Security of GPS/INS based On-road Location Tracking Systems

Sashank Narain, Aanjhan Ranganathan, and Guevara Noubir
Northeastern University, Boston, MA, USA

***Abstract*—Location information is critical to a wide variety of navigation and tracking applications. GPS, today's de-facto outdoor localization system has been shown to be vulnerable to signal spoofing attacks. Inertial Navigation Systems (INS) are emerging as a popular complementary system, especially in road transportation systems as they enable improved navigation and tracking as well as offer resilience to wireless signals spoofing and jamming attacks. In this paper, we evaluate the security guarantees of INS-aided GPS tracking and navigation for road transportation systems. We consider an adversary required to travel from a source location to a destination and monitored by an INS-aided GPS system. The goal of the adversary is to travel to alternate locations without being detected. We develop and evaluate algorithms that achieve this goal, providing the adversary significant latitude. Our algorithms build a graph model for a given road network and enable us to derive potential destinations an attacker can reach without raising alarms even with the INS-aided GPS tracking and navigation system. The algorithms render the gyroscope and accelerometer sensors useless as they generate road trajectories indistinguishable from plausible paths (both in terms of turn angles and roads curvature). We also design, build and demonstrate that the magnetometer can be actively spoofed using a combination of carefully controlled coils. To experimentally demonstrate and evaluate the feasibility of the attack in real-world, we implement a first real-time integrated GPS/INS spoofer that accounts for traffic fluidity, congestion, lights, and dynamically generates corresponding spoofing signals. Furthermore, we evaluate our attack on ten different cities using driving traces and publicly available city plans. Our evaluations show that it is possible for an attacker to reach destinations that are as far as 30 km away from the actual destination without being detected. We also show that it is possible for the adversary to reach almost 60–80% of possible points within the target region in some cities. Such results are only a lower-bound, as an adversary can adjust our parameters to spend more resources (e.g., time) on the *target* source/destination than we did for our performance evaluations of thousands of paths. We propose countermeasures that limit an attacker's ability, without the need for any hardware modifications. Our system can be used as the foundation for countering such attacks, both detecting and recommending paths that are difficult to spoof.**

I. Introduction

The ability to track one's location is important to a wide variety of safety- and security-critical applications. For example, logistics and supply chain management companies [1], [2], [3] that handle high-value commodities (e.g., currency notes) continuously monitor the locations of every vehicle in their fleet carrying valuables to ensure their secure transportation to the intended destination. Law enforcement officials use ankle bracelets [4], [5] to monitor the location of defendants or parole and notify them if the offender strays outside an allowed area. Ride-hailing applications such as Uber and Lyft use location information for tracking, billing, and assigning drivers to trips. Furthermore, the locations of public transport [6], [7], [8] are continuously monitored to ensure smooth and timely operation of services. With the advent of autonomous vehicles and transport systems, the dependence on location information is only bound to increase. The majority of above applications rely on GPS [9], the de facto outdoor localization system in use today. It is estimated that more than 8 billion GNSS[1] devices [10] will be in use by the year 2020.

However, it has been widely demonstrated that GPS is vulnerable to signal spoofing attacks. One of the main reasons is the lack of any form of signal authentication. It is today possible to change the course of a ship [11], force a drone to land in a hostile area [12] or fake the current location in a road navigation system [13] by simply spoofing GPS signals. The increasing availability of low-cost radio hardware platforms make it feasible to execute such attacks with less than few hundred dollars worth of equipment. There has been several evidences of jamming and spoofing reported in the media. For example, as quoted in Gizmodo [14] "Because the toll-taking for commercial trucks relies on GPS tracking, they can avoid paying through jamming. If a $45 device made your daily commute free, you too might be tempted to commit a federal crime." Another report [15] mentions "Gary Bojczak admitted buying an illegal GPS jammer to thwart the tracking device in his company vehicle". Several cryptographic [16], [17], [18], [19] and non-cryptographic [20], [21], [22], [23], [24], [25], [26] countermeasures have been proposed to detect or mitigate signal spoofing attacks. These techniques are either unreliable (e.g., large number of false alarms), effective only against naive attackers or require modifications to the GPS receiver/infrastructure. Alternate localization technologies using Wi-Fi or Cellular [27], [28] lack the accuracy and coverage required for the mentioned applications, consume significant amount of power and are susceptible to interference.

Inertial navigation i.e., the use of sensors such as accelerometer, gyroscope and compass to navigate during temporary GPS outages have been around for decades, specifically in aircrafts, spacecrafts and military vehicles [29], [30], [31]. The advancements in sensor manufacturing technologies have resulted in widespread integration of these sensors into many commonly used devices such as smart phones, tablets, fitness trackers and other wearables. Many vehicle tracking and au-

[1]Global Navigation Satellite Systems (GNSS) is an umbrella term for satellite based localization systems such as GPS, Galileo, Glonass etc.

tomotive navigation systems have integrated GPS with inertial measurement units to improve localization and tracking of individual vehicles [32], [33], [34], [35]. Inertial sensors are key to the balancing and navigation technologies present in modern segways. Low-cost inertial sensors have also proliferated into the consumer drone industry today. One of the key advantages of inertial navigation is its robustness and resilience to any form of wireless signal spoofing and jamming attacks as there is no need for the sensors to communicate or receive information from any external entity such as satellites or other terrestrial transponders. This makes them very attractive for use in security- and safety-critical localization and tracking applications where GPS (or any wireless) spoofing and jamming attacks are a concern. The main drawback of inertial navigation units is the accumulating error of the sensor measurements. These accumulated sensor measurement errors affect the estimated position and velocity over a longer duration of time and hence limit the maximum period an inertial unit can act independently. This affects aerial and maritime navigation capabilities significantly as the tracked vehicle has all the six degrees of freedom to move. However, in the context of road navigation, the vehicle is limited by the road network and can only navigate within the constraints of these existing roadways. These inherent constraints imposed by the road networks have made low-cost inertial sensors very valuable for quick attack detection and immediate tracking of cheating entities [36], [37], [38], [39], [40], [41].

In this work, we evaluate the security guarantees of GPS/INS based on-road location tracking systems. Specifically, we address the following research questions: Given a geographic area's road network and assuming that both GPS and inertial sensor data are continuously monitored for tracking an entity's location, is it possible for an attacker to fake his navigation path or final destination? If yes, what are the attacker's constraints and possibilities? Can we exploit the physical motion constraints that exist in an urban road network and design a secure navigation algorithm that generates travel routes that are hard to spoof? For example, can a driver of a vehicle carrying high-value commodities (e.g., currency notes) spoof his assigned route and deviate without detection by the monitoring center? Can a parole with GPS/INS ankle monitor spoof his location and travel routes without detection?

We make the following contributions. First, we demonstrate that it is indeed possible for an attacker to hijack vehicles far away from the intended destination or take an alternate route without triggering any alarms even though the GPS location as well as inertial sensors are continuously monitored. We develop a suite of algorithms which we refer to as ESCAPE that leverage the regular patterns that exist in urban road networks and automatically suggests potential alternate *escape* routes while spoofing the assigned route with start point s, and end point d. Spoofing means that the adversary will travel on an alternate path indistinguishable from the spoofed (assigned) path. Our ESCAPE suite of algorithms accounts for intersections turn angles, roads curvatures, and magnetometer bearings to calculate the escape routes an attacker can take without detection while spoofing. We implement a real-time integrated GPS/INS spoofer that can dynamically generate spoofing signals depending on the current traffic fluidity, traffic lights, and any unexpected congestion the attacker might encounter while driving the escape path. We note that our prototype is, to the best of our knowledge, the first integrated GPS/INS spoofing system that can, in real-time, dynamically adjust the spoofing signals based on the true conditions. We further evaluated our attack's performance using open source city plans and driving traces in ten major cities across the globe. Our simulation results show that an attacker can potentially take the vehicle as far as 30 km before the monitoring system can detect a potential attack. We also drove on ten different paths of varying lengths using our real-time integrated spoofer and our results show that the attacker can hijack the vehicle to more than 2 km (the average deviation for our city), without once losing a GPS lock and with a maximum delay of $60ms$ between the real and spoofed paths. Note that even after detection, the tracking system has no knowledge of the true location. Our attack affects several services and applications with effective monetary value running into several millions of dollars. Our attacks essentially renders the gyroscope and accelerometer useless by generating paths acceptable to the monitoring system, but have a signature indistinguishable from the trajectory effectively traveled by the adversary. For the magnetometer, a sensor that can play a critical role in detecting the incongruence of the claimed trajectory with the measured heading, we built and demonstrated the effectiveness of a magnetometer-spoofing device that physically generate a magnetic field compatible with the spoofed trajectory. Finally, we turn around our ESCAPE suite of attack algorithms to build a countermeasure that the tracking services can run to mitigate such spoofing attacks. Specifically, we modified ESCAPE to output secure navigation routes that can be assigned given a start and end point that limit the attacker's possibilities.

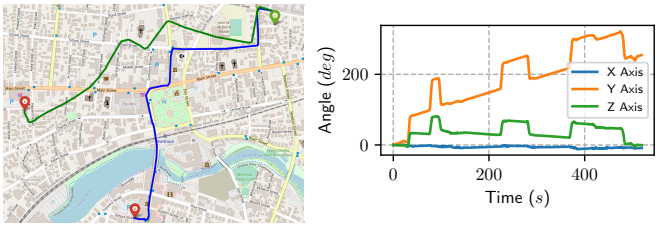
II. Background

A. Overview of GPS

GPS is today the de-facto outdoor localization system used. GPS consists of more than 24 satellites orbiting the earth. Each satellite is equipped with high-precision atomic clocks and transmits messages referred to as the *navigation messages* that are spread using pseudorandom codes unique to that satellite. The GPS receiver on the ground receives these navigation messages and estimates their time of arrival. Based on the time of transmission contained within the navigation message and its time of arrival, the receiver computes its distance to each of the visible satellites. Once the receiver acquires the navigation messages from at least four satellites, the GPS receiver estimates its own location and precise time.

B. GPS Spoofing Attacks

Civilian GPS is easily vulnerable to signal spoofing attacks due to the lack of any signal authentication and the publicly known spreading codes for each satellite, modulation schemes,



(a) Path drift (b) Gyroscope drift

Fig. 1: The constraints imposed by the road networks lead to better accuracy in tracking road applications. Blue path - route estimate with road constraints. Green path - no constraints.

and data structure. A GPS signal spoofing attack is a physical-layer attack in which an attacker transmits specially crafted radio signals that are identical to authentic satellite signals. In a signal spoofing attack, the objective of an attacker may be to force a target receiver to (i) compute a false geographic location, (ii) compute a false time or (iii) disrupt the receiver by transmitting unexpected data. Today, with the increasing availability of low-cost radio hardware platforms [42], [43] and open source GPS signal generation software [44], it is feasible to execute GPS spoofing attacks with less than $100 of equipment. GPS signal generators can be programmed to transmit radio frequency signals corresponding to a static position (e.g., latitude, longitude and elevation), or simulate entire trajectories. For example, an attacker can spoof the navigation route of a vehicle carrying high-value items and hijack it to arbitrary locations without raising alarms. The operators of ride hailing services can fake the route taken. Furthermore, GPS spoofing attacks can delay or even prevent emergency support services from reaching intended destinations.

C. Inertial Sensors Aided Navigation and Tracking

The need to operate effectively in scenarios where GPS is inaccessible, unreliable or potentially jammed or spoofed by adversaries has led to the increased interest in building complementary navigation solutions and spoofing detection techniques. Several countermeasures and alternative localization techniques have been proposed. Of them, inertial sensors are emerging as a popular choice for two main reasons. First, inertial measurements are not affected by wireless jamming and are therefore resilient to denial of service attacks. Second, their widespread availability in majority of modern smartphones makes them easy to deploy and integrate into existing navigation and tracking infrastructure without the need any hardware or software modifications to the GPS receiver.

Inertial navigation is the process of integrating the readings of select sensors such as accelerometers, gyroscopes, and magnetometer into a complete three-dimensional position, velocity, and orientation solution. Inertial navigation systems are classified as *dead-reckoning*, since the estimation process is iterative and uses prior information i.e., calculating from some previously known navigation solution. Accelerometers measure both gravitational and non-gravitational acceleration along each of the three axes. The gyroscopes measure the rate at which an object is rotating, and are used to compute the attitude and heading of the object. The gyroscope measurements aid the accelerometer in figuring out the orientation of the object. Typically, sets of three accelerometers and three gyroscopes, both orthogonally aligned, are combined into a single inertial measurement unit (IMU). The setup commonly contains additional analog and digital circuitry, including conversion and calibration components. The magnetometer measures the magnetic fields and thus determines the cardinal direction to which the object is pointing.

One of the main drawbacks of low-cost inertial sensors (e.g., MEMS [45]) is that the process of dead reckoning in general, results in a build-up of errors over the course of the measurement. Since the position, velocity, and attitude updates are products of single or double integration of raw inertial sensor readings, the errors propagate and affect the final position, velocity and attitude estimates. For example, due to the single integration performed on angular rate measurements, a constant gyroscope bias will produce a linearly growing angular error, the gyro noise will produce a 'random walk' growing with the square root of time. The double integration required to transform the accelerometer output to position produces a quadratically growing position error and a second-order 'random walk', for a constant accelerometer bias and white noise respectively. In numerical terms, a $25\,\mu\mathrm{m^2 s^{-1}}$ accelerometer bias ($\approx 245\,\mu\mathrm{g}$) of a navigation grade sensor would produce a $1.59\,\mathrm{km}$ position error in one hour. The aggravation of sensor errors becomes critical to aviation and maritime applications as the vehicles have more degrees of freedom to move. However, on road, the vehicles are limited by the available road networks and are therefore severely constrained in their possible trajectories. Figure 1 illustrates how the bias errors affect the final position estimates in a road navigation scenario (with motion constraints) and aerial (without any motion constraints). These constraints imposed inherently by the road networks has led to the emergence of using inertial sensors to complement GPS navigation and tracking solutions. Moreover, the inertial sensors are largely immune to jamming which makes them invaluable to the safety and security-critical applications described previously.

III. Spoofing INS-aided Localization Systems

In this section, we demonstrate spoofing attacks on road navigation and tracking applications that rely on both GPS and the inertial sensors for the localization. To the best of our knowledge, this is the first demonstration of spoofing attacks on GPS/INS localization systems.

A. System and Attacker Model

Our attack is independent of how the GPS/INS system is deployed i.e., it can either be an app on a trusted smartphone or a specialized tracking device (e.g., ankle monitors) installed on the entity of interest. The main objective of the monitoring system is to keep track of the location and navigation routes of the entities. We assume an attacker capable of generating and

transmitting fake GPS signals corresponding to any location or navigation route of his choice using tools such as *gps-sdr-sim* [44]. The goal of the attacker is to spoof his location and navigation trajectory without being detected. For example, the attacker can try to deviate from an assigned navigation route and reach as far away as possible from the intended destination before an anomaly is detected and an alarm raised. At that moment, the adversary's location remains undetermined. Alternately, the attacker starts and ends at the intended locations, however using a different route than the one being reported to the monitoring station. We assume an attacker with full physical access to the entity being tracked and is aware of the GPS/INS system deployed for monitoring. For this work, we assume that the tracking device itself is tamper-proof. For example, the attacker can be a driver of a cargo company (or a hijacker) who has full access to the vehicle. He regularly drives this vehicle to transport high-value goods, and is aware of the GPS and INS based tracking system employed by the company. However, he cannot modify the software on the smartphone or physically tamper the tracking device.

B. Overview of the Attack

The primary objective of the attacker is to fake the reported navigation route without raising suspicion. Note that simply spoofing GPS signals is not sufficient as the INS measurements will indicate discrepancies between the reported GPS location and the inertial estimates. In order to successfully execute the attack, it is now necessary for the attacker to identify and spoof navigation paths that have similar distances, road curvature, and turn angles to minimize the discrepancies between the INS and GPS estimates. Our system, which we refer to as ESCAPE, exploits the regular patterns that exist in many cities' road networks and identifies navigation paths that are similar to the route that is reported to the monitoring center. As a result, the inconsistencies between the INS and GPS estimates are negligible and the attack is successfully executed.

The attack begins with the attacker providing the start and end points of the assigned trip to ESCAPE. ESCAPE computes two sets of paths: (i) *spoofed paths* and (ii) *escape paths*. The *spoofed paths* are a set of paths that exist between the start and end points of the trip. These are the paths that the attacker will generate fake GPS signals and spoof the receiver to report to the monitoring center. These should be plausible paths for the source and destination locations. For every spoofed path, ESCAPE computes a set of *escape paths* which the attacker can use to deviate from the intended course while executing the spoofing attack. In other words, a *spoofed path* is the route that is reported to the monitoring center and the *escape path* is the true route taken by the attacker to reach an alternate destination. The attacker then picks an escape path that enables him to reach his intended location. The intended location can either be a point far away from the assigned destination (to buy the adversary some time) or just a diversion before reaching the assigned destination. The selected escape path corresponds to a spoofed path which the attacker can use to generate spoofing signals. Figure 2 illustrates an example of a spoofed path generated between two end points in Manhattan (green line from green marker to red marker) and the destinations of the escape paths (red points) generated for this particular spoofed path. Finally, the attack is executed by spoofing the tracking device to report the *spoofed path* while the attacker actually drives the *escape path*.

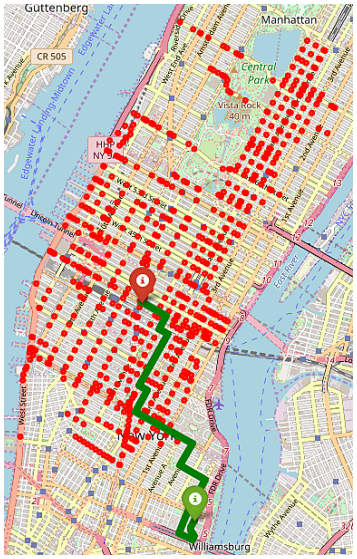
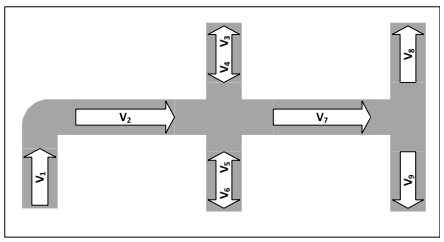


Fig. 2: A spoofed path example in Manhattan and the escape destinations generated for that single spoofed path. Our algorithms generate 100 distinct spoofed paths for a given start and end location, allowing an adversary to undetectably reach an even larger set of escape destinations.

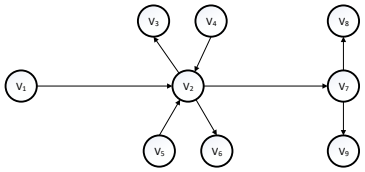
C. Internals of ESCAPE

ESCAPE consists of three main building blocks: (i) graph constructor, ii) spoofed paths generator and (iii) escape paths generator. The graph constructor generates directed graphs based on the road network present in the geographic area of interest. Our attack does not enforce any limits on the geographic area. As the name suggests, the spoofed and escape paths generator blocks are responsible for computing and identifying spoofed and escape paths for the attacker.

1) Graph Constructor: The paths for an area $\mathcal{G}$ are generated from a directed graph $G_{\mathcal{G}} = (V, E)$. We chose OpenStreetMap [46] as the map provider as it has accurate road information for major cities of the world along with various meta-data such as types of roads and buildings. Each geographic area can be represented as $\mathcal{G} = (\mathcal{A}, \mathcal{C}, \theta, \vartheta)$, where $\mathcal{A}$ is a set of atomic sections and $\mathcal{C} = \{\chi = (s, s') | s, s' \in \mathcal{A}\}$ is a set of connections where χ indicates a connection between two atomic sections s and s'. We define an atomic section as a section of road between two intersections, such that it preserves the road's curvature but does not contain turns or sharp curves. A connection is an intersection connecting two atomic sections. These connections may extend the same road or may turn into another road. The turn angle associated with a connection χ is given by the function $\theta(\chi)$ and the atomic



(a) Example Road Network



(b) The Graph Representation

Fig. 3: Sample road and corresponding graph representation.

section's curvature is given by the function $\vartheta(s)$ as defined in Equation 1. We represent each atomic section s by a vertex $v \in V$ and each connection χ by an edge $e \in E$. Figure 3 shows an example road network and the corresponding graph construction. A default speed limit is assigned to each atomic section based on the road type in OpenStreetMap. For example, a 'motorway' symbolizes interstates in the USA that have speed limits $\approx 65mph$. The length, speed limit, and geographic coordinates of the atomic section s are stored as attributes of the corresponding vertex v. The length and speed limit are used to calculate the fastest time of travel between the end points. *It is important to note that this is a one time initialization step for every geographic area.*

2) Spoofed Paths Generator: Spoofed paths generator searches and compiles possible paths between the source and destination points assigned to a specific trip. *The spoofed paths are defined as a set of N routes $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_N\}$ such that $\mathcal{S}_i$ has a higher likelihood of spoofing than $\mathcal{S}_j$, where $i < j$ and $\mathcal{S}_i, \mathcal{S}_j \in \mathcal{S}$. Each route $\mathcal{S}_i$ contains a list of geographic coordinates starting and ending at the input source and destination.* Given the geographic area of the attacker, the algorithm generates paths that maximize the probability of finding similar road curvature and turn angles in other sections of the area. Therefore, it maximizes the number of escape paths. It leverages the fact that urban areas have regular patterns where most roads typically run straight and turn angles are at right angles. This is achieved by implementing a scoring scheme that ranks paths containing such regular patterns higher than other non-regular paths between the same source and destination. Figure 4 shows the curvature and turn angle distribution for Manhattan and provides an intuition for our approach. We note that most turn angles are 90° which implies that given a path with all $\approx 90^\circ$ turns, the probability of finding another path with similar turn angles will be high.

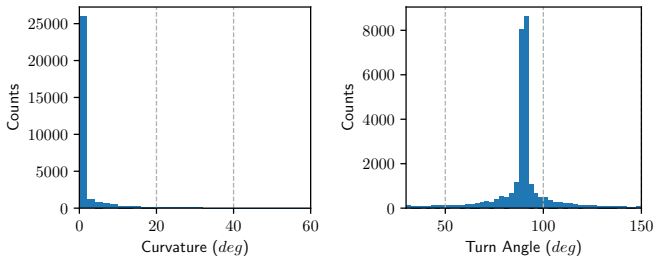
The idea underlying the spoofed paths generator is to find paths that contain attributes likely to be found in other sections of the graph. When such paths are found, they increase the likelihood of finding similar paths to other destinations in the graph. To this extent, we implement a scoring scheme that analyses the road curvature and turn angles of the geographic area and maximizes the score of paths that contain curvature and turns having a higher probability of occurrence. The path search algorithm is implemented as a modified Depth First Search (DFS) algorithm. A typical DFS implementation computes a single path between a given source and destination. This limits an attacker's ability to generate multiple spoofed paths between these end points. We extend the basic DFS algorithm to compute all *plausible* non-cyclic paths between the source and destination. A path is plausible if it is within 20% of the shortest path. To scale the algorithm to large graphs (typical for large cities), we incorporate filtering and scoring functions in order to speed up computation by eliminating unlikely paths and pruning low scoring paths at every iteration.

Input: $G = (V, E)$, $Loc(s)$, $Loc(d)$, N_P
Output: $\mathcal{S} = \{p_1, \ldots, p_{N_P}\}$

1 $Initialization : \mathcal{S} \leftarrow \emptyset$; $p \leftarrow [\,]$; $v \leftarrow \emptyset$
2 $s \leftarrow getSourceVertex(Loc(s))$
3 $d \leftarrow getDestinationVertex(Loc(d))$
4 `GenerateSpoofedPaths(`s, d`)`
5 $\mathcal{S} \leftarrow selectTopPaths(\mathcal{S}, N_P)$
6 **function** `GenerateSpoofedPaths(`s, d`)`**:**
7 $\quad p \leftarrow p + [s]$
8 $\quad v \leftarrow v \cup \{s\}$
9 $\quad$ **if** $s = d$ **then**
10 $\quad\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{p\}$
11 $\quad$ **else**
12 $\quad\quad$ **for** $e \in V$ *such that* $(s, e) \in E$ **do**
13 $\quad\quad\quad$ **if** $e \notin v$ **and** $Filter(s, e, p)$ *passed* **then**
14 $\quad\quad\quad\quad p.score \leftarrow p.score * Score(s, e, p)$
15 $\quad\quad\quad\quad$ `GenerateSpoofedPaths (`e, d`)`
16 $\quad\quad$ **end**
17 $\quad p \leftarrow p - [s]$
18 $\quad v \leftarrow v - \{s\}$

Algorithm 1: Spoofed Paths Algorithm

The spoofed paths generator algorithm (Algorithm 1) takes as input a graph $G = (V, E)$, the source $Loc(s)$ and destination $Loc(d)$ geographic coordinates, and a count of output paths N_P. The algorithm outputs a set of spoofed paths $\mathcal{S}$ sorted by the path score. The algorithm starts by initializing the current path p and a set of visited vertices v (line 1). It uses the attacker's source s and destination d vertices as parameters to `GenerateSpoofedPaths` to recursively compute the output paths (lines 2 – 4). In the end, these paths are sorted by score and the top N_P paths are saved as the final set of spoofed paths $\mathcal{S}$ (line 5). Inside the `GenerateSpoofedPaths` function, the algorithm adds the vertex s to the current path p and the visited set v (lines 7 – 8) and adds this path p to the output set $\mathcal{S}$ when the destination vertex is found (lines 9 – 10).



(a) Curvature Distribution (b) Turn Distribution

Fig. 4: Curvature and Turn Distribution for Manhattan.

Otherwise, the algorithm traverses the path's outgoing edges e such that $(s, e) \in E$. During this traversal (lines 12 – 16), filtering is applied to prune edges that are unlikely to occur (line 13) and a scoring function is applied to rank remaining edges (line 14). The filtering and scoring methodology are described next. The `GenerateSpoofedPaths` function is recursively invoked for each outgoing edge e (line 15). Note that, in the end, the source s vertex is removed from the current path p and visited set v to backtrack and proceed with the depth-first search (lines 17 – 18).

Scoring: Recall that all the vertices of the graph $G = (V, E)$ are atomic sections, and the edges connect two atomic sections (c.f. Section III-C). The turn angle of an edge $\chi = (s, s')$, where $(s, s') \in E$, is given by the function $\theta(\chi)$ and the curvature of an atomic section s is given by the function $\vartheta(s)$. This curvature $\vartheta(s)$ can be computed from the geographic coordinates of the atomic section. Let $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_N\}$ denote the set of bearings computed from N geographic coordinates. Let $\mathcal{B}_0$ be the bearing of an imaginary line connecting the first and last geographic coordinates of this atomic section. The curvature $\vartheta(s)$ of this atomic section is calculated as the normalized absolute difference of all bearings in $\mathcal{B}$ from the reference bearing $\mathcal{B}_0$, i.e.,

$$\vartheta(s) = \frac{\sum_{i=1}^{N} |\mathcal{B}_i - \mathcal{B}_0|}{N}. \tag{1}$$

The set of all road curvatures $\vartheta = \{\vartheta(s') | \forall s' \in V\}$ and turn angles $\theta = \{\theta(\chi') | \forall \chi' \in E\}$ represents the road structure of the geographic area. Figure 4 shows these attributes for Manhattan. Note that most of the calculated curvature values are 0° and most turn angles are at 90°. This is typical of Manhattan and other cities synonymous with grid-like road structures. To use this information for scoring, a probability distribution table is precomputed for the area. This table can be represented as $P(\mathcal{G}) = \{P(c, t) | c \in \vartheta, t \in \theta\}$, where each entry is the probability of occurrence of a specific curvature and turn (angles rounded to the nearest integer) combination.

A path on the graph with M vertices can be represented using each vertex's curvature and the next edge's turn angle, i.e., $p = [(c_1, t_1), \ldots, (c_{M-1}, t_{M-1}), (c_M, 0)]$, where $c_i \in \vartheta$ and $t_i \in \theta$. In the beginning, the path is initialized to a score of 1. For each vertex $\hat{s}$ and edge $\hat{\chi} = (\hat{s}, s')$ added to the path, the probability $P(\vartheta(\hat{s}), \theta(\hat{\chi}))$ is obtained from the table $P(\mathcal{G})$. Note that, owing to the algorithm construction, all connecting edges have equal probability of occurrence and are independent of the current path. Therefore, the score at each vertex is multiplied with the previous path score to calculate the compound probability of all vertices in the path. The final path score is calculated as

$$score = \prod_{i=1}^{M} P(\vartheta(s_i), \theta(\chi_i)). \tag{2}$$

Filtering: The algorithm is designed to generate all paths between the input source and destination. For a large graph, the number of possibilities can be in the order of billions making this search very inefficient. To scale the computation, the algorithm uses the following filters to speed-up the search of plausible paths, while enabling ranking. Given the current path p, source s, edge e and destination d, the algorithm filters the edge when the path's distance summed with the euclidean distance between the edge and destination exceeds a maximum allowed distance, i.e., $d(p) + d(c, d) > F * d(\mathcal{P}_I)$ where $d(.)$ denotes the distance of a path and $\mathcal{P}_I$ denotes the shortest time path between the source and destination. For this work, we set $F = 1.2$ to only allow paths that are similar in distance to the computed shortest path. The algorithm also maintains the best N paths at all times, and any new path p' having a worse score is filtered. For our evaluation, we chose $N = 100$ in order to determine the attack efficiency in many cities for many paths (the algorithm runs in around 1 minute for each source/destination pair). However, a determined attacker with sufficient resources can easily use a larger N to increase the count of spoofed paths. Furthermore, the adversary will only be interested in a single source/destination pair of locations on each instance of the attack, and can therefore take more time to derive the largest set possible of spoofed and escape paths. The shortest path $\mathcal{P}_I$ is also bounded by a rectangle (with added padding of $m = 1000$ meters) such that all edges outside the rectangle become out of scope. Note that the above algorithm parameters are tunable and set to conservative values in this work. We believe that the attack performance can substantially improve when these parameters are tuned more aggressively, e.g., setting $F = 1.5$ and $N = 1000$ (large values of N are very reasonable when focusing on a single source/destination).

3) Escape Paths Generator: The idea behind the escape paths generator is to find all the paths an attacker can travel to reach different destinations without raising alarms. To avoid detection, all computed paths must have similar accelerometer and gyroscope patterns to spoofed paths. *The escape paths corresponding to a spoofed path $\mathcal{S}_i$ are a set of M routes $\mathcal{E}_i = \{\mathcal{E}_{i_1}, \ldots, \mathcal{E}_{i_M}\}$ such that $\mathcal{E}_{i_j} \neq \mathcal{S}_i$, but semantically similar to $\mathcal{S}_i$, for any $\mathcal{E}_{i_j} \in \mathcal{E}_i$. The paths are semantically similar when they have similar distances, road curvature and turn angles. These paths start at the input source, but end at different destinations from the intended destination.*

Input: $G = (V, E)$, $\mathcal{S}_I$
Output: N_P, $\mathcal{E} = \{p_1, \ldots, p_{N_P}\}$

1 $Initialization : \mathcal{E} \leftarrow \emptyset$; $N_P \leftarrow 0$; $p \leftarrow [\,]$; $v \leftarrow \emptyset$
2 $s \leftarrow getSourceVertex(\mathcal{S}_I)$
3 $t \leftarrow getTurnsCount(\mathcal{S}_I)$
4 `GenerateEscapePaths(s, t)`
5 **function** `GenerateEscapePaths(s, t)`**:**
6 $p \leftarrow p + [s]$
7 $v \leftarrow v \cup \{s\}$
8 **if** $len(p.turns) > t$ **then**
9 **return**
10 **if** $len(p.turns) = t$ **then**
11 $\mathcal{E} \leftarrow \mathcal{E} \cup \{p\}$
12 $N_P \leftarrow N_P + 1$
13 **for** $e \in V$ *such that* $(s, e) \in E$ **do**
14 **if** $e \notin v$ **and** $Filter(s, e, p, \mathcal{S}_I)$ *passed* **then**
15 $p.curve \leftarrow updateCurvature(s, e, p)$
16 $p.turns \leftarrow updateTurns(s, e, p)$
17 $p.score \leftarrow p.score * Score(s, e, p, \mathcal{S}_I)$
18 `GenerateEscapePaths` (c, t)
19 **end**
20 $p \leftarrow p - [s]$
21 $v \leftarrow v - \{s\}$

Algorithm 2: Escape Paths Algorithm

Given a spoofed path, the escape paths algorithm (Algorithm 2) generates a set of escape paths with similar distances, road curvatures and turn angles to the spoofed path. The algorithm is similar to that of the spoofed paths generator. The main differences being that the algorithm uses each spoofed path $\mathcal{S}_I$ generated in the previous stage as input, where $\mathcal{S}_I \in \mathcal{S}$, and outputs a set of escape paths $\mathcal{E}$. Also, the escape paths generator algorithm uses the turn count in the spoofed path as a parameter to `GenerateEscapePaths` (lines 3 – 4) and checks whether the desired turn count has been reached for the escape path under consideration (lines 10 – 12).

The deviations from the spoofed paths (to avoid INS detection) can be determined by analyzing the noise sensitivity of the inertial sensors used for tracking. We demonstrate that commodity accelerometers and gyroscopes present challenges in accurately calculating the distances, road curvature and turn angles which can allow an attacker to travel to multiple destinations without detection. We show that magnetometers are easily spoofed rendering them incapable of detecting anomalies in the heading direction of the vehicle. Our analysis of the accelerometer and gyroscope noise and the potential of magnetometer spoofing are reported in Section IV-A. Unlike the spoofed paths generator algorithm that ranked paths by score, the escape paths computed by this algorithm always have a score of 1. The intuition is that all paths that pass the algorithm's filters are certain to avoid detection by INS tracking systems.

Filtering: In this algorithm, we represent the input spoofed path by $\mathcal{S}_I = \{(d_I, \vartheta_I, \theta_I)\}$ where d_I and ϑ_I denote the set of distances and road curvatures between intersections and θ_I denotes the turn angles at the intersections. We first present the idea of filtering using just turn angles θ_I, and later expand the discussion to include distances d_I and road curvatures ϑ_I. Let $\theta_I = \{\theta(\chi_1), \ldots, \theta(\chi_K)\}$ be the derived turn angles of the spoofed path, where K is the number of intersections. A turning connection $\chi' = (s, e)$ in the escape path, where $(s, e) \in E$, is valid for an intersection $k \in K$ when the turn angle difference is below a set threshold value $\mathcal{T}_\theta$, i.e., $|\theta(\chi_k) - \theta(\chi')| \leq \mathcal{T}_\theta$. The parameter $\mathcal{T}_\theta$ depends on the noise sensitivity of the gyroscope sensor.

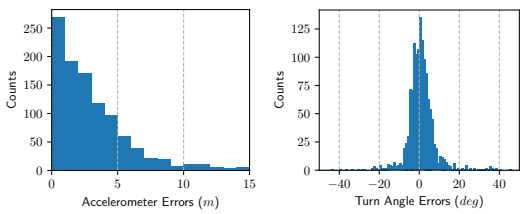
The filter for distances d_I is similar to turn angles. Let $d_I = \{d_1, \ldots, d_{K+1}\}$ be the derived distances of the spoofed path traveled between K intersections. For an intersection $k \in K$, d_k represents the path's distance from the previous intersection $k-1$, i.e., $d_k = d(k) - d(k-1)$ where $d(.)$ denotes the total distance of the spoofed path at a given intersection. Note that $k = 0$ is the source of the path and $k = K+1$ is the destination of the path. A connection χ' in the escape path is valid for intersection k when its path distance from previous intersection $k-1$ is between a range defined by the k^{th} intersection of the spoofed path, i.e., $d_k * T_{d1} \leq d'(k) - d'(k-1) \leq d_k * T_{d2}$. Here, $d'(.)$ denotes the distance of the escape path at an intersection. The above parameters T_{d1} and T_{d2} depend on the noise sensitivity of the accelerometer sensor.

The filter for road curvature ϑ_I is more complex than turn angles and distances. The reason is that, given an intersection $k \in K$, the distance d_k and turn angle $\theta(\chi_k)$ are scalars while $\vartheta(s_k)$ is a vector that must be derived from bearings of the road segment s_k between intersections $k-1$ and k. Two different vectors of bearings $\mathcal{B}_k$ and $\mathcal{B}'$ for road segments s_k and s', respectively, cannot be compared directly as they may be of different lengths and in different orientations, e.g., $\mathcal{B}_k$ may be directed north when $\mathcal{B}'$ is directed east. Our idea of calculating the road curvature similarity, denoted by $\mathcal{C}(s_k, s')$, is to translate these bearings to the same size N using linear interpolation, convert the interpolated bearings to curvature, and then compare the curvatures. Let $\mathcal{B}_{Ik}$ and $\mathcal{B}'_I$ represent the interpolated bearings for $\mathcal{B}_k$ and $\mathcal{B}'$, respectively. The curvature of a road segment s with M bearings $\mathcal{B} = [b_1, \ldots, b_M]$ can be derived by subtracting successive bearings for all the bearings in $\mathcal{B}$, i.e., $\vartheta(s) = [(b_2 - b_1), \ldots, (b_M - b_{M-1})]$. Let $\vartheta(s_k)$ and $\vartheta(s')$ be the curvatures derived from $\mathcal{B}_{Ik}$ and $\mathcal{B}'_I$, respectively. The curvature similarity of the two segments can then be represented as:

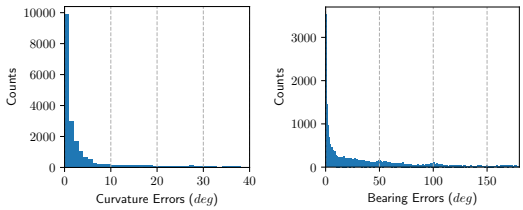
$$\mathcal{C}(s_k, s') = \{|c_k - c'| \quad \forall c_k \in \vartheta(s_k), \forall c' \in \vartheta(s')\}. \tag{3}$$

A connection χ' in the escape path is valid for intersection k when the maximum curvature similarity value is below a set threshold value $\mathcal{T}_\vartheta$, i.e., $max(\mathcal{C}(s_k, s')) \leq \mathcal{T}_\vartheta$. Like turn filtering, this parameter $\mathcal{T}_\vartheta$ also depends on the gyroscope noise sensitivity.

To avoid detection, the above discussed constraints must hold for all K intersections of the escape path. Therefore, an



(a) Distance errors (b) Turn angle errors



(c) Curvature errors (d) Bearing errors

Fig. 5: Sensor error distributions measured in real experiments.

escape path is considered valid if and only if all the following conditions are met.

$$
\begin{aligned}
&|\theta(\chi_k) - \theta(\chi')| \leq \mathcal{T}_\theta, && \forall\, k = 1, \ldots, K \\
&d_k * T_{d1} \leq d'(k) - d'(k-1) \leq d_k * T_{d2}, && \forall\, k = 1, \ldots, K+1 \\
&max(\mathcal{C}(s_k, s')) \leq \mathcal{T}_\vartheta, && \forall\, k = 1, \ldots, K+1
\end{aligned}
$$

IV. Attack Impact: Implementation and Evaluation

In this section, we present the implementation of our attack and evaluate its effectiveness in various cities across the globe. First, we evaluate the accuracy of inertial sensors and derive realistic noise threshold settings for ESCAPE algorithm. Then, we describe the details of our experimental setup and the methodology. Finally, we present the results of our evaluation using two metrics, (i) displacement from the assigned destination and (ii) coverage area of the escape paths.

A. Accuracy of Inertial Sensors

The sensor data for evaluating the noise sensitivity of accelerometers and gyroscopes was obtained from an open dataset [47]. This dataset comprises of accelerometer, gyroscope and magnetometer samples recorded from ≈ 140 real driving experiments in the cities of Boston and Waltham, MA, USA. The sensor samples were collected on 4 smart phones (HTC One M7, LG Nexus 5, LG Nexus 5X, and Samsung S6). The GPS traces for these routes were also recorded for ground truth comparison. The work focused specifically on gyroscope noise during turns. We extend that work to also include noise sensitivity when distance is calculated from the accelerometer sensor, as well as when road curvature is calculated from the gyroscope sensor.

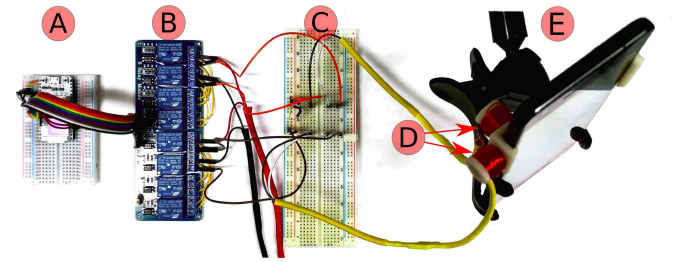
1) Accelerometer Accuracy: The accelerometer sensor can be used to calculate the distance traveled for a path. This data can be represented as a vector $a = [(a_1 + n_1), \ldots, (a_T + n_T)]$ sampled at discrete time intervals $t \in T$, where a_t is the true acceleration experienced by the device on the x, y and z axis, and n_t is an unknown noise quantity caused by several factors. For example, the sensors have an inherent bias due to manufacturing defects such as axis misalignment. Another source of noise is the vibrations caused by the mechanical structure of the vehicle and the engine. Additional noise is induced on the sensor due to external environments such as road conditions and traffic.

We are interested in finding the range of divergence from the actual values due to n_t, when distance is calculated from accelerometer data. To obtain this range, we calculate the distances between intersections using accelerometer data for each sensor path in the data-set, and compare it to the actual distances obtained from OpenStreetMap. To reduce the impact of noise, we perform the calibration and rotation techniques described in [47] before calculation. We also average multiple samples together to further reduce the impact from noise. As distances may significantly vary between intersections, we represent the distance error as a ratio of the derived accelerometer distances to the actual distances. More precisely, if d_s is a vector of N derived accelerometer distances and d_a is a vector of N actual distances, then errors e_a can be represented as a vector $e_a = [(d_{s_1}/d_{a_1}), \ldots, (d_{s_N}/d_{a_N})]$. Figure 5a shows the distribution of the errors e_a. Note that the desired value for an error should be near 1, however, we see large variations ranging between 0.1 to 5. This indicates that the accelerometer sensor is unsuitable for distance calculation and enables an attacker to travel much larger distances than the intended path. Recall that the escape paths generator algorithm uses parameters T_{d1} and T_{d2} to filter connections of the escape paths based on distances (Section III-C3). These parameters are chosen from the error distribution e_a such that the allowed range is based on the 75^{th} percentile of the distribution, i.e., $T_{d1} = 0.2$ and $T_{d2} = 3.3$.

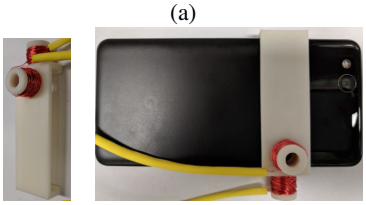
2) Gyroscope and Magnetometer Accuracy: The gyroscope sensor can be used to measure the turn angles and the curvature of the path. This data can also be represented as the vector $g = [(g_1 + n_1), \ldots, (g_T + n_T)]$, where g_t is the rate of angular change experienced by the device on the x, y and z axis, and n_t is an unknown noise quantity. In this case, however, the impact of n_t is not as significant as accelerometers and the measurements are closer to the actual values.

We are interested in finding the turn angle errors and the curvature errors calculated from the gyroscope data, in comparison to the actual values derived from OpenStreetMap. To calculate the turn errors, we use a similar approach to [47] in that we define a turn error as the absolute difference between the gyroscope derived turn angle and the actual turn angle. However, we are interested in the overall error distribution for all the phones instead of individual phones. Figure 5b shows the distribution of the turn angle errors for all the turns in the data-set. The distribution reaffirms that the gyroscope is much more accurate than the accelerometer where 75% of the turn errors are within 5.5°.

To calculate the curvature errors, recall our technique for calculating curve similarity $\mathcal{C}(s_k, s')$ for two road segments

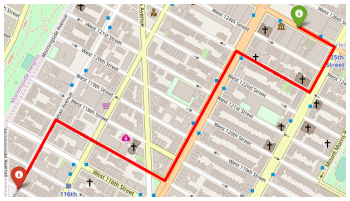


(a)

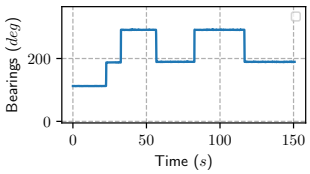


(b)

Fig. 6: a) Experimental setup used for magnetometer spoofing. b) The two-coil system attached to a Google Pixel 2



(a) Example Route in Manhattan



(b) Spoofed Bearings

Fig. 7: Example of magnetometer spoofing.

s_k and s' between the $(k-1)^{th}$ and k^{th} intersections (Equation (3)). The road curvature $\vartheta(s_k)$ is already known in the form of the gyroscope data. However, this curvature must be interpolated to the same length as $\vartheta(s')$. Given the union of curve similarity sets for all K intersections for N sensor paths $\mathcal{C} = \bigcup_{i=1}^{N} \mathcal{C}_i$, where $\mathcal{C}_i = \bigcup_{j=1}^{K} \mathcal{C}(s_j, s'_j)$, the curvature errors e_c is simply a set of absolute differences between all the points in the two curves, i.e., $e_c = \{|c_s - c_a| \quad \forall [c_s, c_a] \in \mathcal{C}\}$. Figure 5c shows the distribution of the curve errors. Recall that the escape paths generator algorithm defines parameters T_θ and T_ϑ to filter connections based on turn angles and curvature, respectively (Section III-C3). Based on the 75^{th} percentile of the error distributions, we set the parameters to $T_\theta = 5.5°$ and $T_\vartheta = 2.8°$ in our evaluations.

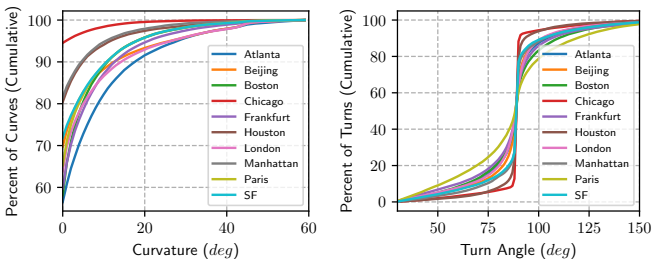
We compute the bearing errors using the same technique as curvature. Figure 5d shows the distribution of the bearing errors. Note that the errors are much larger than the Gyroscope owing to nearby magnetic fields from fans, speakers and other electromagnetic devices. These errors are very difficult to reduce and requires performing regular hard-iron calibration of the device inside the vehicle.

3) Magnetometer Spoofing: As a proof of concept, we built a prototype of a magnetometer spoofer for the Google Pixel 2 smart phone. Our experimental setup is shown is Figure 6a and consists of the following modules: (A) an ESP32 microcontroller, (B) a 8-channel relay module, (C) resistors for controlling current flow, (D) a two coils system, and (E) a Google Pixel 2 mounted on a car mount. We first identified the exact location of the magnetometer which is on the top-left of the phone ($42mm$ from the top and $7mm$ from left edge of the phone). We designed and 3D printed a two-coils system, shown in Figure 6b, that snaps on to the phone and allows the wrapping of enameled magnet wire. We focused on controlling the x and y axes as they are easily reachable. Using two coils each targeting one of the axes allows full control of the magnetic field in a plane. We used the following solenoid magnetic field formula to estimate the intensity: $B = k\mu_0 nI$ where k is the relative permeability, $\mu_0 = 4\pi 10^{-7}$ H/m, n is the coil turn density, and I is the electric current. Our coils turn density n is 155 turns/meter since we used 5 layers of 28 AWG enameled magnet wire. Without a core ($k = 1$), we estimated a magnetic field of $98uT$ with a current of $5mA$, which is strong enough to impact the magnetometer. Note that if the magnetometer is not accessible in other systems, it is possible to use larger coils or channel the magnetic field using materials with higher relative permeability. While the relative permeability of air is 1, it is $5,000$ for iron, and $200,000$ for iron annealed in hydrogen. To control the current in each of the coils, we used the ESP32 microcontroller (Heltec WiFi Kit 32) with a sufficient number of GPIO/DAC pins to control the 8-channel relay module augmented with variable resistors for current tuning. The spoofer was written in Python and takes as input a sequence of bearings and durations. It sets the current in the coils to trigger turns with a timing that matches the input durations. Figure 7 shows an example spoofing route.

B. Integrated Attack

An integrated GPS/INS/Magnetometer spoofing system is not trivial. It has to address two categories of challenges: (1) synchronizing all spoofing components: GPS spoofing SDR platform, magnetometer spoofing coils, and sensing components real GPS, and INS sensors; and (2) dynamically generating the GPS spoofing RF signals consistent with the real time vehicle trajectory. Handling the synchronization is a matter of carefully architecting the system and implementing the control loop. However, real time spoofing of the GPS and magnetometer to account for traffic conditions such as traffic lights and fluidity is more challenging. While the magnetometer is fully under the control of the adversary and it is therefore a matter of synchronizing the control of the two coils with the actual turns, the real time spoofing of the GPS is more challenging. As noted in the related work, there are various systems that generate *offline* GPS RF signals for a pre-determined trajectory. It is challenging when the trajectory changes in real time as it is critical that the GPS receiver does not lose the lock on the satellites, which would happen if a new GPS position is abruptly spoofed.

We extended an existing open source tool [48] that spoofs GPS RF signals to dynamically interpolate a desired trajectory in real time, independently adjusting each of the spoofed



(a) Curvature Distributions (b) Turn Distributions

Fig. 8: Comparison of the curvature and turn distribution.

satellite signals at a fine grain, therefore enabling the GPS receiver to maintain its lock and enabling a tight control loop of the spoofed GPS signal. In order to account for traffic fluidity, congestion, and lights, we spoof being at a position that is proportionally equivalent on the physical road section. Therefore, the spoofed position remains constant when the vehicle is not moving and this position changes with a velocity proportional to the real velocity when the vehicle is moving. The paths are mapped such that when the driver turns on the real path, the spoofed position also turns on a valid intersection with a similar turn angle to the real turn. To the best of our knowledge, this is the first real time dynamic GPS spoofer. The GPS spoofer also integrates with other sensor spoofing peripherals, in this case the magnetometer.

C. Evaluation Methodology and Setup

We implemented the ESCAPE attack algorithms in PyPy. We used two servers running Intel Xeon CPUs at $2.40GHz$ with 12 cores and $20GB$ of RAM to execute the algorithms and evaluate its performance i.e., how far can an attacker escape, given a start and end point, without being detected.

Selection of cities: We evaluate the effectiveness of our attack on the road networks of 10 major cities across the globe. The following cities were chosen across the continents of North America, Europe and Asia for the evaluation. The cities were chosen to represent the entire spectrum of urban characteristics such as major logistics and transportations hubs, dense population, city planning (e.g., grid-like or circular), etc. Figure 8 shows the cumulative road curvature and turn distributions for all selected cities. Recall that the road curvatures are calculated using Equation (1). We can observe that Chicago and Manhattan have mostly straight roads and right angled turns while the road networks of London and Paris have very unique characteristics.

Generation of spoofed and escape routes: The evaluation was performed by running simulations for every selected city. This simulation data comprised of 1000 randomly generated paths in every city, such that the path distances were uniformly distributed between $1km$ and $21kms$. The intention was to evaluate the potential of spoofing also as a function of the path distance. The simulation paths were generated as follows: (i) a random 'Home' and 'Work' location were chosen from OpenStreetMap tags inside the interest area, (ii) the geographic coordinates of the end points were retrieved, and (iii) the coordinates were given as input to the attack algorithms to compute the spoofed and escape paths.

Integrated system: We built an attack evaluation system that includes a GPS receiver (ublox NEO-M8N) to obtain the real location of the vehicle in real time, a bladeRF x40 SDR to generate the spoofed GPS signal, our magnetometer spoofer (described earlier), all connected to a Dell XPS laptop running the control software. On the target side, we use a Google Pixel 2 target (fitted with the magnetometer spoofer). In order to avoid transmitting the spoofed GPS signal over the air (and potentially interfering with other neighboring devices), we use a second external GPS receiver with an external antenna connected to the bladeRF using an RF coax cable. The second GPS receiver is connected to the Pixel phone through a USB link. The control software integrating all the components is written in Python and leverages existing libraries (e.g., geodesic calculations, nearest node search).

As for the attack setup, we create a mapping between the escape and spoof paths. The mapping simplifies the spoofing real time execution, as each attacker's real location corresponds to a spoofed location. More specifically, we (1) split both paths into segments such that a segment is a part of road between two turns, (2) interpolate coordinates of the escape path such that each coordinate is at exactly 1 meters from previous coordinate, and (3) interpolate coordinates of spoof path using escape paths segments, i.e., this interpolation has exactly the same number of points and the distance is based on its ratio with the corresponding escape segment.

Executing the Integrated Attack: Three key blocks (Figure 9) operate in parallel handling different functions (threads). (1) The first block gets the real GPS location of the attacker using a ublox NEO-M8N operating at 10 Hz. It uses the GPS location to compute the nearest escape location on the map, and the corresponding mapped spoof location on the map. The spoof bearing can be estimated based on the spoof location. The efficiency of calculating the spoof location is important to avoid lags between sensor turns and the spoof GPS turns. To compute the nearest escape location efficiently, we use a 2D-Tree (a two dimensional representation of a KD-Tree) which has an average time complexity of $O(\log n)$ with a worst-case time complexity of $O(n)$, where n is the count of total coordinates in the escape path. In our experiments, using a Dell XPS laptop (i7 Quad-Core processor with 16GB RAM), the average time for computing the spoofed location is $\approx 4ms$ with worst-case of $\approx 60ms$. This is negligible delay for human perception. (2) The second block updates the spoofed location to the bladeRF x40 Software Defined Radio (SDR) every $10ms$. To avoid transmitting and disturbing the GPS signals of neighboring vehicles, the Tx of the SDR was connected to another ublox NEO-M8N and acted as an external antenna for the GPS receiver. The device under test (DUT) obtained signals from this GPS receiver using USB OTG. (3) The third block updates the spoofed bearing to the magnetometer.

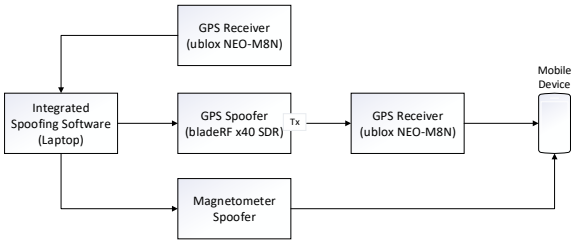
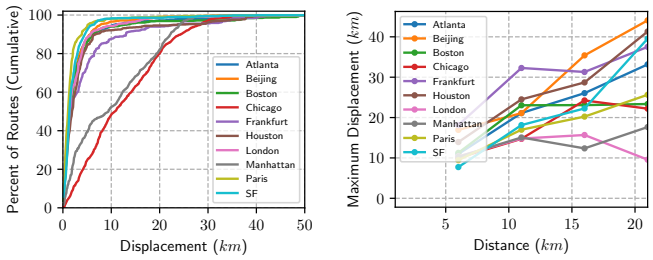


Fig. 9: Block diagram of the integrated spoofing system.



(a) (b)

Fig. 10: a) Attacker's displacement from assigned destination for the generated paths. b) The maximum displacement in every city for specific path lengths

D. Evaluation Results

We measure the performance of our attack using the two metrics: (i) displacement and (ii) coverage area.

Displacement from Intended Destination: We define displacement from the intended destination as the farthest distance an attacker can reach for a chosen trip (i.e., given a start and end point) without being detected. For every evaluation route, escape and spoofed paths are generated as described previously. We then calculate the euclidean distance between the destinations an attacker reaches by taking the escape route and the actual intended destination i.e., the assigned end point for the trip. We present our results in Figure 10. Figure 10a shows the attacker's deviation from the intended or assigned destination for the generated routes in all 10 cities. It can be observed that in majority of the cities, more than 20% of the routes allow more than 10 km deviation from the intended destination. There are at least 10% of the routes in all selected cities where the attacker is able to reach points as far as $30\,\mathrm{km}$ away from the assigned destination. Chicago and Manhattan perform the worst among the selected cities with more than 40% of the routes allowing a displacement of $15\,\mathrm{km}$ or above. This is due to the regular patterns that exist in these cities' road network. Figure 10b shows the maximum displacement in each city for specific assigned route lengths. It is important to observe that in Manhattan and Chicago the maximum displacement caused is independent of the assigned route distance. This is due to the structure of the cities itself. For example, Manhattan is a narrow strip with grid like structures and therefore maximum displacement saturates at some point. However, for a city like Beijing there are routes that allow an attacker to spoof his location to as far as $40\,\mathrm{km}$ away from the intended location.

Coverage Area of Spoofed Paths: The goal of this evaluation is to determine the percentage of area an attacker can cover by traveling the escape paths generated for a given source $Loc(s)$ and destination $Loc(d)$ geographic coordinates. Let A denote the total geographic area of interest to an attacker. For this evaluation, we define this area as a circle of radius $r = d(Loc(s), Loc(d))$ with center at $Loc(s)$ where r is the euclidean distance between the source and destination. The above area may comprise of water bodies which must be accounted for more accurate coverage. Let A_L denote the area of land within the interest area. Within A_L, let A_C denote the area that the attacker can cover if he is willing to walk a small distance r' from an escape destination. $(A_C/A_L) * 100$ gives the escape path's coverage area percentage. The area A_L is not trivial to calculate as the location of water bodies are not pre known within the interest area. The area A_C is also not trivial to calculate as the escape destinations may be densely populated and many may overlap. To solve this, we implemented Monte-Carlo simulations. The simulation works by generating millions of uniformly distributed points within the interest area. It maintains two separate counters: P_L to count all the points that are on land (i.e., within r' meters of any road), and P_C to count all points within an escape destination's radius (i.e., within r' meters of any escape destination). With these counters, the area A_L can be calculated as $A_L = (P_L/P) * A$, where P is the total number of points, and the area A_C can be calculated as $A_C = (P_C/P) * A$. Therefore, the final coverage area percentage of the escape paths using Monte-Carlo simulation can be expressed as $(P_C/P_L) * 100$. The coverage percent is the ratio of the coverage area calculated (using a 100 m walking radius) to the total area of land calculated using the Monte-Carlo simulation.

The results are shown in Figure 11. It can be observed that cities with more regular grid-like patterns such as Chicago and Manhattan, New York City are more vulnerable to attacks. In these cities, an attacker can, on average, cover more then 30% of the target land area without being detected. For many routes, they can even cover more than 60% of the target land area. However, more irregular cities like London, Frankfurt and Atlanta offer more resistance. It is important to note that it is still possible to reach 20% of the target geographic region even in these most limiting cases. The percent of coverage reduces as route or trip distances increases because as trip length increases so does the probability of the presence of an unique road segment, but also because the area of interest grows quadratically in the distance between source and destination. For instance, for a distance of $20km$, the area of interest is $400km^2$ and the coverage is $40km^2$ which is still significant. Also, note that the above calculations present a lower-bound on the total coverage area A_C. This is because errors in distance calculation from the accelerometer allows the attacker to cover much larger distances. For example, in a number of escape

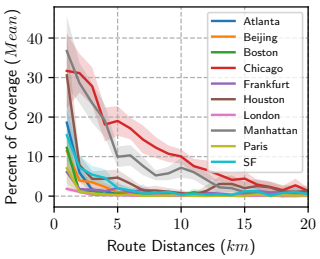
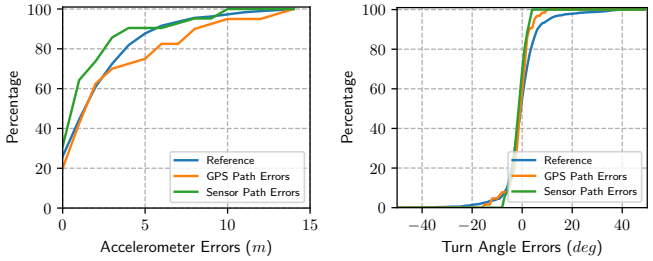


Fig. 11: Attacker's mean coverage area with 95% confidence interval for different route distances.

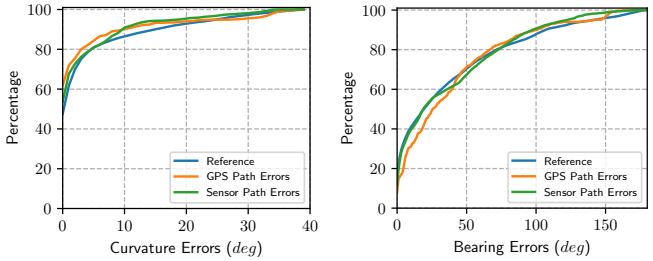
routes computed in our evaluation, up to 82% of final escape destinations were located even beyond the area of interest used for evaluation, with a mean of $\approx 46\%$.

Evaluation of integrated attack: We evaluated the performance of the integrated spoofer on 10 routes. The spoofed routes were chosen such that either the source or destination was close to the university area and the other end-point was within a $6km$ radius of the university. For each of the spoofed path, the system selected a corresponding escape path with the maximum displacement. In our set, we found this displacement to range between $.7km$ to $2.1km$. During the experiment, we drove the escape paths while spoofing the GPS and magnetometer with the spoofed path. The following information were recorded: the accelerometer, gyroscope and magnetometer sensors and the GPS locations. We also drove the spoofed paths to obtain baseline recordings. The attack was successful for all the chosen paths despite traffic stops, lights, or congestion. The spoofed locations (displayed on the phone) matched the chosen locations and the turns were synchronized between the escape and spoofed paths. We have uploaded a short video[2] demonstrating the attack. The maximum delay between receiving current escape path location and computing the spoofed location was $\approx 60ms$ with most spoofed locations computed within $4-5ms$ of obtaining the current location. For quantitative comparison, we also drove the spoofed path and recorded all the sensor measurements and compared it with the escape path measurements. The results are shown in Figure 12. The path errors are computed as the difference in the sensor measurements between the spoofed route and the escape route. The 'GPS path errors' are based on the calculations performed using GPS data and the 'sensor path errors' are the estimates using the inertial sensors. We then compare it with the open 'reference' dataset [47] recorded from 140 different routes. We observe that the error distributions are similar to the reference dataset. For example, the gyroscope errors (both curvature and turns) are within the range reported by the reference dataset with $\approx 85-90\%$ ($\approx 90-95\%$ for sensor baseline) of turns and $\approx 75-80\%$ ($\approx 70-75\%$ for sensor baseline) of curves within the ranges defined in our algorithm. We also observed that the accelerometer and magnetometer data are more sensitive to noise (e.g., magnets in car's dashboard) than the gyroscope.

[2]Video of our experiment – https://youtu.be/Tvj8Fv5jFLw



(a) Distance calculation errors (b) Turn angle errors



(c) Curvature errors (d) Bearing errors

Fig. 12: Sensor error distributions for the GPS-based estimates and sensor-based measurements in comparison with the reference open dataset's distribution.

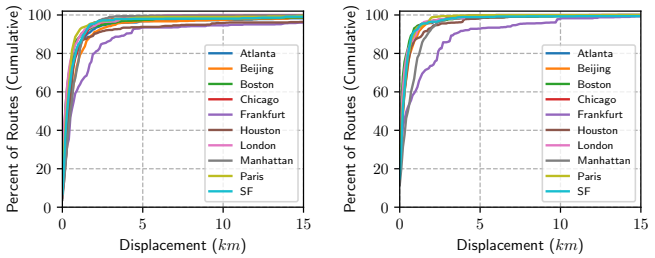
V. Countermeasures

A. Deploying Accurate Accelerometer and Gyroscope Sensors

An obvious approach to mitigating the threat would be to use high quality sensors. To measure the impact of sensor noise on the potential of spoofing, we re-ran the simulations on the cities using lower thresholds for the sensor noise. For this evaluation, we set the thresholds using the 25^{th} percentile of the error distributions (c.f., Figure 5). The following thresholds were set for the escape paths generator algorithm: $T_\theta = 1.4°$, $T_\vartheta = 0.2°$, $T_{d1} = 0.6$ and $T_{d2} = 1.6$. Figure 13a shows the results of the simulations for all the cities. Using the above thresholds, we see a significant reduction in the percentage of routes that allow more than $5\,\mathrm{km}$ of displacement. However, there are several limitations with this approach. First, the sensors satisfying the above parameters are equivalent to aviation and military-grade sensors which are bulky and expensive (several thousands of dollars) to deploy. Furthermore, they consume significant amount of power ($\gtrsim 5\,\mathrm{watts}$) making it unsuitable for use in majority of tracking applications. Moreover, the attacker can still induce noise in the sensors by driving recklessly (e.g., rapid accelerations, lane switching).

B. Secure Navigation Path Selection

We present a path selection algorithm that provides better mitigation than deploying accurate sensors, without requiring any changes to existing GPS/INS tracking systems. The idea is to generate a single "secure path" for travel. This path is less favorable for spoofing because the curvatures or turn angles of the path are more unique and, therefore, less likely to be found in other sections of the road network. Furthermore, even



(a) Military-grade sensors (b) Secure Path Selection

Fig. 13: Displacement error comparison between using military grade sensors and our secure path selection algorithm.

if there exists some potential for spoofing in this path, the escape routes can be known well in advance and appropriate countermeasures can be taken to prevent it. Recall that the attack algorithm searches for navigation routes with a high probability of occurrence in other sections of the road network (c.f., Section III-C2). The final path score was calculated using Equation (2). For generating secure paths that are more resilient to spoofing, the algorithm simply negates this path score, i.e., $score = -\prod_{i=1}^{M} P(\vartheta(s_i), \theta(\chi_i))$. This has the effect of assigning the highest score to paths containing unique road curvature and turn angles with low probability of occurrence. If we choose the count of output paths N_P as 1, in some instances, a path with unique curvature and turns only in the beginning or towards the end of the path can get chosen. This could allow the attacker to achieve higher displacement than other potential paths, which is undesirable. To mitigate this, the algorithm sets N_P as 100 and chooses the path that outputs the least number of spoofable paths. In other words, the application or service provider (e.g., logistics company) can assign "secure navigation routes" that are hard to fake because of unique road characteristics. Figure 13b shows the results of evaluations for all cities using the same parameters as the original simulations, albeit the scoring method. Comparing with the original simulations, we see the attacker significantly limited in the amount of alternate routes available to him.

VI. Related Work

In 2001, the Volpe report [49] first identified malicious interference with the civilian GPS signal as a serious problem. Following this several researchers have demonstrated the insecurity of GPS-based navigation by diverting the course of a yacht [11], forcing drones [12] to land in a hostile area and taking over navigation systems of transportation trucks [24] using spoofed GPS signals. In addition to commercial GPS simulators [50], [51], it is today possible to build low-cost GPS signal spoofers that generate GPS signals for any chosen trajectory or navigation route using existing public repositories [44] and less than $300 [43] of hardware equipment. Advanced attacks [52], [53] in which the attackers *takeover* a target receiver that is already receiving navigation messages from authentic satellite signals without the receiver noticing any disruption or loss of navigation data. It was also shown that a variety of commercial GPS receivers were vulnerable and in some cases even caused permanent damage to the receivers. Zeng et al. [54] explored the feasibility of stealthily spoofing GPS-based road navigation systems. The attacker generates fake GPS signals that closely resembles the shape of a route shown on the navigation software. The goal of the attack is to fooling the user to drive a route that *looks* similar to the original path. The success of the attack depends on the user's ability to match the navigation instructions with their surroundings (e.g., street names). The attack further exploits the limited area of focus that first-person views of majority of navigation system provide. The ESCAPE attack proposed in this paper makes no such assumption and succeeds against an autonomous system that keeps track of a vehicle's movement patterns using both GPS and inertial sensors. In fact, Zeng et al.'s [54] propose the use of inertial navigation sensors as a countermeasure against their attack. In general, several countermeasures were proposed against GPS spoofing attacks that are both cryptographic [16], [17], [18], [19] and non-cryptographic [20], [21], [22], [23], [24], [25], [26] to detect or mitigate GPS signal spoofing attacks. These techniques are either unreliable (e.g., large number of false alarms), effective only against naive attackers or required modifications to the GPS receiver/infrastructure itself. Alternate localization technologies using WiFi or cellular networks [27], [28] lack the accuracy and coverage required for the above mentioned applications and are vulnerable to jamming attacks [55], [56], [57], [58], [59]. In the context of on-road navigation and tracking, using data from inertial sensors [29], [30], [31] alongside GPS is emerging as a popular choice for applications where spoofing and jamming are a threat. The absence of any communication between the inertial sensors and the external world for estimating the location makes it robust to signal spoofing and jamming attacks. Many works [36], [37], [38], [39], [40], [41], [60] analyze and show that inertial sensors are promising for detection and mitigation of GPS spoofing attacks. Many commercial-off-the-shelf GPS/INS products [32], [33], [34], [35] are available and used in many civilian and military applications. Recently, analog attacks have also been demonstrated on inertial sensors. WALNUT [61] shows how analog acoustic injection attacks can affect the digital integrity of a capacitive MEMS accelerometer. Son et al. [62] showed that acoustic interference on MEMS gyroscopes in drones can cause them to crash. Shoukry et al. [63] demonstrate how to deliver fake readings to an anti-lock braking system via the magnetic wheel speed sensors using electro magnetic interference in an automotive setting. In this paper, we show that magnetometers are vulnerable to electromagnetic interference attacks and an attacker can precisely control its output. Given the emergence of GPS/INS solutions, we believe our work emphasizes fundamental security limitations of GPS/INS for road navigation and tracking applications.

Acknowledgments

The work was partially supported by NSF grants 1740907, 1643249, and 1850264.

REFERENCES

[1] J. R. Coffee, R. W. Rudow, R. F. Allen, M. Billings, D. A. Dye, M. L. Kirchner, R. W. Lewis, K. M. Marvin, R. D. Sleeper, W. A. Tekniepe *et al.*, "Vehicle tracking, communication and fleet management system," Aug. 26 2003, US Patent 6,611,755.

[2] Y. A. Novik, "System and method for fleet tracking," Jan. 15 2002, US Patent 6,339,745.

[3] "Verizon Connect Fleet Management System," https://www.verizonconnect.com/solutions/gps-fleet-tracking-software/.

[4] "Massachusetts Probation Service's Electronic monitoring program ," https://www.mass.gov/service-details/electronic-monitoring-program.

[5] "Geo-Satis Electronic Monitoring Solution," https://geo-satis.com/.

[6] "US Department of Transportation: In-vehicle Performance Monitoring and Feedback," https://www.transportation.gov/mission/health/In-vehicle-Performance-Monitoring-and-Feedback.

[7] G. Mintsis, S. Basbas, P. Papaioannou, C. Taxiltaris, and I. Tziavos, "Applications of gps technology in the land transportation system," *European journal of operational Research*, 2004.

[8] "Developing GPS monitoring for the public transport fleet," http://civitas.eu/measure/developing-gps-monitoring-public-transport-fleet.

[9] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[10] G. GSA, "Market report issue 3," 2017, https://www.gsa.europa.eu/.

[11] "UT Austin Researchers Successfully Spoof an $80 million Yacht at Sea," http://news.utexas.edu/2013/07/29/ut-austin-researchers-successfully-spoof-an-80-million-yacht-at-sea.

[12] T. Humphreys, "Statement on the vulnerability of civil unmanned aerial vehicles and other systems to civil gps spoofing," *University of Texas at Austin (July 18, 2012)*, 2012.

[13] K. C. Zeng, Y. Shu, S. Liu, Y. Dou, and Y. Yang, "A practical gps location spoofing attack in road navigation scenario," in *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*. ACM, 2017.

[14] "Jamming GPS Signals Is Illegal, Dangerous, Cheap, and Easy," https://gizmodo.com/jamming-gps-signals-is-illegal-dangerous-cheap-and-e-1796778955.

[15] "N.J. Man In A Jam, After Illegal GPS Device Interferes With Newark Liberty Operations," https://newyork.cbslocal.com/2013/08/09/n-j-man-in-a-jam-after-illegal-gps-device-interferes-with-newark-liberty-operations/.

[16] T. E. Humphreys, "Detection strategy for cryptographic GNSS anti-spoofing," *IEEE Transactions on Aerospace and Electronic Systems*, 2013.

[17] M. G. Kuhn, "An asymmetric security mechanism for navigation signals," in *Information Hiding*, 2005.

[18] S. C. Lo and P. K. Enge, "Authenticating aviation augmentation system broadcasts," 2010.

[19] K. Wesson, M. Rothlisberger, and T. Humphreys, "Practical cryptographic civil GPS signal authentication," *Journal of Navigation*, 2012.

[20] D. M. Akos, "Who's afraid of the spoofer? GPS/GNSS spoofing detection via automatic gain control (AGC)," *Navigation*, 2012.

[21] A. Ranganathan, H. Ólafsdóttir, and S. Capkun, "Spree: A spoofing resistant gps receiver," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 2016.

[22] M. L. Psiaki, S. P. Powell, and B. W. O'Hanlon, "GNSS spoofing detection using high-frequency antenna motion and carrier-phase data," in *Proceedings of the ION GNSS+ Meeting*, 2013.

[23] K. Wesson, D. Shepard, J. Bhatti, and T. E. Humphreys, "An evaluation of the vestigial signal defense for civil GPS anti-spoofing," in *Proceedings of the ION GNSS Meeting*, 2011.

[24] J. S. Warner and R. G. Johnston, "GPS spoofing countermeasures," *Homeland Security Journal*, 2003.

[25] A. Broumandan, A. Jafarnia-Jahromi, V. Dehghanian, J. Nielsen, and G. Lachapelle, "GNSS spoofing detection in handheld receivers based on signal spatial correlation," in *Proceedings of the IEEE Position Location and Navigation Symposium (PLANS)*, 2012.

[26] A. Jafarnia-Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle, "GPS vulnerability to spoofing threats and a review of antispoofing techniques," *International Journal of Navigation and Observation*, 2012.

[27] P. A. Zandbergen, "Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning," *Transactions in GIS*, 2009.

[28] N. O. Tippenhauer, K. B. Rasmussen, C. Pöpper, and S. Čapkun, "Attacks on public wlan-based positioning systems," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009.

[29] D. Titterton, J. Weston *et al.*, *Strapdown Inertial Navigation Technology. 2nd Edition*. IET, 2004.

[30] J. Farrell and M. Barth, *The Global Positioning System and inertial navigation*. McGraw-Hill New York, 1999.

[31] J. Wendel, O. Meister, C. Schlaile, and G. F. Trommer, "An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter," *Aerospace Science and Technology*, 2006.

[32] "KVH Systems - Using Inertial Systems to Overcome GPS Spoofing," https://www.kvhmobileworld.kvh.com/.

[33] "VectorNAV - Embedded Navigation Solutions," https://www.vectornav.com/products.

[34] "Honeywell Aerospace - Embedded GPS/INS," https://aerospace.honeywell.com/en/products/navigation-and-sensors/embedded-gps-or-ins.

[35] "Navtech GPS soplutions," https://www.navtechgps.com/oxts_xoem_inseries/.

[36] S. Khanafseh, N. Roshan, S. Langel, F.-C. Chan, M. Joerger, and B. Pervan, "Gps spoofing detection using raim with ins coupling," in *Proceedings of the Position, Location and Navigation Symposium—PLANS*, 2014.

[37] N. A. White, P. S. Maybeck, and S. L. DeVilbiss, "Detection of interference/jamming and spoofing in a dgps-aided inertial system," *IEEE Transactions on Aerospace and Electronic Systems*, 1998.

[38] J.-H. Lee, K.-C. Kwon, D.-S. An, and D.-S. Shim, "Gps spoofing detection using accelerometers and performance analysis with probability of detection," *International Journal of Control, Automation and Systems*, 2015.

[39] S. Dehnie and R. Ghanadan, "Methods and systems for detecting gps spoofing attacks," Dec. 30 2014, US Patent 8,922,427.

[40] R. E. Ebner and R. A. Brown, "Integrated gps/inertial navigation apparatus providing improved heading estimates," Aug. 12 1997, US Patent 5,657,025.

[41] L. M. P. A. Serrano, C. S. Dixon, and M. J. Perren, "Receiver and method for authenticating satellite signals," Jul. 28 2011, US Patent App. 12/780,337.

[42] "Ettus research llc," http://www.ettus.com/.

[43] "Hacking A Phone's GPS May Have Just Got Easier," http://www.forbes.com/sites/parmyolson/2015/08/07/gps-spoofing-hackers-defcon/.

[44] "Opensource software-defined GPS signal simulator," https://github.com/osqzss/gps-sdr-sim.

[45] G. M. Rebeiz, *RF MEMS: theory, design, and technology*. John Wiley & Sons, 2004.

[46] OpenStreetMap, "OpenStreetMap Project," https://www.openstreetmap.org/.

[47] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "Inferring user routes and locations using zero-permission mobile sensors," in *2016 IEEE Symposium on Security and Privacy (S&P)*, 2016.

[48] "GPS-SDR-SIM - Software-Defined GPS Signal Simulator," https://github.com/FrankBuss/gps-sdr-sim.

[49] J. A. Volpe, "Vulnerability assessment of the transportation infrastructure relying on the global positioning system," *http://www.navcen.uscg.gov/*, 2001.

[50] "LabSat GPS Simulator," http://www.labsat.co.uk/.

[51] "GSG-xx Series Multi-channel advanced GNSS simulator," http://www.spectracomcorp.com/.

[52] T. Nighswander, B. M. Ledvina, J. Diamond, R. Brumley, and D. Brumley, "GPS software attacks," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2012.

[53] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM Conference on Computer and communications security*, 2011.

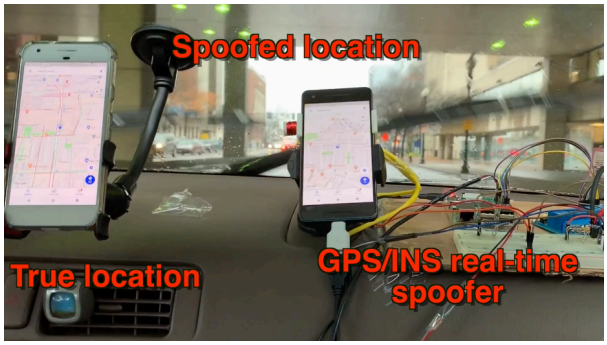
[54] K. C. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang, "All your GPS are belong to us: Towards stealthy manipulation of road navigation systems," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[55] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, "Interleaving jamming in wi-fi networks," in *Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2016.

[56] L. Xin, D. Starobinski, and G. Noubir, "Cascading denial of service attacks on wi-fi networks," in *2016 IEEE Conference on Communications and Network Security (CNS)*, 2016.
[57] K. Firouzbakht, G. Noubir, and M. Salehi, "On the performance of adaptive packetized wireless communication links under jamming," *IEEE Transactions on Wireless Communications*, 2014.
[58] ——, "On the capacity of rate-adaptive packetized wireless communication links under jamming," in *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2012.
[59] T. D. Vo-Huu, E.-O. Blass, and G. Noubir, "Counter-jamming using mixed mechanical and software interference cancellation," in *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2013.
[60] A. Mosenia, X. Dai, P. Mittal, and N. K. Jha, "Pinme: Tracking a smartphone user around the world," *IEEE Transactions on Multi-Scale Computing Systems*, 2018.
[61] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks," in *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, 2017.
[62] Y. M. Son, H. C. Shin, D. K. Kim, Y. S. Park, J. H. Noh, K. B. Choi, J. W. Choi, and Y. D. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security symposium*, 2015.
[63] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2013.

APPENDIX

A. In-car Experimental Setup



The photograph of our in-car integrated GPS/INS spoofer setup used in the evaluations. We have uploaded a video of our experiment in this link https://youtu.be/Tvj8Fv5jFLw . Note that parts of the video has been edited (fast forward, and parts cut) to focus the illustration on the relevant parts.