Redesigning the Control Segment of GPS for Resilience Against Byzantine Fault

Kelly Fam and Claudia Richoux,
Information Assurance Internship,
Air Force Research Laboratory,
Information Directorate

January 17, 2018

Executive Summary

We were tasked with the redesign of a ground control system for the Global Positioning System, GPS, that holds robust against points of failure. Currently, the ground portion of GPS consists of four ground antennae (GA), updated once daily with adjustments for the time and position of the satellites, twelve monitor stations (MS) with predetermined locations that report the error in their GPS-determined position once every six seconds, and one Master Control Station (MCS) that processes input using a Kalman filter for error correction.

There are several problems with this approach: first, the centralization of a MCS creates a point of failure. One hot-swapped backup (BMCS) endangers the situation. The functionality of two difficult-to-replace control stations stands between an attacker and the irrecoverable destruction of the GPS system. Second, the Kalman filter does not react to instantaneous large changes, but will react to small errors over time. The Kalman filter may push towards dangerous, inaccurate results if one monitor station or satellite reports inaccurate position calculations or a misaligned clock. Finally, the MCS opens itself to updates from AFWA, USNO, or any of the contractors that produced components of the system. Although this will allow zero-day vulnerabilities to be patched the second they are discovered, it is more secure to have MCS step through its routine the same way than to respond to external events.

By redesigning the entire Control Segment with the implications of possible malicious input, we can design a system that is naturally resilient against failure of any one component.

Assumptions

We start with a few assumptions about the state of the GPS system. First, we only trust that half of the system's components are functional and within error specifications on every measurement at any given time. Next, we assume that each substation is equipped with a PUF (Physically Unclonable Function) to generate cryptographic keys. Finally, we assume that every system trusts its own hardware. All other data is untrusted and malicious.

Solution

To solve the centralization problem, we split up the points of failure with the distribution of MCS across multiple physically separated machines that independently run the Kalman filter and independently poll the MSs and GAs for updates and information. The second issue of Byzantine faults can be corrected using a democratic consensus algorithm, alongside a possible private blockchain, to ensure that no station starts 'lying' to others or causes a Kalman filter to drift off course. The third problem of vulnerable code on the MCS is rectified by having the MCS be a client, rather than a server, and forcing all other components to react to the MCS schedule. Cryptographically composed, this new design ensured with PUFs may eliminate the problem of key security.

1 Problem Statement

In this report, we aim to design an improved ground control system for the Global Positioning System (GPS) that is robust against attack or component failure.

2 Background

Our lives are powered by technology, and that technology stays in sync with the time and location data provided by GPS. Errors in this position or timestamp can cause catastrophe; GPS-based attacks can take out wide swaths of the North American power grid [1] or misdirect artillery [2]. Because of our reliance on GPS, its security and trustworthiness are of paramount importance to national security.

GPS Structure

GPS consists of four important components: Satellites, Monitor Stations (MS), Ground Antennae (GA), and the Master Control Station (MCS). The MS, GA, and MCS work together to correct the clocks and atlas of the satellites, and ensure that end users have minimal error in their calculated positions. [10]

Satellites

The 24-32 Satellites orbit the earth twice daily in a constellation designed so that at least four satellites will be visible at any time from any location. They carry stable clocks, which are synchronized at least daily with each other and with the US Naval Observatory (USNO). GPS Satellites transmit their time and position every six seconds. These transmissions contain the data needed for GPS users to solve the navigation equations and determine their positions.

Monitor Stations

The 12 Monitor Stations are at predetermined locations around the globe whose coordinates are specified to high accuracy. MSs continuously receive GPS transmissions from the satellites and use them to calculate their position. From the difference in the MS known and calculated positions, it is possible to determine how the clocks and positions should be adjusted to minimize error. Now, Monitor Stations send their results back to the MCS every six seconds, where they are handed off to the Kalman filter for further processing.

Ground Antennae

The four Ground Antennae are responsible for transmitting the position updates as determined by MCS up to the Satellites. Each satellite needs to be updated at least daily, but can be updated as often as three times daily.

Master Control Station (and Backup)

Schriever, or the Master Control Station, controls every transmission in the network. Monitor Stations report to MCS, and Ground Antennae get their transmissions from MCS. MCS also processes weather updates from AFWA, time updates from USNO, and software updates from the contractors who produced the GPS components. The Backup MCS is a "hot backup", always ready to take over in case a compromised MCS. This is fine if we assume that there will never be an attack on GPS that targets both MCS and BMCS. Unfortunately, this is not a safe assumption to make. If MCS and BMCS are both taken down, the entire GPS system will drift irrecoverably out of sync. Thus, the MCS and BMCS are dangerous points of failure for the entire GPS system.

Security

The current GPS system was designed from the perspective of always trusting either MCS or BMCS to be reliable. This is an unsafe assumption to make, because it means that the entire GPS system (and thus our nation's infrastructure) could be paralyzed with two well-executed attacks. Additionally, there are several other implementation problems with the current model of GPS. Currently, all components are assumed to be trustworthy most of the time, and have a standard Gaussian distribution of errors. This does not account for one measurement consistently being incorrect in the same way, and a Kalman filter cannot account for these kinds of faults. Democratic consensus, analyzing each component as if it might fail at any moment, and keeping a ledger of each component's actions could prevent faults of this kind. Finally, code that can be controlled by external events, or that has complex control flow, is vulnerable in many ways that simpler code would not be. [3] [4] The current model, where MCS is acting as a server and responding to requests from other components, could be improved by having MCS work on a strict schedule and call out to other components when it is ready for their data. Currently, the GPS system is not resilient against certain kinds of malicious or faulty components, and these faults could cause catastrophic damage to system integrity.

3 Assumptions

To redesign GPS, we make as few assumptions as possible about what components of the system can be trusted. Any given component can trust its own hardware, and will have specific rules about what kinds of transmissions it can respond to, from who, and in which ways. It can trust that there is only one copy of any given station's cryptographic key. Finally, as a constraint to make the Byzantine trust problem solvable with minimal loss of generality, any given station can assume that at least half of the stations of a given type are honest and functioning properly. All other data is assumed to be malicious or faulty. These assumptions are the bare minimum needed to design a system that is theoretically secure against the most possible types of attacks.

4 Techniques and Tools

The first technique we used was point-of-failure analysis. We listed all possible components of the GPS system, and reasoned about which ones could be disabled to cause significant effects to the entire system. The goal of this was to determine what parts of the system might need to be distributed in order to prevent an attacker from taking down the entire system with one flaw. [5]

To implement a zero-trust GPS Ground Component, it is necessary to think in terms of cryptographic trust. [7] Starting from basic cryptographic principles, we assumed that no component of the system was trustworthy, and then progressively added justifiable assumptions of trust to components until the system was functional. To choose what to trust, we used basic principles of Formal Verification (alongside common sense reasoning of what would be most difficult to compromise) to reason about what components to trust. [8]

In order to safely trust components, we used some cryptographic and programming techniques in order to harden the system against common types of attacks against distributed systems. Specifically, Physically Unclonable Functions (PUFs) allow for cryptographic secrets that can be stolen, but not duplicated. [6] This allows the system to trust the results of consensus algorithms. Also, the stations run mostly linear code with minimal responsiveness to user input to ensure the system is hardy against maliciously crafted input. [3] [4]

5 Problem Solution

This chapter explains the methodology of our approach to find the solution to secure the command and control of the GPS.

5.1 Analysis of the Daily Life of GPS

To start, we examined the current flow of information through the GPS system by writing a Python application to represent MCS operations during a 24-hour period. We used this analysis to learn about GPS and determine that it is possible to write MCS code linearly to avoid input-based vulnerabilities. Finally, we found that since all data flows through Schriever, the current system has a single point of failure that must be eliminated in a resilient GPS solution.

5.2 Cryptographic Tools

In 2008, Satoshi Nakamoto introduced blockchains, which are distributed ledgers of digital records saved in chains of blocks. Because it takes either computational work or a private key to add to a Proof-of-Work or permissioned blockchain, it is impossible for an attacker to rewrite this ledger without controlling at least half the network. [11] Blockchains present a solution to the Byzantine Generals Problem. The network remains resilient to these kinds of faults as long as at least half of the nodes remain healthy. Blockchains are applicable to a distributed GPS system, because alongside a cryptographically validated consensus mechanism, they allow nodes to verify the honesty and error rate of their peers. This ability to trust the system as a whole without trusting any one peer allows the GPS system to tolerate faulty or malicious components without losing integrity as a whole.

While reading about Blockchains and their applicability to Byzantine trust, we wrote a brief about their applicability to the Air Force for the Vice Chief of Staff of the Air Force. One example of applicability that we proposed was using a blockchain to store Common Access Card (CAC) data to prevent card duplication or reactivation after being destroyed.

This summer, we also learned about PUFs for cryptography. A secret key that cannot be cloned allows a node's peers to make a decision about whether it is trustworthy or not, because there is no possibility of an attacker with a copy of the key performing malicious activities while the original node remains honest. If communications with node A's key are consistently going against the majority on consensus algorithms, or its signed calculations are consistently outliers compared to its peers, the rest of the network can assume with certainty that node A is compromised.

5.3 Decentralized GPS

From what we now know about cryptographic tools and the needs of the GPS system, we feel confident recommending that Schriever needs to be split off into multiple "Tiny Schrievers", or TS. A distributed system will be much harder to compromise, because an attacker has to compromise half of the total nodes, rather than just disable MCS and its backup.

Second, there needs to be more resilience against Byzantine Fault than just a Kalman filter. Rather than simply averaging every measurement into calculations, the GPS system could keep a blockchain record of all transmissions between nodes in order to identify nodes that may be malfunctioning or compromised. If a node is consistently an outlier in some unusual respect, its error is no longer Gaussian, so its data should not be used for decision making until a human operator verifies the node's integrity. A blockchain with a cryptographically validated democratic consensus algorithm will ensure that faulty nodes are excluded from calculations, and that all peers agree on a decision before they transmit it to a satellite.

Finally, MCS (distributed or otherwise) should run code that cannot be manipulated by bad user input. This means that MCS should be a client, rather than a server, and nodes that need to communicate with MCS must wait to be polled at a scheduled time. All stations should run linear code that carefully segregates code from data, and they should drop any connection that sends improperly formatted data as soon as an error is found.

These code structure guidelines will render the stations themselves difficult to attack, and the distributed zero-trust network structure will make attacks on the network as a whole unfeasible.

5.4 GPS 3.0 Simulation

To demonstrate our ideas for a more secure GPS system, we implemented some minimally functional Python code (Appendix A) that demonstrates the secure flow of information through the system. The code has several distributed TS that each signal when they are performing computations and sending transmissions. For the second suggestion, blockchains and consensus technology were not implemented in this tabletop demonstration, but shared encryption keys were used to sign and validate all transmissions with simplified HMAC. For the third suggestion, the TS were clients that ran on an internally timed schedule, while all other components behaved as servers responding to their requests. All code was as linear as possible, and all communications between services are either validated or comments indicate what validations should be performed in production. Packets that do not pass validation are immediately dropped.

A graphical interface displays all communications throughout the network. In this example, the entire network is running on one machine, with communications occurring over TCP. However, the code was written modularly with lots of comments to enable easy addition of features, like running each 'node' on a different physical machine. This code is meant to be easily evolved into a tabletop demonstration of the full GPS 3.0 standard. A datagram standard for inter-node communications is specified in Appendix B.

6 Risk Assessment

This section will analyze the potential risks for any assumptions made in the design of this system, and offer alternate solutions in the case that this assumption is invalid.

6.1 Self-Trust

If a station cannot trust that its own Hardware and Ruleset are not compromised, there is no way for a station to make any decision at all. If it assumes that any conclusion it comes to is inherently flawed, it cannot take any actions based on that conclusion that will not themselves be flawed. Therefore, a station that does not trust itself cannot take any safe actions, and the system as a whole cannot take any safe actions without any stations that can take safe actions. Example: a TS that does not believe its own outputs are correct cannot, in good faith, send those outputs forward to a GA for transmission.

6.2 Key Uniqueness

If keys can be duplicated or compromised, it becomes impossible to trust any signed message. Assuming that a station is one of either functional, broken, or compromised makes the consensus procedure possible. If we assume that some station A is fully functional, but an attacker is lying on its behalf to some of its peers, the station's vote cannot be categorized as honest or dishonest for the purpose of deciding whether to exclude its measurements from future calculations without unnecessarily degrading the system.

6.3 51% Attacks

If a station cannot trust that at least half of the other stations are telling the truth, it cannot rely on a democratic consensus of the other stations to rule its decisions. Trusting a lower percentage of stations and using a plurality rather than a majority to make decisions would also work, but it would not be possible to keep a ledger of past system actions without 51% trust. This would make it impossible to ensure that all stations of a certain type take the same action, given potentially differing inputs from potentially compromised inputs. For example, a GA that does not trust 51% of its peers cannot participate in a blockchain to ensure that compromised peers are not selectively lying and creating a Byzantine Generals Problem to lead satellites off course.

References

[1] Akkaya, Ilge, Edward A. Lee, and Patricia Derler. "Model-based Evaluation of GPS Spoofing Attacks on Power Grid Sensors." 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES) (2013): n. pag. Web.

[2] Gould, Joe. "Guided-Bomb Makers Anticipate GPS Jammers." Defense News. N.p., 22 July 1970. Web. 30 July 2017.

[3] Buchanan, Erik, Ryan Roemer, and Stefan Savage. "Return-Oriented Programming: Exploits Without Code Injection." Black Hat USA 2008 Briefings. NV, Las Vegas. Aug. 2008. Web.

[4] Kadous, Waleed. "Computer Science: What Are the Advantages and Disadvantages of Von Neumann Architecture vs Harvard Architecture?" Quora. Quora, 9 July 2015. Web. 30 July 2017.

[5] Kangasharju, Jussi. "What Is a Distributed System?" University of Helsinki. Lecture.

[6] Devadas, Srini. "Physical Unclonable Functions and Applications." MIT Security. Boston, MA. Lecture.

[7] Perrin, Chad. "The Meaning of Cryptographic Trust." TechRepublic. N.p., 14 Dec. 2010. Web. 31 July 2017.

[8] Chin, Shiu-Kai. "Certified Security by Design for the Internet of Things." Cyber-Assurance for the Internet of Things (2016): 1-99. Web.

[9] Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3), 382-401. Retrieved from http://www-inst.eecs.berkeley.edu/ cs162/fa12/handouts/Original_Byzantine.pdf

[10] GPS: The Global Positioning System. Retrieved July 29, 2017, from http://www.gps.gov/

[11] Nakamoto, Satoshi. ”Bitcoin: A Peer-to-Peer Electronic Cash System.” n.p. 2009. Web. 22 June 2017.