A, P_{Yf} is the P(Y) code after filtering by the RF front-end at time t_i, and n_{qai} is the quadrature base-band noise term, which is an element from a discrete-time Gaussian white noise sequence. Replacing the subscript a in equation 2 with b gives the model for the signal from Receiver B. The statistics of n_{qai} are given by:

$$E\{n_{qai}\} = 0\,, \quad E\{n_{qai}^2\} = \frac{1}{2}\sigma_{RFa}^2\,, \quad E\{n_{qai} n_{qaj}\} = 0 \text{ for all } i \neq j \tag{3}$$

Here, σ_{RFa}^2 indicates the effective variance of the noise in the raw RF samples from Receiver A. The subscript a here can similarly be replaced with b to describe the quadrature base-band noise term for Receiver B. The expected value of the spoofing test statistic under the hypothesis H_0 that there is no spoofing at Receiver B is:

$$\begin{aligned}\bar{\gamma}_u &= E\{\gamma_u \mid H_0\} \\ &= \sigma_{RFa}\sigma_{RFb} M \Delta t \sqrt{\left(C/N_0\right)_{pya}\left(C/N_0\right)_{pyb}}\end{aligned} \tag{4}$$

The C/N_0 terms here are the carrier-to-noise ratios of the P(Y) code at Receivers A and B. This quantity is computed from the carrier-to-noise ratios of the C/A code at each receiver, and takes into account various loss factors. Loss factors include the decrement in transmitted P(Y) code power as compared to C/A code power, the effect of the front-end filtering on both the C/A and the P(Y) codes, and other minor factors. Again, see Refs. 5 and 6 for the complete derivations. Using the methods described in Refs. 5 and 6, the loss factor computed for the UE receiver was 7.92 dB, and the loss factor computed for the reference receiver was 8.06 dB. The variance of the spoofing test statistic can be computed under two hypotheses: hypothesis H_0 that there is no spoofing at Receiver B, and hypothesis H_1 that Receiver B is being spoofed.

$$\sigma^2_{\gamma_u|H_0} = E\{\gamma_u^2 \mid H_0\} - E\{\bar{\gamma}_u^2\} \tag{5}$$

$$\sigma^2_{\gamma_u|H_1} = E\{\gamma_u^2 \mid H_1\} \tag{6}$$

The derivation of the variance of the spoofing test statistic under hypothesis H_1 makes the assumption that the P(Y) code power at Receiver B is zero due to code and carrier phase misalignment of the spoofed C/A code with the true P(Y) code. Obviously the P(Y) code is still present in the data, but due to this misalignment the P(Y) code cross-correlation power will be negligible. It will later prove useful to normalize the spoofing test statistic; a reasonable normalization to use is to divide by the standard deviation under the hypothesis H_1.

$$\gamma = \gamma_u \big/ \sigma_{\gamma_u|H_1} \tag{7}$$

Given the spoofing test statistic expected value and variance, a suitable normalized test statistic threshold is given as:

$$\gamma_{th} = (\sigma_{\gamma_u|H_0}\,\mathrm{norminv}(\alpha_{fa},0,1) + \bar{\gamma}_u) \big/ \sigma_{\gamma_u|H_1} \tag{8}$$

where *norminv* is the inverse normal cumulative distribution function in MATLAB. In order to avoid inclusion of this function in the real-time code, the *norminv* function is pre-calculated using a fixed probability of false alarm, α_{fa}, and the result stored as a constant. For all tests done here, a false alarm probability of 0.2% was used. The resultant value is then scaled by the standard deviation under hypothesis H_0, and added to the expected value of the un-spoofed test statistic. The threshold is then normalized by the standard deviation under hypothesis H_1. The probability of detecting a spoofing attack is given by:

$$P_d = \int_{-\infty}^{\gamma_{th}} P(\gamma \mid H_1)d\gamma = \frac{1}{2\pi}\int_{-\infty}^{\gamma_{th}} \exp(-0.5\gamma^2)d\gamma \tag{9}$$

It is important to note here that P_d depends on the value of the spoofing detection statistic threshold, which is itself a function of the signal carrier-to-noise ratio, the noise variance, and the integration time. The first two quantities vary with time due to changing environmental conditions (e.g., SV elevation), and the latter quantity is a parameter that the user may set. In the interest of having a fixed integration time for all signals and to avoid complications from taking into account fluctuations in signal level during an integration period, it was deemed expedient to choose a fixed integration time for all signals (2.0 seconds was chosen in the current implementation). As the probability of false alarm is constant, the probability of detection varies with carrier-to-noise ratio. For a C/A code carrier-to-noise ratio of 50 dB-Hz at both the reference and UE receivers, a false alarm probability of 0.2%, and a 2 second integration time, P_d is greater than 99.999%.

C. Implementation-specific issues

The code for this work was based on a previously existing software GPS receiver[7] written in the C and C++ programming languages. Bit-wise parallel algorithms as described in Ref. 8 were implemented as an optimization. In this bit-wise approach, the data are stored as 32-bit integers. The data are quantized to two bits, with the sign bits from one set of 32 samples stored in one integer, and the associated magnitude bits in another. The carrier replicas are similarly packed into integers with sign and magnitude being two separate words. In the course of tracking the C/A code, the receiver stores the quadrature carrier replica used for each data stream, as well as a copy of the data itself. Thus to execute a cross-correlation, we must multiply and accumulate four things: the carrier replicas from both the reference and UE receivers, and the associated data from the reference and UE receivers. To enable a look-up table implementation all of the above inputs were split into 4-bit chunks. The sign bits are all logically exclusive-or'ed together, leaving the data magnitude and carrier replica bits from each receiver. The 4 bits chunks of each of the above elements (data_{ref}, data_{ue}, $\text{carrier}_{\text{ref}}$, $\text{carrier}_{\text{ue}}$, sign) are combined into a 20 bit word and then used as an index into a pre-computed look-up table, where the value at that index is the result of multiplying and accumulating the two base-band mixed data streams. It was determined that the largest possible accumulation value for 4 samples could be stored in two bytes, so the resultant table size was 2^{20} *2 bytes = 2MB.

IV. RESULTS

Several different tests were conducted using this algorithm. Both tests shown here utilize data from a receiver located in Ithaca, New York (42.44 N, 76.48 W), and a receiver located in Austin, Texas (30.287N, 97.736W). For both spoofing tests the receiver in Ithaca, NY was the reference receiver and the receiver in Austin, TX was the receiver under attack and needing detection of spoofing. Figs. 3 and 4 illustrate the spoofing test statistic γ_u and its expected value $\bar{\gamma}_u$ during the first spoofing test. For this test, the signal was un-spoofed for the first 60 seconds, then the receiver was spoofed with the spoofer's best estimate of the true signal for another 60 seconds, then at 120 seconds into the test the spoofer started moving the spoofed C/A code away from its estimate of the truth. During this test only some of the visible signals were spoofed for the purpose of illustrating the receiver response to an unspoofed signal during a spoofing attack. In a real attack, the spoofer would both attempt to spoof all visible signals, and would likely not change the signal in the exact manner shown here.

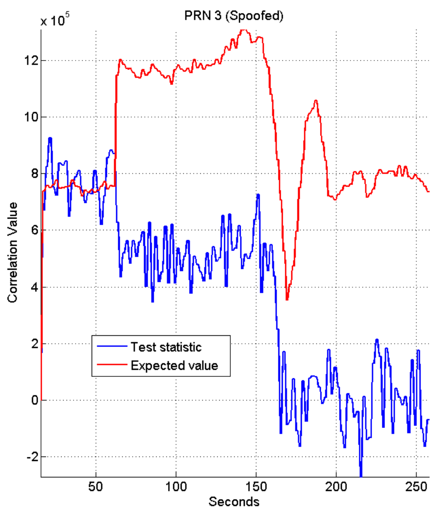

Fig. 3. Receiver response in the presence of spoofing.

To understand the results presented in Figs. 3 and 4, one must first understand the nature of the spoofing attack used. In this attack, the spoofer transmits a C/A code replica that has its code phase aligned (as much as it is able) with the true C/A code phase at the target receiver. The spoofer then increases its transmission power until it is slightly greater than the true signal power. Once the victim receiver is tracking the spoofed signal, the spoofer may then alter the C/A code phase as desired. During the un-spoofed portion of the first test, the expected value of the spoofing test statistic roughly matches the spoofing test statistic value. Once spoofing begins (i.e., at 60 seconds), the increased spoofer signal power causes the receiver's automatic gain control (AGC) to reduce the gain. This results in an increased expected value of the spoofing test statistic because the C/A code carrier-to-noise ratio has increased, which implies the P(Y) code carrier-to-noise ratio should increase. This is shown in Fig. 3 at 60 seconds. In truth, the P(Y) code is un-spoofed and the test statistic has a smaller but non-zero-mean value for several reasons: the gain has been decreased, the spoofer is spoofing with "truth" for this period so the spoofed C/A code phase may be still aligned with the P(Y) code phase, and the spoofed C/A signal carrier may not be exactly in quadrature with the true P(Y) signal carrier. Some time after 120 seconds (once the spoofer has moved the true signal more than one P(Y) code chip period), the spoofing test statistic becomes zero mean, as one would expect in the presence of spoofing. Fig. 4. illustrates the response of an un-spoofed signal while the receiver is being spoofed. The AGC response to the spoofing signal results in a lowered (true) C/A code carrier-to-noise ratio, and the expected value tracks the statistic closely the entire test. These plots indicate that the loss factors used in calculating the expected P(Y) code power as a function of the C/A code power are correct.

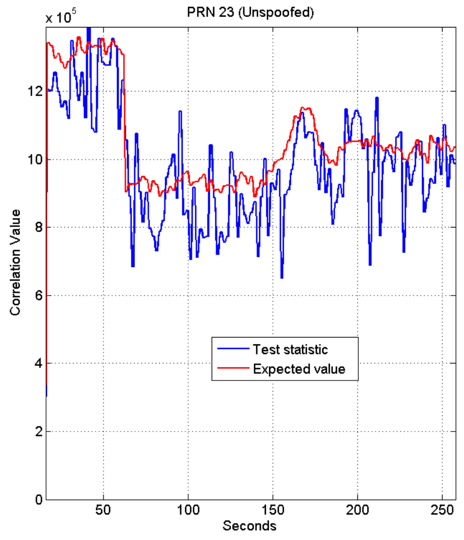

Fig. 4. Receiver response for an un-spoofed PRN in the presence of spoofing.

In the second test, there was no spoofing for the first 80 seconds, after which spoofing was begun. Selected results from this test are shown in Figs. 5 and 6. Fig. 5 shows the actual output of the receiver: the normalized test statistic minus the normalized spoofing detection threshold, $\gamma - \gamma_{th}$, normalized by the standard deviation under hypothesis H_1. If this quantity is below zero, spoofing has been detected, if it is above zero no spoofing has been detected.

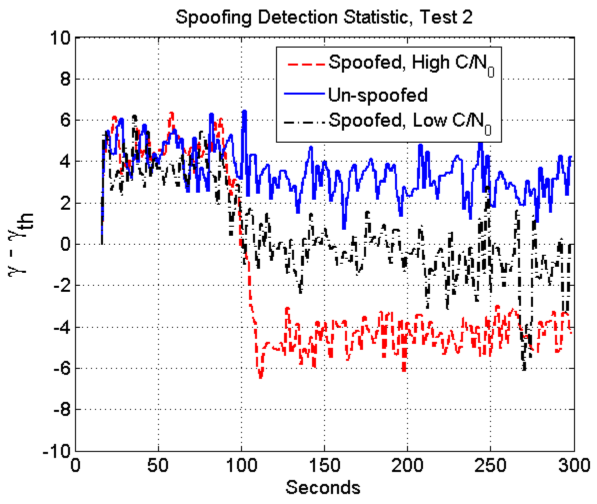

Fig. 5. Receiver spoofing detection statistic minus threshold during a spoofing attack.

The probability of detecting a spoofing attack for the same three signals shown in Fig. 5 is plotted in Fig. 6. For the red dashed lines in Figs. 5 and 6, the carrier-to-noise ratios of both the UE (spoofed) and Reference C/A code signals were high, 50 dB-Hz and 47 dB-Hz, respectively, and the spoofing detection statistic is below the threshold shortly after spoofing begins. The high carrier-to-noise ratios of these signals leads to a very high probability of detection, as shown in Fig. 6, except for a period of approximately 20 seconds right after spoofing begins, due to a drop in the carrier-to-noise ratio for that period when the UE tracking loops suffered some trauma from misalignment of the spoofed and true C/A codes. For the black dash-dotted lines in Figs. 5 and 6 the UE (spoofed) and Reference C/A code signals were somewhat lower at 45 dB-Hz and 42 dB-Hz, respectively. Recall that for the front-ends used in this test, the received P(Y) code power is approximately 8 dB lower than that of the C/A code. Given the fixed probability of false alarm for all signals, this leads to a lower probability of detection, but any detection can be considered reliable due to the low probability of false alarm.

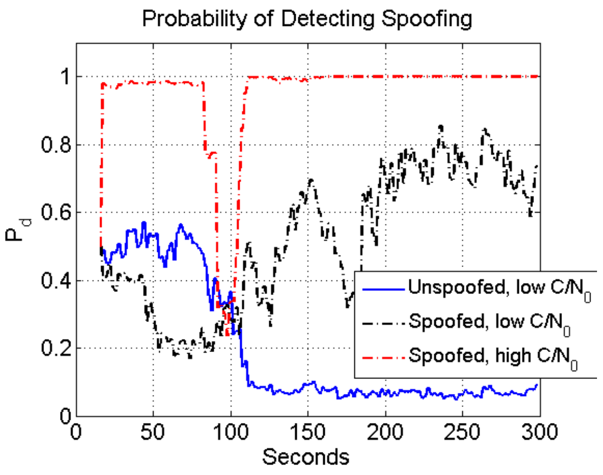

Fig. 6. Probability of detecting spoofing.

For the solid blue lines in Figs. 5 and 6, the UE (un-spoofed) and Reference C/A code signals were relatively low, at 38 dB-Hz and 42 dB-Hz, respectively. Although the probability of detecting spoofing for this signal is quite low, there is no false alarm. This low probability suggests that this spoofing detection method is not ideal for differentiating spoofed from un-spoofed signals in the presence of spoofing, even though it can successfully detect when spoofing is occurring.

V. DISCUSSION

A. Detection Probability

The low probability of detection for weak signals is of concern for this method. Due to the large amount of attenuation of the P(Y) code from the narrow RF filter bandwidth, it may be desirable to scale to integration time inversely with signal power. Although that was not done here, it is straightforward to implement, though it will add to the real-time computation burden.

B. Reference Station Spoofing

One might suspect that spoofing of the reference receiver would not be a problem for the proposed architecture. The reference receiver knows its location, and therefore, it might be able to use this knowledge in order to detect a spoofing attack. In fact, a spoofer could attack the reference receiver using the method of Ref. 2 in a way that does not try to spoof its position or its receiver clock time. Rather, this "auxiliary" spoofer would have another, more subtle goal in its spoofing attack. It would seek to spoof the reference receiver about what is the proper P(Y) code signal that is in phase quadrature with each received C/A code signal. Given such spoofing, the reference receiver would not detect any error in its position or even in its receiver clock. It would, however, transmit an erroneous base-band-mixed quadrature signal to the UE receivers that it was supposed to aid in detecting spoofing. If another spoofer, the main spoofer, then attacked the UE receivers using the same false P(Y) code in phase quadrature with each spoofed C/A code, then such an attack would defeat the present method and, indeed, the method of Ref. 4.

VI. CONCLUSIONS

In summary, a method for detecting spoofing of civil GPS signals in real-time has been implemented and tested. This method seeks to verify the absence of spoofing by looking for strong cross correlations between two receivers, one a reference receiver and the other the potential spoofing victim, of the portion of the P(Y) code that passes through each receiver's narrow-band RF front-end. This heavily filtered P(Y) code should be present in phase quadrature with the C/A codes of both receivers if neither is being spoofed. Lack of a strong cross-correlation should indicate a spoofing attack.

Several spoofing attacks were conducted, with the method successfully detecting spoofing (or lack thereof) in real-time. In the current implementation, the probability of missed detection is high for very weak signals, although all signals have a very small probability of false alarm.

REFERENCES

[1] “Vulnerability assessment of the transportation infrastructure relying on the Global Positioning System,” Tech. rep., John A. Volpe National Transportation Systems Center, 2001.

[2] Humphreys, T.E., Ledvina, B.M., Psiaki, M.L., O'Hanlon, B., and Kintner, P.M. Jr., "Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer," *Proceedings of the ION GNSS 2008*, Sept. 16-19, 2008, Savannah, GA, pp. 2314-2325.

[3] Warner, J.S. and Johnston, R.G., “A Simple Demonstration That the Global Positioning System (GPS) is Vulnerable to Spoofing," Journal of Security Administration, 2003.

[4] Lo, S., De Lorenzo, D., Enge, P., Akos, D., and Bradley, P., "Signal Authentication, A Secure Civil GNSS for Today," *Inside GNSS*, Vol. 4, No. 5, Sept./Oct. 2009, pp. 30-39.

[5] Psiaki, M.L., O’Hanlon, B.W., Bhatti, J.A., and Humphreys, T.E., “Civilian GPS Spoofing Detection Based on Dual-Receiver Correlation of Military Signals,” *Proceedings of the ION GNSS* 2011, Portland OR, pp 2619-2645.

[6] Psiaki, M., O’Hanlon, B., Bhatti, J., Shepard, D., Humphreys, T., “GPS Spoofing Detection via Dual-Receiver Correlation of Military Signals,” IEEE Transactions on Aerospace and Electronic Systems, in press.

[7] Humphreys, T.E., Psiaki, M.L., Kintner, P.M. Jr., Ledvina, B.M., “GNSS Receiver Implementation on a DSP: Status, Challenges, and Prospects,” Proc. 2006 ION GNSS Conf., Institute of Navigation, Fort Worth TX, pp. 237002382.

[8] Ledvina, B.M., Psiaki, M.L., Powell, S.P., and Kintner, P.M. Jr., “Bit-Wise Parallel Algorithms for Efficient Software Correlation Applied to a GPS Software Receiver," IEEE Transactions on Wireless Communications, Vol.3, No.5, September 2004.