

Crowdsourcing GNSS jamming detection and localization

Luka Strizic, *University of Colorado Boulder & Luleå University of Technology*
Dennis M. Akos, *University of Colorado Boulder & Stanford University*
Sherman Lo, *Stanford University*

BIOGRAPHIES

Luka Strizic is a freshly graduated Space Science and Technology MSc student from Luleå University of Technology, pursuing a Computer Engineering MSc at the University of Zagreb. He spent six months at the University of Colorado Boulder as a visiting researcher on the topic of GNSS radio-frequency interference detection.

Dennis M. Akos obtained his Ph.D. degree from Ohio University in 1997. He is an associate professor with the Aerospace Engineering Science Department at University of Colorado at Boulder with visiting appointments at Stanford University. His research interests include GNSS, software-defined radio, digital signal processing, and radio-frequency design.

Sherman Lo is a senior research engineer at the Stanford GPS Laboratory. He received his Ph.D. in Aeronautics and Astronautics from Stanford University in 2002. He has and continues to work on navigation robustness and safety, often supporting the FAA. He has conducted research on Loran, alternative navigation, SBAS, ARAIM, GNSS for railways and automobile. He also works on spoof and interference mitigation for navigation. He has published over 100 research papers and articles. He was awarded the ION Early Achievement Award.

ABSTRACT

GNSS has wide adoption in critical military and civilian infrastructure, requiring reliable operation. Therefore, its susceptibility to interference can result in anything from a minor inconvenience to a life-threatening affair. In this paper, we present a prototype implementation of the GNSS interference detection and localization, or, so-called, J911 system, which crowdsources measurements of GNSS observables from smartphones. Access to Carrier-to-Noise-density ratio (C/N_0) and Automatic Gain Control (AGC) level enables the system to distinguish natural causes of signal degradation from intentional jamming. A mobile application was developed for the Android OS, which records current location, per-GNSS-satellite C/N_0 and other available GNSS observables. The collected data is sent to a central server, where it is subject to interference detection assessment and visualization. With a high enough density of smartphones, localization methods can be employed, such as time difference of arrival (TDOA) or power difference of arrival (PDOA). With the help and oversight of the Department of Homeland Security, an exercise to test interference mitigation technologies was conducted in 2017, called JamX 17, where we fielded fifteen smartphones with GPS- and GLONASS-capable hardware. At the time of testing, the Android OS did not support AGC level reporting, so four SiGe GNSS Sampler Software-Defined Radios (SDR) were used instead. Analysis of the collected data shows that rough tracking of the jammer is possible, based on measurements from many phones. When the jammer gets closer to a smartphone, both GPS and GLONASS C/N_0 decrease in a similar pattern. When the jammer moves away, C/N_0 is restored to nominal levels, but decreases in the next phone that the jammer is closing in on, thus allowing phone-density-limited tracking. The C/N_0 measurements are validated by comparison to the AGC measurements from nearby SDRs, confirming that jamming, rather than natural obstruction, took place. In the test site area where the phone density was highest, a TDOA method was conducted and provided a good estimate of the jammer location.

INTRODUCTION

Services derived from global navigation satellite systems (GNSS) are widely used in many critical military and civil applications. Hence, the susceptibility of GNSS to interference can potentially result in inconvenient, costly and even life-threatening conditions. This is a problem as GNSS jammers, while illegal, can still be easily and cheaply procured. While there are regulations that seek to prevent GNSS interference, they are difficult to enforce. Enforcement requires detection and localization. However, detection and localization currently requires expensive signal processing equipment and manpower [1][2]. Resources dedicated to GNSS interference detection are few and far between. Non-dedicated resources may not distinguish interference from natural or physical outages. As a result, many jamming instances are either undetected or

misidentified. Hence, developing flexible and cost-effective methods to identify and locate the source of interference is necessary.

Smartphones can be extremely valuable sensors for detecting and localizing GNSS interference. They are deployed in the billions and are in high density in our most populated areas, where GNSS interference has the potential to do the most damage. Crowd-sourced information from even a small fraction of these could prove very valuable, as they can detect local interference that cannot be easily detected by reference station networks. Additionally, the Android operating system (OS) will provide access to many GNSS receiver observables that will allow for robust detection of jamming, allowing us to distinguish these events from other natural causes of signal degradation.

For the Department of Homeland Security (DHS) JamX 2017 exercise, we developed an Android application (app) to demonstrate a basic capability of smartphone based GNSS interference detection. The app gathers basic GNSS observables such as satellite information and carrier-to-noise-density ratio (C/N0) and pushes this data to a central server for interference detection assessment. We fielded 15 smartphones throughout the test site for this demonstration.

This report discusses the technology involved, the JamX test setup and the results from our testing of smartphone-based interference detection.

BACKGROUND

Smartphone-based crowd-sourcing of GNSS observations has been suggested for interference detection due to its many advantages [3][4]. The existing infrastructure provides built-in communications. An app based system allows for ease of deployment, updating and feedback [3]. The high adoption rate of smartphones allows for low cost but high density deployment. A density of as little as 100 smartphones per squared kilometer can provide useful detection and localization results, which is easily achieved in urban areas [3]. Lower densities can provide detection, thus providing a deterrent to an attacker though they may not be suitable for providing precise localization due to inaccuracies in measurements, unknown variables and other errors [3][4]. Smartphones have been successfully used to detect GPS jamming in specific, controlled scenarios, such as interference by commercial drivers to circumvent vehicle tracking systems [5] or via an Android application on a single phone with C/N0 readings and dead reckoning [6].

Our work expands upon previous research with an actual implementation of a GNSS jamming detection and localization system around low cost, consumer hardware with a bigger scale, distributed design and wider deployment in mind.

INTERFERENCE DETECTION

GNSS interference detection can be accomplished using observables, such as C/N0 and automatic gain control (AGC) levels, available to most GNSS receivers. C/N0 is a standard metric used to quantify the power of a tracked GNSS satellite signal relative to noise and is similar to signal to noise ratio (SNR). This is calculated for each individually tracked signal. AGC is a standard piece of receiver equipment used to set a level of gain such that power coming to the analog to digital converter (ADC) is relatively constant. Thus, AGC level essentially indicates the amount of energy entering the antenna. C/N0 and AGC can provide indications of anomalous energy that may be the result of interference. More powerfully, these two complementary measures can be used together to differentiate degradation due to interference from spoofing and natural causes. These are convenient measures for a crowd-sourced detection system. Android OS provides access to C/N0 measurements. Android 8 or "Oreo", introduced in August 2017, also provides the ability to access AGC outputs. Future smartphones should provide AGC measurements provided the smartphone original equipment manufacturer (OEM) and the GNSS chipset support it.

C/N0 expresses the power of a GNSS signal compared to the background and thermal noise. When a jammer approaches a GNSS receiver, such as a smartphone, the interfering noise level increases. This increases noise on all satellite signals and hence decreases C/N0 on all satellites. As the jammer gets closer, the received interference may become powerful enough to cause the GNSS receiver to lose track of the satellites. However, reduction of C/N0 and loss of tracking can occur due to natural causes such as going under foliage or going indoors. Hence, C/N0 measurements from individual receivers are generally not adequate for robust jamming detection. Additional information from AGC, C/N0 time history, different frequency bands and other receivers can be used to make C/N0-based jamming detection more robust.

Using C/N0 for rapid interference detection typically requires a comparison to expectations. For the comparison, we must know the nominal C/N0 for the receiver. This is typically around 30-40 dB-Hz. We also should know which satellites should be reachable from the location of the receiver. This allows us to determine if we should be getting C/N0 value for specific satellites, indicating if something has caused our receiver not to track an available satellite.

An AGC is used in receivers to amplify the incoming signal, after down-conversion and filtering to a specific level. After amplification, it is sampled, then digitized/quantized and passed on to the digital part of the receiver via the ADC. The specific amplification level needed is determined by the receiver design and, more specifically, its digital section. To be able to accommodate input signals of varying power, AGC mechanism is responsible for altering the gain level of the variable gain amplifier (VGA) by varying the voltage input to one of its pins, as illustrated in Figure 1. The required AGC voltage is controlled by the number of samples in each of the quantization bins, as illustrated by the histogram in Figure 2. For example,

if the incoming signal is quantized with two bits into four possible values, a goal could be to have 32% of the samples in maximum and minimum bins, which correspond to 3 and -3 in the figure, respectively. If the percentage of samples in the maximum and minimum bins starts increasing, the AGC voltage is lowered to reduce gain, thus increasing the number of samples which fall into bins closer to 0. Inversely, if the number of samples in maximum and minimum bins starts decreasing beyond 32%, the AGC voltage is increased.

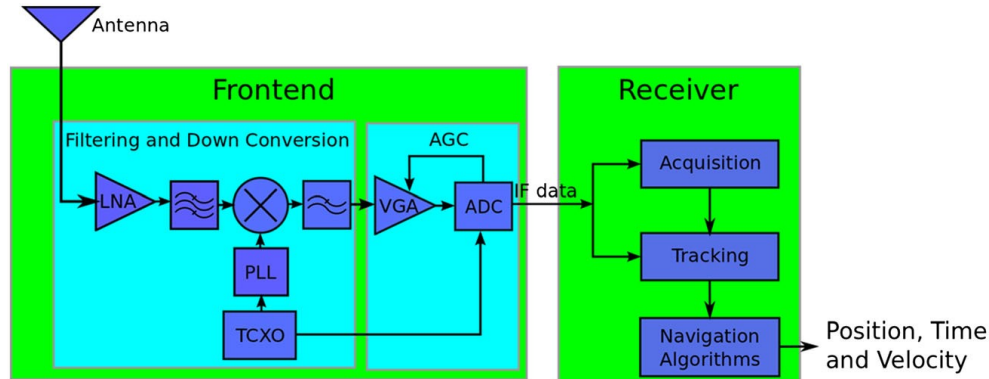


Figure 1. Block diagram of a GNSS receiver with Low Noise Amplifier (LNA), Phase Lock Loop (PLL), Temperature Compensated Crystal Oscillator (TCXO) and AGC. The AGC utilizes a Variable Gain Amplifier (VGA)

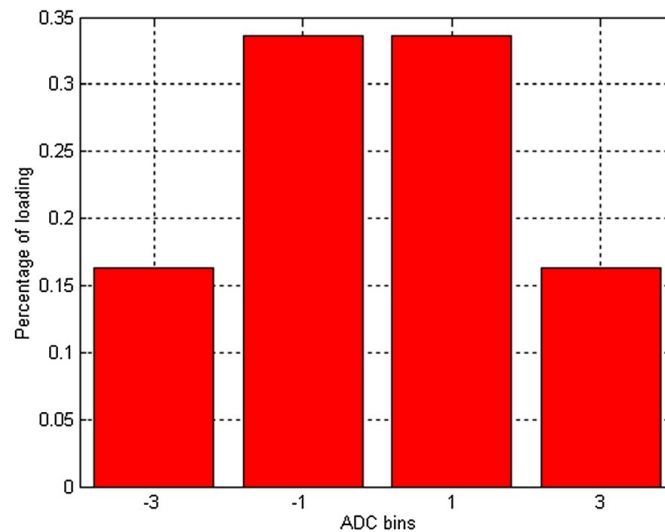


Figure 2. Bins for a 2 bit ADC

When a jammer is turned on, it emits a relatively powerful and noisy signal into the GNSS spectrum. This high power signal causes the AGC to reduce its amplification of the input signal, including the useful signal from the GNSS satellite, by dropping the AGC voltage. In that case, the digitized samples correspond more closely to the jammer's noise and do not contain enough information to resolve and extract the weak signal from the GNSS satellites -- the receiver is jammed.

To use AGC for interference detection, we need to know the nominal AGC voltage. Each device has its own nominal AGC voltage, as different receivers use different discrete components and different antennas. This is important to ascertain as many factors (i.e. imperfections in manufacturing processes, differences in operating temperature, etc.) can result in different nominal AGC levels even in the same model. The nominal AGC voltage/level can be determined by collecting samples over time and, for example, averaging them in ideal conditions. Statistical deviations of AGC from its nominal average or standard deviation provides useful information. A voltage increase means that satellites are obstructed, thus less power is received, and a voltage decrease indicates interference or jamming that deposits more power into the spectrum. A scenario is illustrated in Figure 3, where around hour 59, the AGC level decreases significantly below the standard deviation, suggesting interference in the GNSS spectrum.

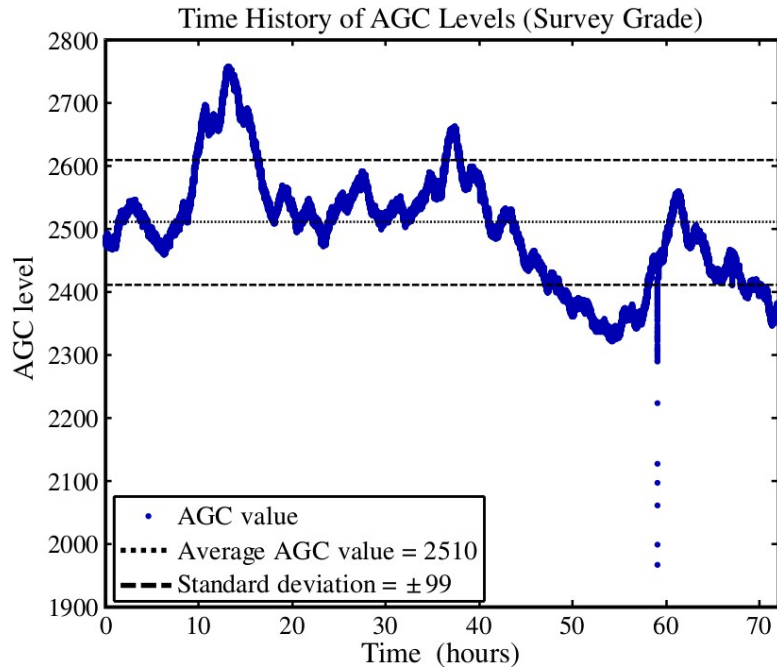


Figure 3. AGC samples with the average

While C/N0 and AGC can be used individually for interference and potentially spoof detection, when used together, they can distinguish different causes of GNSS degradation [7][8]. They can differentiate cases of environmental changes (i.e. going indoors) from man-made interference. Figure 4 shows AGC and C/N0 values from different scenarios. It shows the different relationship between C/N0 and AGC for spoofing, radio-frequency interference (RFI) and normal/nominal conditions. Generally speaking, jamming/RFI increases incoming energy (i.e. lowers AGC levels) while decreasing C/N0 whereas spoofing increases incoming energy while having similar or higher C/N0 levels. In summary, the combined use of C/N0 and AGC increase jamming detection robustness by reducing false positives.

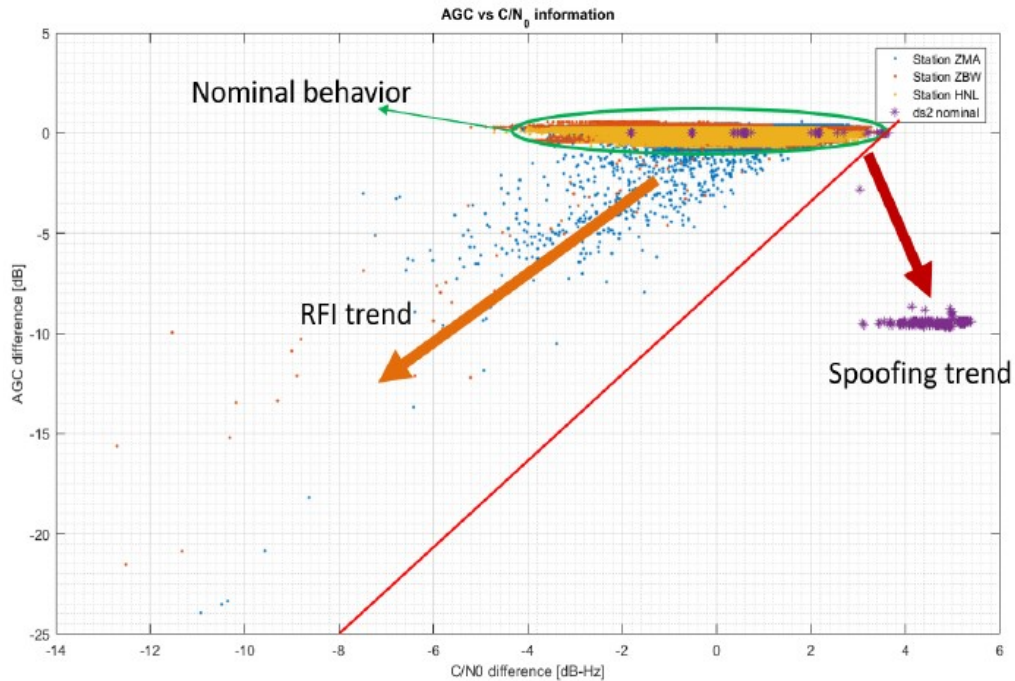


Figure 4. Effects of RFI, Spoofing and Nominal Conditions on AGC & C/N0[7]

LOCALIZATION

A powerful capability of a crowd-sourced GNSS measurements is the ability to localize jamming. C/N0 and AGC are coarse measures that are not meant to provide ranging, bearing or positioning information. Hence, single measurements cannot indicate jammer location. However, measurements from multiple smartphones can allow for geolocation of a jamming source due to the geometric diversity of these measurements. This capability requires that the smartphone measurements be shared or sent to a central server for processing. Time difference of arrival (TDOA) processing and power difference of arrival (PDOA) are both possible ways to provide refined localization. Measurement errors due to various factors means that three independent difference measurements, derived from four devices, will not generate an exact solution. Hence, least squares or similar algorithms need to be used#.

TDOA measures the difference in times of the signal's arrival to two or more receivers with known locations. All possible locations of the jamming emitter form a hyperboloid, to justify the time difference, and the intersection of multiple such hyperboloids gives, ideally, a single possible location.

PDOA measures the difference in the received power at two or more receivers with known locations. Once a propagation model is taken into account, the power difference correlates to a ratio of distances from the emitter to each station. A sphere of possible locations fits the distance ratio and the intersections of multiple spheres, ideally, give a unique emitter location.

EXPERIMENTAL SYSTEM

The tested system has both software and hardware components as well as server and field equipment. In addition, a SiGe software defined radio (SDR) was used to collect AGC data. The SiGe is a simple, low cost SDR and we had four units which were generally located near a smartphone. These are used as Android OS does not yet support AGC measurements and hence these can act as a proxy for the future when these measurements are available. This section focuses on the smartphone app and server software developed.

The system developed was tested at the 2017 DHS JamX exercise. This exercise supported tests of different technologies to address interference issues such as GNSS interference. It was held at a remote location in eastern Idaho over three nights in July 2017. We tested on two of those nights – we termed Night 1 (July 19-20) and Night 2 (July 20-21). GNSS jamming was tested on this exercise using various commercial jammers bought via the Internet. These jammer had varying specified levels of radiated power from milliwatt (mW) to watt (W) levels. They also potentially had different jamming waveforms. Each jammer was operated statically in our test area before being driven in a roughly 2 mile loop by the test area at low speeds.

Smartphone application

We developed a bespoke app for capturing, storing and sending GNSS measurement information. It is designed to capture GNSS information if available from the OS with position, accuracy, satellite, C/No, pseudo range, and AGC values all supported. This app supports all constellations (GPS, GLONASS, Galileo and Beidou). It is built without using new features introduced in newer versions of Android while taking advantage of the new data made available by these newer versions. Hence it can operate on earlier version of Android, while it has an eye towards the future and newer Android data capabilities. These observables are captured at a 1 Hz rate and can be uploaded to a server should connectivity be available.

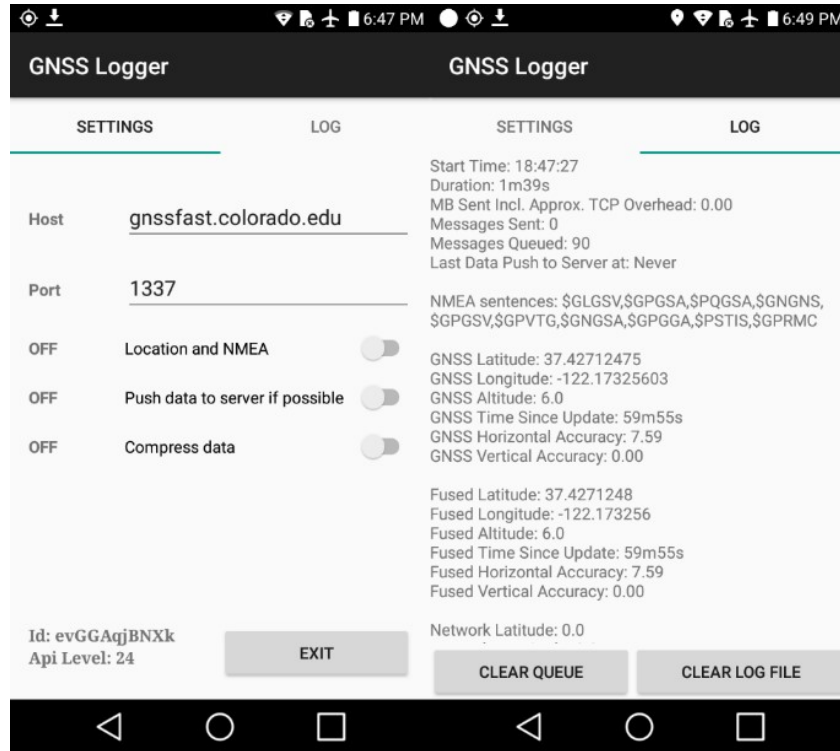


Figure 5. GNSS Interference Detection App. Data Collection Setting (Left) and Data Log (Right)

The application was developed in the programming language Java, which is well supported in the Android ecosystem, through the software development kit (SDK) and the integrated development environment (IDE). It is based on the open-source GNSSLogger application developed by Mohammed Khider of Google to demonstrate GNSS capabilities of Android OS [9]. The settings screen is shown on the left side of Figure 5, with options explained later in this section. The logging screen, shown on the right of Figure 5, includes all the information that is sent to a remote server, for user's preview.

Raw GNSS measurements, NMEA sentences and location

The application can gather the following location- and GNSS-related information offered through the Android application programming interface (API):

- location updates sourced by GNSS,
- location updates sourced by the cellular network,
- location updates sourced by the fused provider,
- standardized National Marine Electronics Association (NMEA) sentences.

Location updates provided by the cellular network are based on the known cellular tower locations and are fairly imprecise, when compared to GNSS-sourced location. The fused provider is available through Google Play Service. While not built into Android, Google Play Services were available on all the tested phones, therefore it was used too. It takes into account GNSS, cellular network, visible WiFi and Bluetooth stations when calculating current location. Ephemeris and information about acquired satellites is sent to the application using NMEA sentences, which are parsed to extract the desired information. An example of NMEA sentences:

```
$GPGSV,3,1,11,22,77,258,42,31,60,059,42,03,55,305,41,14,38,072,36*72
$GPRSV,3,2,11,01,33,235,38,26,28,136,30,23,26,278,38,32,19,082,32*7E
$GPGSV,3,3,11,25,12,036,26,11,08,225,29,16,09,159,*49
$GPGGA,080741.00,4332.360641,N,11249.798235,W,1,17,0.4,1527.6,M,-17.4,M,*,*63
```

Later Android versions (7.0 and up) offer an interface to programmatically extract the same information that is contained in the NMEA sentences, however it was not used in the application to avoid diverging in source code between older and newer Android versions. Version 8.0 introduced API to get AGC level, which will be useful in future revisions of the application.

Internet connectivity

Messages are created periodically on the phone and sent to the server whenever a connection can be established. The user can select the remote destination -- host address and port -- where the messages get sent to. A message contains all the information that is received from the OS and the times of last updates. An example is shown:

```
{
  "other_nmea": "$GPGGA,080741.00,4332.360641,N,11249.798235,W,1,17,0.4,1527.6,M,-17.4,M,, *63",
  "GLGSVs": [
    "$GLGSV,2,1,07,69,70,061,35,79,66,069,37,80,37,187,26,70,35,318,33*60",
    "$GLGSV,2,2,07,68,25,111,32,86,11,330,32,85,11,279,30*56"
  ],
  "GPGSVs": [
    "$GPGSV,3,1,11,22,77,258,42,31,60,059,42,03,55,305,41,14,38,072,36*72",
    "$GPGSV,3,2,11,01,33,235,38,26,28,136,30,23,26,278,38,32,19,082,32*7E",
    "$GPGSV,3,3,11,25,12,036,26,11,08,225,29,16,09,159,*49"
  ],
  "id": "cjACrijAIQA",
  "timestamp": 1500624462232,
  "lat_gnss": 43.53934467869471,
  "lon_gnss": -112.82997114268005,
  "alt_gnss": 1510.1347219543397,
  "hacc_gnss": 4,
  "vacc_gnss": 0,
  "timestamp_gnss": 1500624459999,
  "lat_network": 0,
  "lon_network": 0,
  "alt_network": 0,
  "hacc_network": 0,
  "vacc_network": 0,
  "timestamp_network": 0,
  "lat_fused": 43.5393447,
  "lon_fused": -112.8299711,
  "alt_fused": 1510.1347219543397,
  "vacc_fused": 0,
  "hacc_fused": 4,
  "timestamp_fused": 1500624459999,
  "agc_bei": 0,
  "agc_gal": 0,
  "agc_glo": 0,
  "agc_gps": 0
}
```

The message is constructed in JavaScript Object Notation (JSON) format and is ASCII encoded. Because of its encoding, there is a lot of redundancy. The message, as given in the example, is 929 bytes long. If, instead, binary encoding was used, with just sending the extracted data, where every numerical value is presented with 8 bytes as a double-precision floating-point number and timestamps as 8 byte integers, the message would be 647 bytes long, which is a significant saving even with the approximate transmission control protocol (TCP) overhead of 40 bytes with each message. Furthermore, presenting numerical values with 4 bytes as single-precision floating-point numbers and timestamps as 4 byte integers can still give enough precision but would further reduce the message size.

Message size and frequency may be important considerations to reduce the load on cellular towers in densely populated areas, if E911 infrastructure were to be used for jamming detection (so-called J911) [3].

Storage

To prevent data loss in unexpected scenarios, where the developed application crashes, every message that is created is stored into a file on the phone in the JSON format. If recovered from the phone, the file can be "replayed" to the remote server. When Internet connectivity is unavailable, the messages are stored in working memory of the application. To prevent data loss

between restarts of the application, all unsent messages are saved from the working memory into a SQLite database that is a standard part of the Android OS and independent of other applications.

Server software

A publicly reachable machine was used to host the backend, which consisted of a web server, implemented in Python, and a PostgreSQL database. When a message on the phone is ready, it is sent through the opened TCP connection to the server. After receiving a message, the server interprets it as a JSON object and extracts all data from it. If no problems were encountered in the message formatting, the data is stored into the PostgreSQL database, laid out in Figure 6.

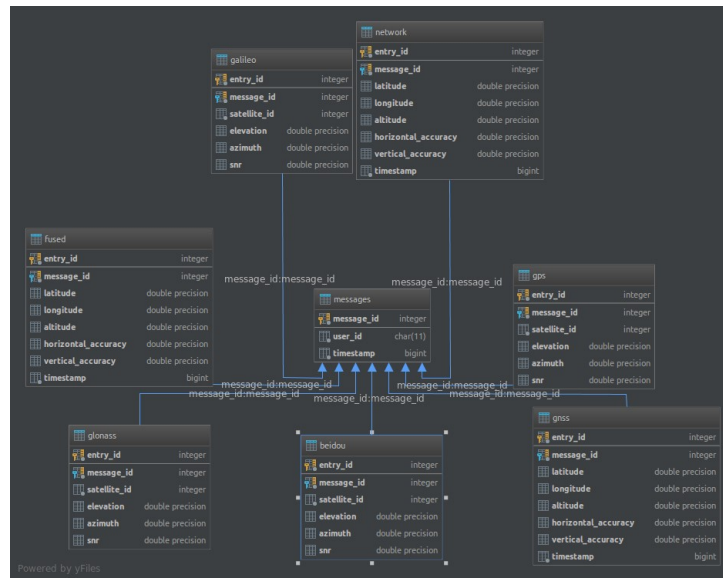


Figure 6. Layout of the PostgreSQL database.

Processing algorithm and visualization software

With the data in the central server, processing was developed to identify jamming. For the initial development, a simple algorithm was created for detection and identification. Each receiver reported, in a message to the server, the C/N0 of all satellites that it sees and each message was treated individually without consideration of other receivers or prior messages. For our initial analysis, the average of the highest four C/N0 reported in a message for each constellation was used as an interference metric. If less than four satellites' C/N0 was measured at a time, the missing ones were assumed to be zero. The single number for each constellation for an instant in time is compared to a fixed threshold to determine if that given receiver indicates jamming at that given time. A more sophisticated version may have examined the time history of the receiver to set thresholds. Additionally, satellites may be weighted based on their likelihood to be attenuated by jamming or other sources.

Frontend software was developed to visualize all the measurements taken. Since all captured information gets stored into the database, the frontend is independent of other parts of the stack, as long as the database layout does not change. MATLAB was used to query the database for specific time periods and specific phones and then used to plot locations of phones and the measured C/N0, overlaid on a map of the area, over the duration of the test. Screenshots of the plots that the frontend makes are given in section "Results".

J911

In the United States of America, E911 is an upgrade to the 911 system that enables the automatic reporting of telephone number and location of every 911 caller, wired or wireless, to public safety entities. Its European equivalent is called E112. Such a system allows for prompt reaction even when communicating a location is difficult or impossible.

Two phases are involved in obtaining the caller's location. Phase I sends the cell tower's location to the PSAP, which is easier to obtain, while Phase II sends the cellphone's location. Enhanced 911 infrastructure is an excellent candidate to be reused for J911, a system for real-time GNSS jamming monitoring on the national level, since the changes needed for J911 implementation would need to be made only in software, across the whole stack [3]. All essential hardware is already in place, except the smartphones with GNSS chips supporting required features. However, with the current relatively short smartphone life-cycle, it is a matter of a few years when a significant portion of the Android market will have a minimum or higher Android OS version and necessary hardware needed to run the developed application with both AGC and CN0 measurements, if historical trends continue [12].

EXPERIMENT SET UP

The baseline test equipment were smartphones with our GNSS measurement application installed. The observables that were captured by the application were captured at a 1 Hz software-configurable rate. Several different Android smartphones were used for the tests with Android OS versions ranging from 5.1 to 7.0:

- 9 x Alcatel Ideal (ALC)
- 1 x Galaxy Note 3 (GN3)
- 1 x Galaxy S5 (GS5)
- 2 x LG Aristo (LGA)
- 1 x LG Tribute HD (LGH)
- 1 x HTC Desire 530 (HTC)

All the smartphones have a multi-constellation capable GNSS chip with GPS and GLONASS measurements available, despite documentation indicating that all four GNSS are supported. The Samsung phones utilize a Broadcom GNSS chip while the others use a Qualcomm GNSS solution. The Alcatel is the lowest cost phone and can be found for \$20 while the Samsung phones are several year old flagships. The smartphones were distributed across approximately one kilometer squared, as depicted in Figure 9, for the first and second night of testing. Most were set on the ground though some were in elevated positions on a post or on a hill.

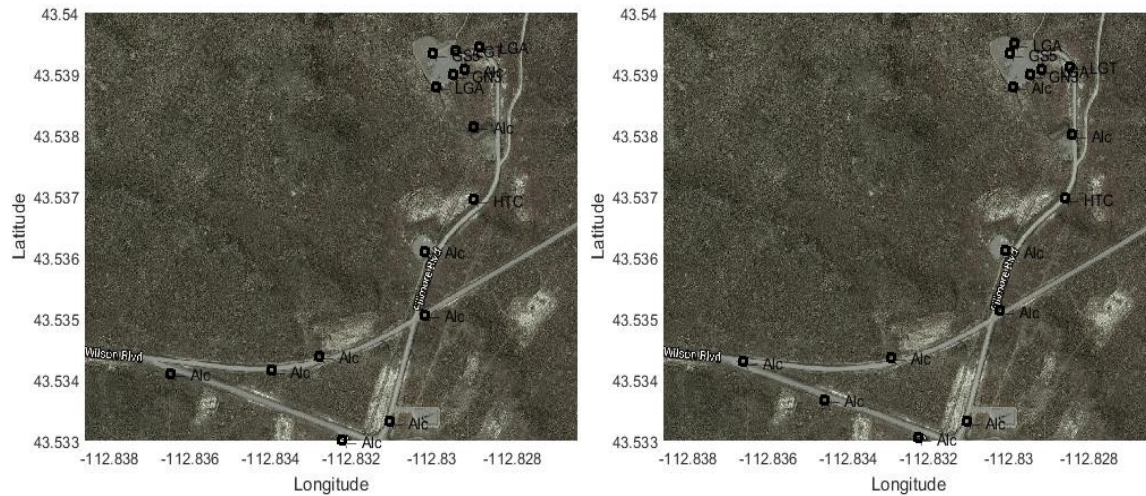


Figure 9.

Smartphone Locations Night 1 (Left, July 19) and Night 2 (Right, July 20): ALC = Alcatel Ideal, LGA = LG Aristo, LGT = LG Tribute HD, GS5 = Samsung Galaxy S5, GN3 = Samsung Galaxy Note 3

A satellite image of the test site and a topographical map are shown on the left and right side of Figure 12, respectively. From the topographical map, a rough idea of the terrain can be had and the presence of changes in elevation, significant for electromagnetic wave propagation, can be seen.

AGC measurements were also taken at four sites that are nearly collocated with some of the smartphones. The set up for Night 1 and 2 are shown in Figure 10 and Figure 11, respectively.

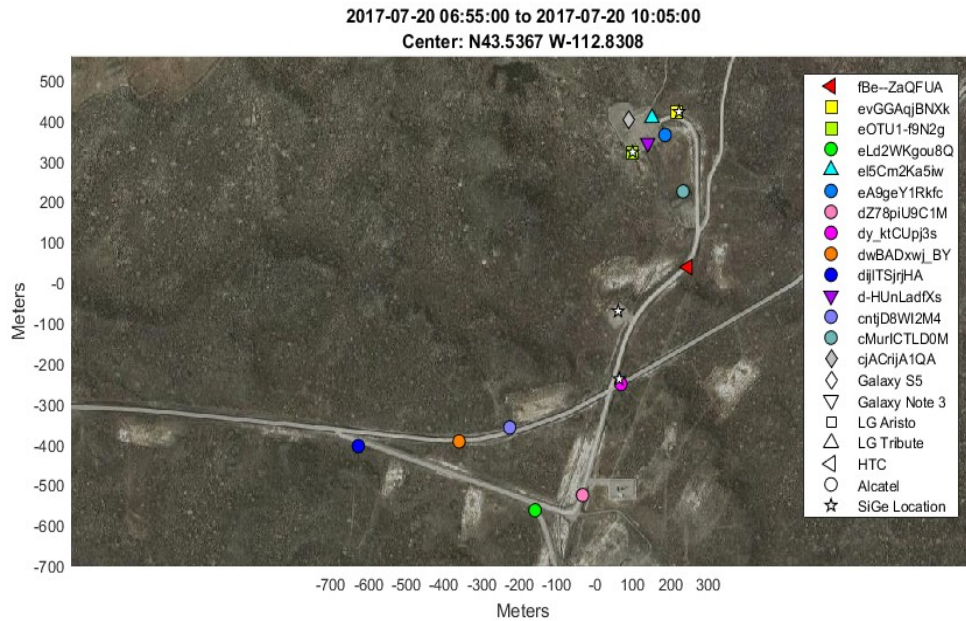


Figure 10. SiGe (with Smartphone) Locations Night 1 (July 19-20)

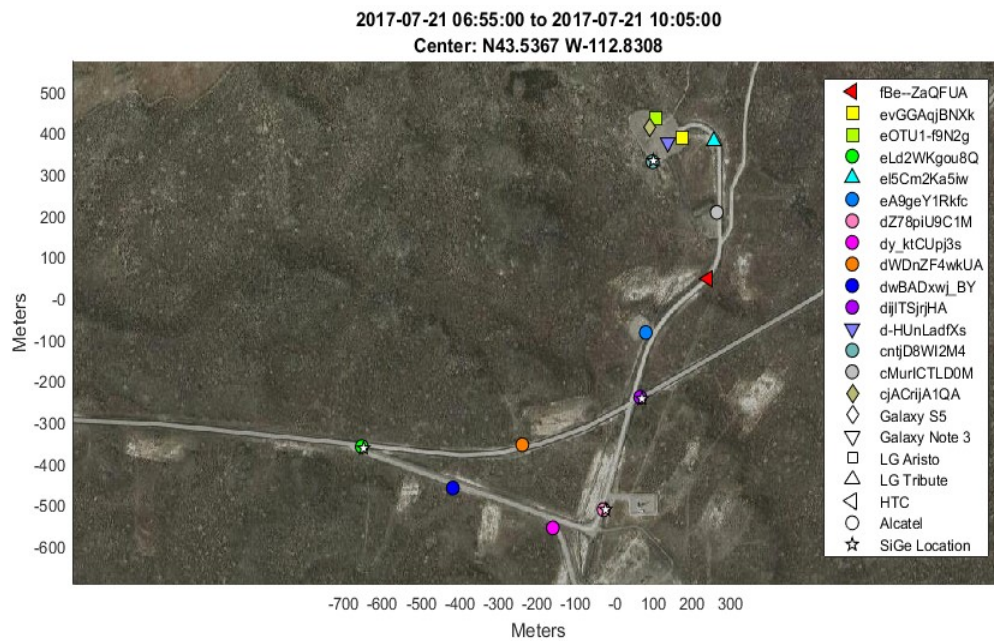


Figure 11. SiGe (with Smartphone) Locations Night 2 (July 20-21)

With the smartphones distributed and set up, various jammers, turned on one at a time, were driven in a car around the test site. Data was collected and plotted over time. Due to space constraints of this paper, one interesting sequence of measurements is shown from the first night of the test in the next section, section “Results”.



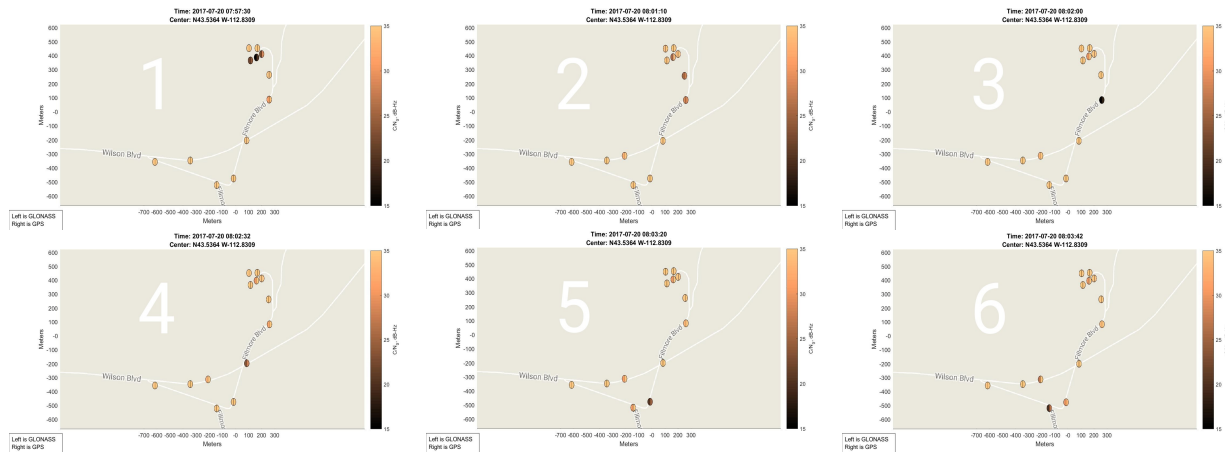
Figure 12. Satellite image (Left, from Google Maps) and topographical map (Right, ArcGIS) of the test site

RESULTS

Carrier-to-Noise-density smartphone detection

Analysis of the videos and plots made by the frontend showed that tracking the jammer is possible. It should be noted, however, that the measurements are fairly noisy. Additionally, power saving measures in some version of the Android OS would not let the application run for longer than an hour without any additional activity. It is hard to insure a message construction frequency of 1 Hz precisely, thus some one-second intervals have more than one message in them, while other one-second intervals do not have any messages in them, which is perceived as flickering in the video. Nonetheless, at least twelve phones were recording reliably throughout the test. Indeed, the benefit of this system is that it should work with many smartphones not operating or providing data.

Figure 13 shows progression of the jammer, as observed by the smartphones. There was no accurate truth reference available. Nevertheless, the way which the jammer moved, taking into account that it was in a car on the road, can be clearly seen. The dark circles indicate low average C/N_0 suggesting proximity to the jammer. The plots show the movement of the jammer in time which shown in coordinated universal time (UTC). The order of the plots is also given in numbers on the top left of the figure and from this we can clearly see the travel of the jammer over time. This travel order is also shown in Figure 15.



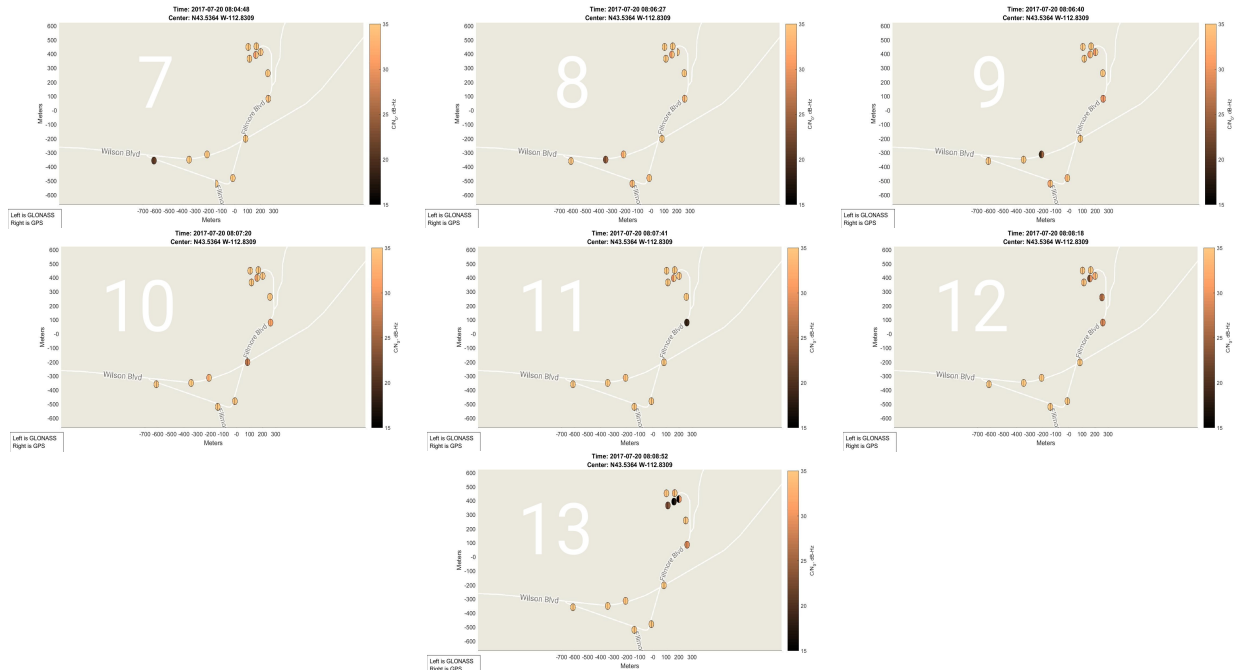


Figure 13. Progression of Jamming Indication over time for one Jammer drive.

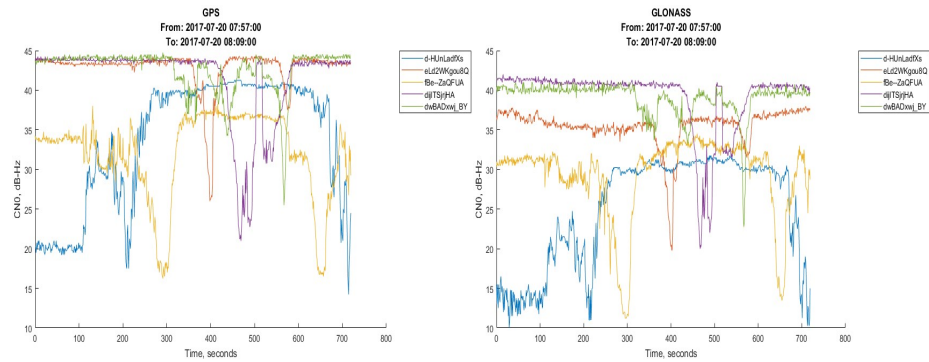


Figure 14. GPS (Left) and GLONASS (Right) C/N0 progress over time for five phones

Alternatively, the progress of the jammer can be seen in Figure 14 which depicts the average C/N0 over time for the top four tracked GPS or GLONASS satellites for several receivers. The location of the receivers are shown in Figure 15 and not all phones are shown in Figure 14, to maintain readability. As the jammer moves, we can observe different phones having significant dips in C/N0 level when the jammer gets close to them. Both GPS and GLONASS exhibit very similar patterns.

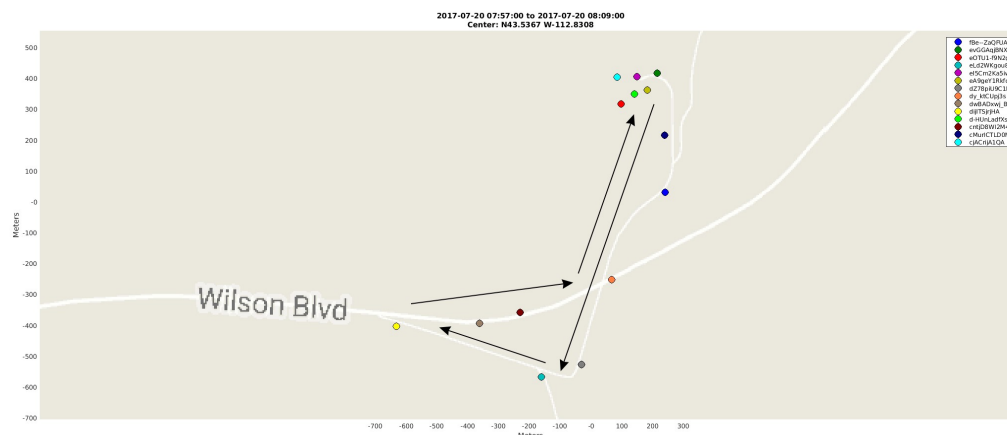


Figure 15 shows positions of the various phone identities as well as the motion direction of the jammer. At the start of the plot, the jammer is at the northernmost group of smartphones, where d-HUNLadFXs is a representative one from the group. As the jammer moves southward on the road, C/N0 of d-HUNLadFXs improves, but fBe-ZaQFUA worsens, located at the middle, between the two junctions on that stretch of the road. Some "spilling" can be noticed, from the northernmost group all the way to the middle of the road to d-HUNLadFXs, which is likely to be caused by the geography of the terrain. Next smartphone to have a noticeable dip is the southernmost, eLd2WKgou8Q, quickly followed by dijITSjrjHA, the westernmost smartphone. The next one that sees a dip is dwBADxqj_BY, located in the middle of the hypotenuse of the "triangle" of roads. Afterwards, the reverse of the first two phones is seen, again with some "spilling". It is easy to recognize from the plots which smartphones is close to a jammer, given their history.

AGC measurements

AGC measurements taken at four sites provides us some sense of its utility in aiding or corroborating C/N0 results. Figure 16 shows the AGC levels of the four SiGe over time for Night 1. The difference performance level of each of the different jammers can be seen. Figure 17 shows a zoomed in comparison of AGC and C/N0. These measure corroborate each other and show how these measures can be used to verify that jamming, rather than obstruction, caused the GNSS degradation. Because it was taken with a SDR and time tagged with GNSS, more sophisticated localization can be conducted on these measurements. Localization using time difference of arrival (TDOA) can be conducted. The TDOA derived position hyperbolas are shown in Figure 18 for the case when the jammer is in test site 2A, the open area at the end of the road where we have a high density of phones. The TDOA localization, where the hyperbolas intersect, provides a good estimate of the jammer location.

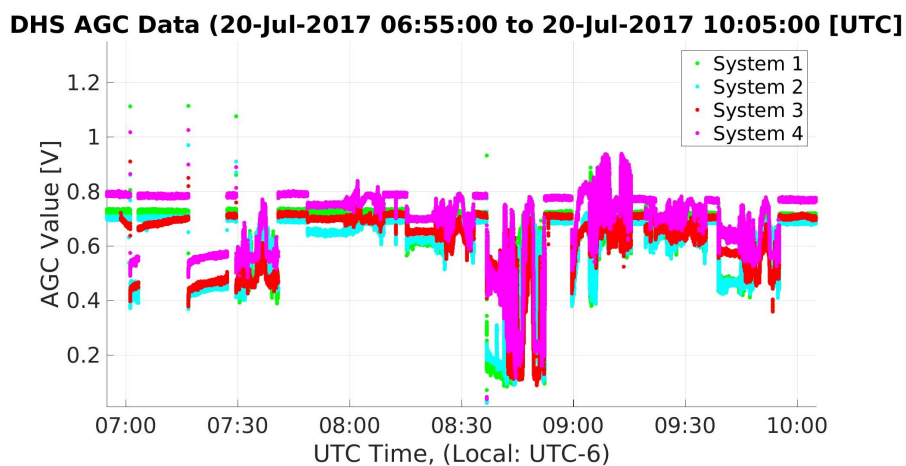


Figure 16. SiGe AGC level over time on Night 1

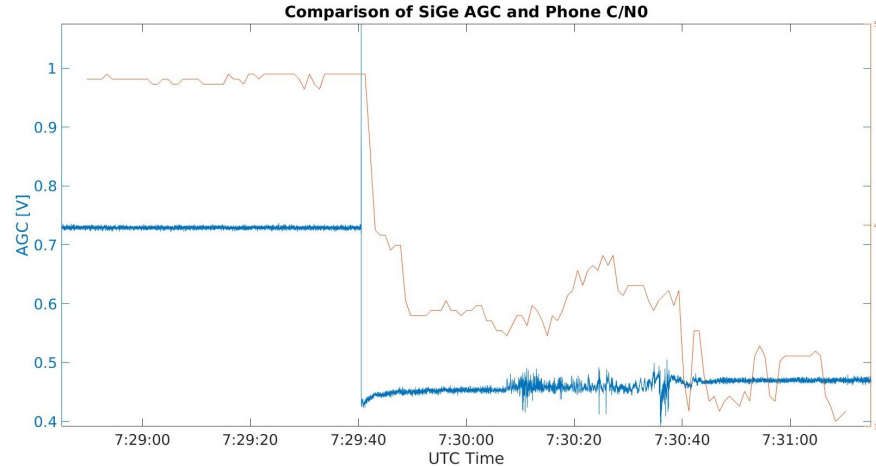


Figure 17. Zoomed comparison of SiGe AGC (blue) and collocated phone C/N0 (red)

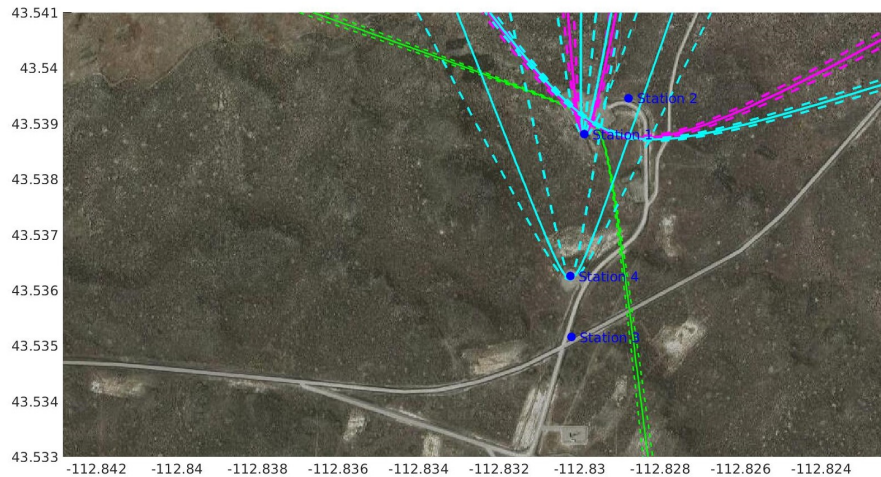


Figure 18. Time Difference of Arrival position hyperbolas from Night 1, 7:29 am GMT, dashed line indicate 1 std bounds

FUTURE WORK

Increased receiver density

A test with a higher density of smartphones would make locating the jammer visually easier and would provide more data to be analyzed. With enough samples gathered, and a high enough receiver density, successful localization with PDOA would be more likely. The limiting factor for TDOA is time accuracy and a high time resolution, which cannot be achieved on smartphones. On the SiGe Sampler, however, TDOA worked well enough with the limited phone density.

Improved robustness and spoofing detection with AGC

Android OS version 8 (just released at the time of writing) introduces an API to get AGC level of the smartphone GNSS chipset. While AGC has in general been shown to be useful for both GNSS interference and spoofing detection [10], the utility of AGC within a smartphone needs to be validated. The electromagnetic environment in smartphones is volatile due to the various radios that are located inside the limited space, but the extent of its effects on AGC measurements and spoofing detection should be researched once Android 8 is out and more phones with supporting hardware appear on the market. Furthermore, tracking satellites on multiple frequencies, that might not be jammed, will help maintain location and time fix, improving the jammer localization process.

Better data processing algorithms

Current data processing methods are quite simple, but more advanced data processing or averaging algorithms may give better results, in terms of how easy it is to subsequently detect jamming or differentiate going indoors from true jamming.

Leverage other smartphone information

The received GNSS power in a smartphone can be influenced by many factors. Due to the smartphone design constraint, its orientation could significantly influence the power of received signals. Since design and hardware between different phones differs, the ideal phone orientation for one phone might not be ideal for another. Quantization of orientation's effects on measurements would be beneficial in future test. Afterwards, the use of internal smartphone measures of orientation may help calibrate out the effects of orientation. Similarly being indoors, in a pocket or under foliage can also reduce signal power and C/N0. Smartphone sensors such as light sensor, other radio-frequency measures, etc. may be useful for detecting these scenarios and reducing false positives.

Data compression

There is a lot of spatial redundancy when ASCII encoding and JSON format are used, as well as temporal redundancy. Using binary encoding reduces spatial redundancy, but temporal still remains, thus bandwidth savings could be achieved and load on cellular towers reduced if messages were batched together, compressed and then sent to the server.

Authentication

The backend that was developed for this test accepts incoming connection from any actor, allowing for easy abuse of the experimental system. Authentication between an Android device and the backend should be implemented for a more robust solution.

CONCLUSION

The work presented in this paper has laid a practical foundation towards using smartphones for crowd-sourcing GNSS jamming detection, by building an Android application and support server software. It demonstrates that smartphones are indeed a suitable platform for collection of GNSS observables, especially when newer hardware and Android OS versions become available. The JamX GNSS jamming results showed that measurements from multiple phones give insight into jammer's location, given the constraint that the jammer is on the road. All the different phones behaved the same way with a jammer presents -- big variation from the nominal levels, a dip in C/N0, which is easy to spot. Increasing the phone density would provide significant benefits. There was also some success in TDOA localization using the SiGe Sampler. The work is still in its preliminary phase and additional development will improve performance.

ACKNOWLEDGMENTS

The authors thank the DHS Science and Technology (S&T) Directorate for sponsoring and supporting the JamX 2017 exercise. The authors also thank the Stanford Center for Position Navigation and Time (SCPNT) for supporting this work.

DISCLAIMER

The views expressed herein are those of the authors and are not to be construed as official or reflecting the views of the European Commission.

REFERENCES

- [1] Joseph C. Grabowski, "Personal Privacy Jammers: Locating Jersey PPDs Jamming GBAS Safety-of-Life Signals," GPSWorld, April 2012, <http://gpsworld.com/personal-privacy-jammers-12837/>
- [2] Jeff Coffed, "The Threat of GPS Jamming: The Risk to an Information Utility," Report of EXELIS, Jan, 2014.
- [3] Logan Scott, "J911: The case for fast jammer detection and location using crowdsourcing approaches," In Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011), pages 1931–1940, 2001.

- [4] Barry Richard Secrest, David Richie, and Brian Rapp, "Mathematical underpinnings of crowdsourced GPS jammer geolocation using signal strength readings from various locations," 7th International Conference on Localization and GNSS, 2017.
- [5] Gorkem Kar, Hossen Mustafa, Yan Wang, Yingying Chen, Wenyuan Xu, Marco Gruteser, and Tam Vu. Detection of on-road vehicles emanating GPS interference. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pages 621–632. ACM, 2014
- [6] I. Kraemer, P. Dykta, R. Bauernfeind, and B. Eissfeller, "Android GPS Jammer Localizer Application based on C/N0 measurements and pedestrian dead reckoning," In Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION/GNSS 2012), pages 3154–3162, 2012.
- [7] E. G. Manfredini, D. M. Akos, Y.-H. Chen, S. Lo, T. Walter, P. Enge, "Effective GPS Spoofing Detection Utilizing Commercial Receivers Technology," submitted to Navigation: The Journal of the Institute of Navigation.
- [8] J. Gross, T. E. Humphreys, "GNSS Spoofing, Jamming, and Multipath Interference Classification using a Maximum-Likelihood Multi-Tap Multipath Estimator," ION ITM, Monterey Jan 2017.
- [9] M Khider. Gnsslogger. <https://github.com/google/gps-measurement-tools> , 2017.
- [10] D. M. Akos, "Who's Afraid of the Spoofer? GPS/GNSS Spoofing Detection via Automatic Gain Control (AGC)", NAVIGATION, Journal of The Institute of Navigation, Vol. 59, No. 4, Winter 2012, pp. 281-290.
- [11] Evan Mason, "9-1-1 system." https://web.archive.org/web/20170920022734/https://en.wikipedia.org/wiki/File:9-1-1_System.svg, 2012.
- [12] Bidouille, "Android version distribution history." <https://web.archive.org/web/20170910233737/https://www.bidouille.org/misc/androidcharts>, 2017.