2016 IEEE Symposium on Security and Privacy

Inferring User Routes and Locations using Zero-Permission Mobile Sensors

Sashank Narain*, Triet D. Vo-Huu†, Kenneth Block‡ and Guevara Noubir§
College of Computer and Information Science
Northeastern University, Boston, MA, USA
Email: *sashank@ccs.neu.edu, †vohuudtr@ccs.neu.edu, ‡block.k@husky.neu.edu, §noubir@ccs.neu.edu

***Abstract*—Leakage of user location and traffic patterns is a serious security threat with significant implications on privacy as reported by recent surveys and identified by the US Congress Location Privacy Protection Act of 2014. While mobile phones can restrict the *explicit* access to location information to applications authorized by the user, they are ill-equipped to protect against side-channel attacks. In this paper, we show that a zero-permissions Android app can infer vehicular users' location and traveled routes, with high accuracy and without the users' knowledge, using gyroscope, accelerometer, and magnetometer information. We modeled this problem as a maximum likelihood route identification on a graph. The graph is generated from the OpenStreetMap publicly available database of roads. Our route identification algorithms output both a ranked list of potential routes as well a ranked list of route-clusters. Through extensive simulations over 11 cities, we show that for most cities with probability higher than 50% it is possible to output a short list of 10 routes containing the traveled route. In real driving experiments (over 980 Km) in the cities of Boston (resp. Waltham), Massachusetts, we report a probability of 30% (resp. 60%) of inferring a list of 10 routes containing the true route.**

I. Introduction

The mobile revolution has profoundly changed how we share information and access services. Despite its immense benefits, it opened the door to a variety of privacy-invasion attacks. Leakage of location information is a major concern as it enables more sophisticated threats such as tracking users, identity discovery, and identification of home and work locations. Furthermore, discovery of behaviors, habits, preferences and one's social network are at risk, and can potentially lead to effective physical and targeted social engineering.

The topic of location privacy has been extensively studied since the early days of mobile phones. Cellular communication systems, as early as GSM, *attempted* to protect users' identity. Sensitivity to location privacy influenced the use of temporary identifiers (e.g., TMSI) which increased the difficulty of tracking users. In recent years, the attack surface of location privacy significantly expanded with the pervasiveness of mobile and sensing devices, open mobile platforms (running untrusted code) and ubiquitous connectivity. Users are also increasingly aware and concerned about the implications of disclosure of location information as reported in recent surveys [1], and the US Congress Location Privacy Protection Act of 2014 [2].

This material is based upon work partially supported by the National Science Foundation under Grants No. CNS-1409453, and CNS-1218197.

One user tracking threat example involves extracting the MAC address of probe packets that are periodically transmitted by Wi-Fi cards. This is known to be exploited by marketing companies and location analytics firms. In shopping malls for instance, companies such as Euclid Analytics state on their website that they collect "the presence of the device, its signal strength, its manufacturer, and a unique identifier known as its Media Access Control (MAC) address" [3]. This is used to analyze large spatio-temporal user traffic patterns. Another example is by the startup Renew, which installed a large number of recycling bins in London with the capability to track users. This allows Renew to identify not only if the person walking by is the same one from yesterday, but also her specific route and walking speed [4, 5]. The threats to privacy, as a result of exploiting MAC address tracking, triggered Apple to include a MAC address randomization feature in its iOS 8 release, receiving praises from privacy advocates [6].

While attacks based on the physical and link layer information are a serious concern [7], their practicality remains limited to adversaries with a physical presence in the vicinity of the user or requires access to the ISP infrastructure. Attacks that exploit the open nature of mobile platforms, including application stores, raise more concerns as they can be remotely triggered (e.g., from distant countries beyond the jurisdiction of a victim's country's courts of law), and require virtually no deployment of physical infrastructure. The simplest way to obtain a user's location is by accessing the mobile device location services which typically rely on GPS, Wi-Fi, or Cellular signals. To mitigate breaches of location privacy, mobile phones operating systems such as Android provide mechanisms for users to manage permissions and control access to sensitive resources and information. For instance, an Android mobile app needs to request a permission to access location information, allowing the user to decline. This is a good start despite the fact that many users are still careless about checking such permissions as illustrated by recent charges by the Federal Trade Commission against 'Brightest Flashlight' app for deceiving consumers and sharing the location information without their knowledge [8]. This app with 4.7 stars rating and over one million users is an example of seemingly innocuous applications that deceive users.

While a careful user can easily detect that a Flashlight app should not access his/her location information, a harder problem is how to protect users' location privacy against

DOI 10.1109/SP.2016.31

side channel attacks, when the app does not request any permissions. Mobile phones are embedded with a variety of sensors including a gyroscope, accelerometer, and magnetometer. This expanding attack surface is an attractive target for those seeking to exploit privacy information [9, 10], especially when users are becoming more aware of location tracking systems and attempt to minimize their exposure by disabling, limiting usage of, or removing tracking apps.

We investigate the threat and potential of tracking users' mobility without explicitly requesting permissions to access the phone sensors or location services. Currently, any Android application can access the gyroscope, accelerometer, and magnetometer without requiring the user permission or oversight. Even security aware users tend to underestimate the risks associated with installing an application that does not request access to sensitive permissions such as location. We focus on the scenario of a user traveling in a vehicle moving along roads with publicly available characteristics. We model a user trajectory as a route on a graph $G = (V, E)$, where the vertices represent road segments and the edges represent intersections. We formulate the identification of a user trajectory as the problem of finding the maximum likelihood route on G given the sensors' samples. Using techniques similar to trellis codes decoding, we developed an algorithm that identifies the most likely routes by minimizing a route scoring metric. Each of the vertices/edges is tagged with information such as turn angle, segment curvature and speed limit and can be extended to incorporate additional information such as vibration or magnetic signatures. In order to assess the potential of this approach in realistic environments, we developed a location tracking framework. The framework consists of six building blocks: (1) road graph construction from the OpenStreetMap project publicly available data, (2) processing sensor data and generating a compact sequence of tags that match the semantic of a graph route, (3) maximum likelihood route identification algorithm, (4) simulation tool, (5) mobile app to record sensor data, and (6) a trajectory inference for real mobility traces. We carried out extensive simulation on 11 cities around the world with varying population and road densities and topologies (including Atlanta, Boston, London, Manhattan, Paris, Rome), and preliminary real measurements in Boston and Waltham, MA (spanning over 980Km), on four Android phones, with four drivers. In the simulations, we show that for most cities with probability higher than 50% it is possible to output a short list of 10 routes containing the traveled route. In real experiments in the cities of Boston (resp. Waltham), Massachusetts, we report a probability of 30% (resp. 60%) of inferring a list of 10 routes containing the true route. Our contributions can be summarized as follows:

- A graph theoretic model for reasoning about location and trajectory inference in zero-permissions apps.
- A framework for processing sensors data, simulating/experimenting and evaluating location/trajectory inference algorithms on real city road networks.
- An efficient location/trajectory inference algorithm, that incorporates road segments curvature, travel time, turn angles, magnetometer information, and speed limits.
- A comprehensive simulated evaluation of the proposed algorithm's effectiveness on 11 cities and a preliminary real-world evaluation on 2 cities, demonstrating the feasibility of the attacks and efficiency of the algorithm.

While this paper focuses on how an adversary can infer a driving trajectory with a seemingly innocuous Android app that does not request any permissions from the user, this can easily lead to inferring the home and workplace of the victim. Further information about a user's identity can be derived by inspecting the town's public database. This work motivates the question of understanding the implications of mobile phone sensors on users' privacy in general. Enabling access to sensor information is critical for feature-rich applications and for their usability. However, preventing malicious exploitation and abuse of this information is critical.

II. Problem Statement

A. Motivating Scenario

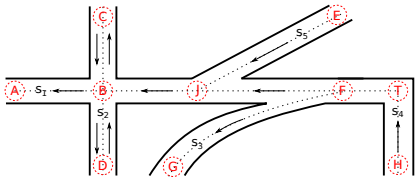
The victim is engaged in the act of driving a vehicle where she and an active smartphone are co-located within the aforementioned vehicle. The adversary's goal is to track the victim without the use of traditional position determining services such as GPS, cell tower pings, or Wi-Fi/Bluetooth address harvesting. To prepare for an attack, the adversary uploads a seemingly innocuous mobile app to a publicly accessible Application Store. The app is subsequently downloaded and installed by the victim on her smartphone. While providing the victim with its advertised features, this malicious app additionally collects sensor data from the accelerometer, gyroscope and magnetometer. This data is readily available as today's mobile operating systems such as Android and iOS do not yet limit access to these resources[1].

The attack is triggered when the app detects that a victim is starting to drive. Sensor data is recorded, without visible indication of the recording activity, and uploaded to a colluding server whenever Internet access is available. Based on the sensor data, the adversary can derive driving information such as turn angles, route curvatures, accelerations, headings and timestamps. Combined with publicly available geographic area attributes, the adversary can learn the actual route taken without the need of any location services/information.

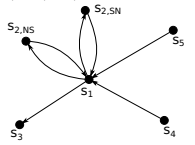
B. Location Privacy Leakage from Sensor Data

We introduce our terminology and notations used to describe the problem space. Consider a geographic area represented by a set of roads. Each road is either straight or has curvature that is detectable by the smartphone's sensors. When a road bisects, furcates, joins with other roads, or turns into a different direction, a connection is created (cf. Figure 1a). These connections divide roads into multiple so-called *atomic parts*, which only connect with other atomic parts at their

[1]As of Feb. 2016 (Android 6), access to accelerometer, gyroscope, and magnetometer is automatically granted during app installation without any user warnings or explicit permission requests.



(a) Connections are created when a road bisects (B), furcates (F), joins (J) with another road, or turns (T) into a different direction. Created atomic parts: $\overrightarrow{BA}$, $\overrightarrow{BC}$, $\overrightarrow{BD}$, $\overrightarrow{CB}$, $\overrightarrow{DB}$, $\overrightarrow{JG}$, $\overrightarrow{EJ}$, $\overrightarrow{FJ}$, $\overrightarrow{FG}$, $\overrightarrow{TF}$, $\overrightarrow{HT}$.



(b) Graph construction: every one-way road segment s_1, s_3, s_4, s_5 is represented by one vertex, while two vertices $s_{2,\text{NS}}$ and $s_{2,\text{SN}}$ are created for the north-south (NS) and south-north (SN) directions of the road segment s_2, respectively.

Fig. 1: Example of a geographic area and its mapping to a graph.

end points. Therefore, a geographic area $\mathcal{G}$ can be uniquely described as $\mathcal{G} = (\mathcal{B}, \mathcal{C}, \theta, \vartheta)$, where $\mathcal{B}$ is a set of atomic parts, and $\mathcal{C} = \{\chi = (r, r') | r, r' \in \mathcal{B}\}$ consists of connections $\chi = (r, r')$ which is an ordered pair indicating the connection between two atomic parts r and r'. The turn angle associated with a connection χ, which captures the real-world travel direction from r to r', is given by the function θ. A positive angle $\theta(\chi) > 0$ indicates a left turn, and a negative value $\theta(\chi) < 0$ indicates a right turn. Finally, the atomic parts preserve the road curvature determined by $\vartheta(r)$. The computation of θ and ϑ functions is based on the public map information.

We define a route taken by the driver as a sequence $\mathcal{R}$ of connected atomic parts, $\mathcal{R} = (r_1, \ldots, r_N)$, where $(r_i, r_{i+1}) \in \mathcal{C}$. Two routes $\mathcal{R}$ and $\hat{\mathcal{R}}$ are identical if the sequences of atomic parts have the same size and are component-wise equal, i.e., $\mathcal{R} = \hat{\mathcal{R}}$ if $r_i = \hat{r}_i$ for all i. Along the driving trajectory, the app obtains a set of sensor data $\mathcal{D} = \{(a_t, g_t, m_t)\}$ consisting of the vectors a_t, g_t and m_t taken from the accelerometer, gyroscope and magnetometer respectively. These vectors are sampled according to discrete time periods $t = 0, \delta, 2\delta, \ldots$, where δ is the sampling period. Based on $\mathcal{D}$, an adversary launches the tracking attack as follows.

Definition 1 (Sensor-based Tracking Attack)**.** *Let $\mathcal{A}$ be the attack deployed by the adversary on the received sensor data $\mathcal{D}$ given geographical area $\mathcal{G}$. The outcome of the attack is a ranked list $\mathcal{P}$ of K possible victim routes $\mathcal{P} = \mathcal{A}(\mathcal{G}, \mathcal{D}) = \{\hat{\mathcal{R}}_1, \ldots, \hat{\mathcal{R}}_K\}$, where $\hat{\mathcal{R}}_i$ has higher probability than $\hat{\mathcal{R}}_j$ of matching with the victim's actual trajectory, if $i < j$.*

Most interesting is whether a small set of results yield a route list containing the truth route. We aim to design an attack that satisfies this objective with success probability significantly higher than a random guess. In particular, we evaluate the attack efficiency according to the following metrics.

Definition 2 (Individual Rank)**.** *Given the user's actual trajectory $\mathcal{R}$ and the outcome of the attack $\mathcal{P} = \mathcal{A}(\mathcal{G}, \mathcal{D})$, the individual rank of the attack is k, if $\mathcal{R} = \hat{\mathcal{R}}_k$. The rank is uninteresting if $\mathcal{R}$ is not found in $\mathcal{P}$.*

The individual rank k reflects the attack's success in estimating that the victim's route is in top k of the outcome list. We are interested in the probability of such event happening, i.e., $P_k^{idv} := P(\mathcal{R} \in \{\hat{\mathcal{R}}_1, \ldots, \hat{\mathcal{R}}_k\})$, and evaluate the attack performance based on it (cf. Section V). While P_k^{idv} shows the possibilities of the victim's route being in a top k rather than telling which among the top is the actual route, we note that if k is reduced to 1, the probability P_1^{idv} is precisely the probability of finding the victim's route. This probability, though small (e.g., $P_1^{idv} \approx 13\%$ for Boston and $\approx 38\%$ for Waltham in our preliminary real-driving experiments), is still considerably high given the fact that the search space contains billions of routes. In practice, a top k with small k (e.g., $k \leq 5$) is a very serious breach. An adversary may collect such lists through the span of multiple days and refine the lists to find exactly the victim's daily commute route. Moreover, with more resources, the adversary can quickly check every potential route in the list to learn about the victim.

While the individual rank reflects the performance of the attack in terms of finding the exact route, in practice a rough estimation of the victim's route is usually enough to create a significant privacy threat. For example, targeted criminal activity (i.e., robbery and kidnapping) could result from the physical proximity knowledge derived from the attack. To justify this threat, we define a *cluster* of routes as a set $\{\hat{\mathcal{R}}_1, \ldots, \hat{\mathcal{R}}_l\}$, in which any two routes are similar. The similarity of routes $\hat{\mathcal{R}}$ and $\hat{\mathcal{R}}'$ is justified by $d(\mathcal{R}_i, \mathcal{R}_j) < \Delta$, based on the distance $d(\hat{\mathcal{R}}, \hat{\mathcal{R}}')$ and a threshold Δ, where we define $d(\hat{\mathcal{R}}, \hat{\mathcal{R}}') = \sum_{i=1}^{N-1} \| \text{Loc}(\hat{\chi}_i) - \text{Loc}(\hat{\chi}'_i) \|$ as the sum of distances between connection points $\hat{\chi}_i = (\hat{r}_i, \hat{r}_{i+1})$, $\hat{\chi}'_i = (\hat{r}'_i, \hat{r}'_{i+1})$ on $\hat{\mathcal{R}}$ and $\hat{\mathcal{R}}'$, and $\text{Loc}(\cdot)$ denotes the geographic coordinates.

By clustering, the attack now returns the outcome as a ranked list similar to one in Definition 1. Nevertheless, routes belonging to the same cluster are removed and only the best one of the corresponding cluster is included in the list. Specifically, if $\mathcal{A}_{cluster}(\mathcal{G}, \mathcal{D}) = \{\hat{\mathcal{R}}_1, \ldots, \hat{\mathcal{R}}_K\}$, then $d(\hat{\mathcal{R}}_i, \hat{\mathcal{R}}_j) \geq \Delta$ for any i, j, and $\hat{\mathcal{R}}_i$ is a representative route of cluster i. We now introduce the *cluster rank* metric as follows.

Definition 3 (Cluster Rank)**.** *Given the user's actual trajectory $\mathcal{R}$ and the outcome of the attack $\mathcal{P} = \mathcal{A}_{cluster}(\mathcal{G}, \mathcal{D})$, the cluster rank of the attack is k, if $d(\mathcal{R}, \hat{\mathcal{R}}_k) < \Delta$. The rank is uninteresting if no such k is found.*

Similarly to individual rank, we are interested in the probability of a route being in the top k of clusters, i.e., $P_k^{clt} := P(\mathcal{R} \in cluster_1 \cup \ldots \cup cluster_k)$. Based on the cluster rank metric, the adversary may eliminate similar routes and focus computation power on additional routes to improve the search results. Clustering is useful when similar roads / turns are present to effect a nearly identical result. For instance, the adversary may group routes with the same end points while ignoring different roads in between, or if they differ only at one end point (start or end), e.g., roads going from / to residential

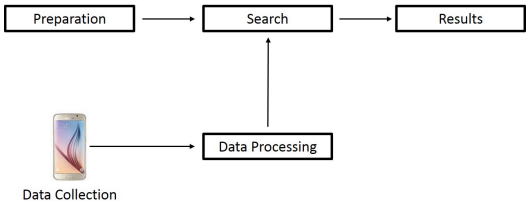
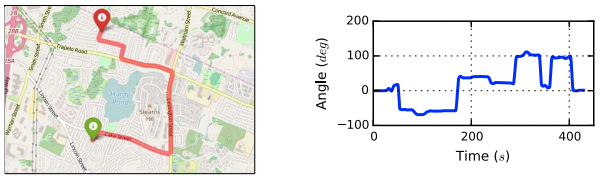


Fig. 2: Block diagram of our attack.



(a) Experimental route contains 6 turns from Start (green) to Stop (red).

(b) Angle trace contains 6 slopes (turns) and a few slight variations (curves).

Fig. 3: Experimental route and angle trace derived from gyroscope.

complex or office areas. This may give the adversary more confidence in a certain area than the individual rank.

C. Challenges

There are several challenges to the attack feasibility including the geographic area size, impact of sensor noise, driver behavior, and road similarity.

Area Size: The geographic area's size has an impact on the attack's accuracy. Even in small cities such as Waltham (Massachusetts, USA), there can be billions of possibilities for a victim's route. Moreover, routes with loops may also significantly increase the search space.

Noisy Sensor Data: The quality of sensor data is key for high attack accuracy. Unfortunately, today's smartphones are equipped with low-cost sensors that do not guarantee high accuracy. Sensor accuracy is also dependent on the sensor's previous state, e.g., the acceleration can immediately increase due to a street bump, but requires settling time before providing new useful information. Moreover, the magnetometer is influenced by nearby magnetic fields from fans, speakers and other electromagnetic devices.

Driver Behavior: The driving style of a driver also impacts the estimation of the actual route. For instance, a driver may frequently speed up or slow down due to traffic conditions or change lanes to overtake other vehicles. These actions induce additional noise in the sensor data in the form of spacial perturbations or distortions.

Road Similarity: Even in ideal scenarios when clean sensor data is obtained, the similarity of roads impacts the estimation of the actual route. This is especially true for cities with grid-like road structures such as Manhattan, New York.

D. Adversarial Model

Mobile Application: We assume that the rogue app collects sensor data continuously, either actively or in background, and intermittently transfers the data to the colluding server. As a typical one hour trip collects approximately 800KB of uncompressed data (80KB/hour for processed and compressed data), detection by a user in the form of degraded network behavior should be negligible in locations with active 3G and 4G networks or nominal Wi-Fi signal strength.

Device Position: We compensate for device orientation at attack initiation (i.e., the time when the vehicle starts moving). During travel, the device's orientation should remain relatively fixed within the reference frame of the vehicle. This supports attack efficacy in a variety of realistic phone placements such as the phone attached to a mount, residing in a cup holder, in the driver's pocket or in her handbag.

Location Information: While the attack described in this work does not rely on the location information of the victim's trajectory at any point (e.g., no known starting point), we assume a rough knowledge of her living/travel area (e.g., known to live in/frequent Manhattan, New York).

III. Approach

A. Overview

In its basic form, the system consists of a smartphone that collects data and a post-processing server that generates a ranked list of potential routes or clusters of routes. Figure 2 illustrates the design's main components.

- *Preparation:* Road information from public map resources are extracted and converted to specific database structures. This is a one-time initialization step and the structures can be reused for all subsequent attacks.
- *Sensor Data Collection:* Sensor data is recorded by the app and sent to the colluding server. This step uses movement detection based on accelerometer data to trigger sensor recording exclusively during vehicle movement.
- *Data Processing:* On receiving the sensor data, the server analyzes the data to derive the victim's trace of turn angles, curvatures, heading, accelerations and timestamps.
- *Search:* The search algorithm is run on the processed data and a ranked list of matching routes is produced.

Sensor data provides important information about a victim's movements. Among the three sensor types (accelerometer, gyroscope and magnetometer), the gyroscope is the most useful for this attack because of the following reasons: (a) The gyroscope provides more accurate data than the others; (b) The gyroscope reveals turn angles and road curvature of the undertaken route which are nearly static attributes and traceable on a public map resource. We heavily weight the gyroscope data in this attack as the accelerometer and magnetometer strongly depend on dynamic factors such as traffic/road conditions or proximate magnetic fields, which are challenging to predict. Timestamps, accelerometer and magnetometer readings are used as supporting data to reduce noise and refine the results.

Data received from the gyroscope is a sequence of three dimensional vectors reporting the rate of angular change along the victim's trajectory. Figure 3 illustrates an example of an experimental route and corresponding angle sequence (processed

from gyroscope data) relative to initial heading. Here, large changes in the angle trace indicate turns at intersections. Right and left turns are represented by negative and positive slopes, while minor variations (e.g., less than 30° in the example) in between are attributed to road curvature.

We transform the Sensor-based Tracking Attack (Definition 1) to the problem of matching the angle trace and curvature with possible routes. The objective is to identify sequences of intersections and curvatures that match the slope change found in the angle trace. Our approach consists of graph construction based on OpenStreetMap [11], a public map resource, and matching routes on this graph with the actual angle trace using techniques similar to trellis codes decoding [12]. Note that in our context, the graph size is many orders of magnitude larger than typical trellis codes used in communications. In addition, while trellis codes make transitions and produce an output at each state, the victim's trajectory may traverse any number of atomic parts (transitions) without making a turn (output), rendering the problem more complex.

B. Graph Construction

Our search is performed on a directed graph structure. For the sake of clarity, we first introduce some new definitions. Consider a geographic area $\mathcal{G} = (\mathcal{B}, \mathcal{C}, \theta, \vartheta)$. We assert that a connection between two atomic parts is a *non-turn connection* if the turn angle at the connection is below a threshold ϕ_{g3} (e.g., $\phi_{g3} = 30°$, cf. Section IV-D). In this graph construction, we are interested in identifying such connections that can connect atomic parts together to create straight or curvy roads without including significantly large turns. We call such sequence of non-turn connected atomic parts a *road segment* (or simply *segment*). Specifically, a sequence $\boldsymbol{s} = (r_1, \ldots, r_l)$, where $r_i \in \mathcal{B}$, is a road segment if $\theta(r_i, r_{i+1}) \leq \phi_{g3}$ for $i = 1, \ldots, l-1$. Intuitively, a segment is a route without large turns at connections between its atomic parts. Additionally, we call segment $\boldsymbol{s}$ a *maximal-length segment*[2] if no atomic part can be added to $\boldsymbol{s}$ to form a longer segment while still preserving the non-turn condition. When a connection between two atomic parts has a turn angle greater than ϕ_{g3}, it becomes a connection between two segments, i.e., if $r \in \boldsymbol{s}, r' \in \boldsymbol{s}'$ and $\chi = (r, r') \in \mathcal{C}$, then $\theta(r, r') > \phi_{g3}$. In this case, we call χ a *segment connection* or simply an *intersection*.

Our idea for constructing the directed graph $G = (V, E)$ is to represent each segment $\boldsymbol{s}$ by a vertex $v \in V$ and each segment connection χ by an edge $e \in E$. An example construction is illustrated in Figure 1b. Intuitively, one will stay at one vertex on the graph as long as she does not turn into another segment. A turn at an intersection makes her traverse to another vertex through an edge connecting them. Based on the public map resource, we accordingly build our graph for the whole geographic area. For each edge e corresponding to segment connection χ, we use $\theta(\chi)$ as the edge's weight. The length, speed limit, and curvature of a road segment $\boldsymbol{s}$ are stored as attributes of the corresponding vertex v. This information combined with the sensor data is used to match the victim's angle trace during the search. We note that for any two segments $\boldsymbol{s}$ and $\boldsymbol{s}'$ such that $\boldsymbol{s}' \subset \boldsymbol{s}$ (i.e., one is a sub-sequence of the another), we simply remove $\boldsymbol{s}'$ from the graph, because any atomic part r and connection χ of $\boldsymbol{s}'$ involved in the route search are also present in $\boldsymbol{s}$, rendering $\boldsymbol{s}'$ redundant. Therefore, graph G essentially contains only vertices corresponding to maximal-length segments, resulting in more efficient route search with greatly reduced graph size.

[2]Maximal-length segment is analogous to a longest route between two nodes with an additional condition: weight (turn angle) must be small.

C. Search Algorithm

Our search algorithm evaluates the routes when traversing the graph and keeps the good routes at the end of each step. When the search completes, a list of candidates is returned with their evaluated score. At each step of the search, outgoing edges from a given vertex are investigated for the next candidate segment connection. The evaluation uses a metric that is based on the difference between the edge weights and the angle trace's slopes. We improve the performance of the basic search by incorporating an evaluation of segment curvatures on the candidate routes. The curvatures of potential routes are computed from coordinates of points extracted from the map, while curvatures of the actual route are estimated based on gyroscope samples collected between the slopes. These details are discussed in Sections IV-A and IV-B.

D. Refining the Results

As the search based on gyroscope data is unaware of the absolute orientation of the routes, we refine the results and reduce the search time by using heading information derived from the magnetometer to immediately eliminate bad routes (e.g., east-west routes are filtered out when the actual trace indicates north-south direction).

In addition, we exploit the accelerometer to identify idle states and discard samples in such periods for better estimation. We also extract speed information, available from Nokia's HERE platform [13], for each road and filter out routes by comparing the actual travel time between intersections with the time estimated for the segment under investigation. We provide the details of this discussion in Sections IV-C and IV-D.

IV. System Design

A. Basic Search Algorithm

The search technique includes maintaining a list of scored candidate victim routes while traversing the graph. Candidate routes have higher probability of matching the recorded mobility trace. For the current discussion, we assume that the adversary only exploits the gyroscope data to launch the attack, i.e., we consider only g_t from $\mathcal{D} = \{(a_t, g_t, m_t)\}$. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$ be the derived sequence of turn angles at N intersections after processing gyroscope data g_t. The details of sensor data processing are discussed in Section IV-D. In Sections IV-B and IV-C we refine the algorithm and improve the performance by adding filtering rules and applying a more

complex scoring method. Our goal at the moment is to find $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N) \in G$, the potential sequences of turns that maximize the probability of matching $\boldsymbol{\theta}$ given the observation of $\boldsymbol{\alpha}$. This probability, denoted $P(\boldsymbol{\theta}|\boldsymbol{\alpha})$, can be rewritten as:

$$P(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{P(\boldsymbol{\theta}, \boldsymbol{\alpha})}{P(\boldsymbol{\alpha})} = \frac{P(\boldsymbol{\alpha}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{\alpha})}$$

As $P(\boldsymbol{\alpha})$ is the probability of a measurement $\boldsymbol{\alpha}$ without conditioning on $\boldsymbol{\theta}$, it is independent of $\boldsymbol{\theta}$. Thus, maximizing $P(\boldsymbol{\theta}|\boldsymbol{\alpha})$ is equivalent to maximizing $P(\boldsymbol{\alpha}|\boldsymbol{\theta})P(\boldsymbol{\theta})$. The distribution of a priori probability $P(\boldsymbol{\theta})$ may depend on the driver, city, and day/time of travel (e.g., home-to-work and work-to-home routes during weekdays have significantly higher probability than other routes). Since our goal is to demonstrate the generality of the attack even if the adversary knows nothing about the victim's travel history, we consider $P(\boldsymbol{\theta})$ to be equiprobable, i.e., any route has the same probability of being taken by the victim. This presents the worst-case attack scenario and gives a lower bound on the performance. If the a priori probability $P(\boldsymbol{\theta})$ is known, we expect the attack to achieve higher success probability than the performance we report in this work. Under the assumption of equiprobable a priori probability, the goal of maximizing $P(\boldsymbol{\alpha}|\boldsymbol{\theta})P(\boldsymbol{\theta})$ is equivalent to maximizing the probability $P(\boldsymbol{\alpha}|\boldsymbol{\theta})$ alone.

Samples taken from the gyroscope include noise as an additional unknown amount in the angle trace, yielding the angle $\boldsymbol{\alpha} = \boldsymbol{\theta} + \boldsymbol{n}$, where $\boldsymbol{n}$ is the random noise vector. We will show through experimental results in Section V, that the gyroscope noise can be approximated by a N-dimensional zero-mean normal distribution $\mathcal{N}(0, \sigma)$ with standard deviation σ. Accordingly, $P(\boldsymbol{\alpha}|\boldsymbol{\theta})$ can be rewritten as:

$$P(\boldsymbol{\alpha}|\boldsymbol{\theta}) = P(\boldsymbol{n} = \boldsymbol{\alpha} - \boldsymbol{\theta}) = \left(2\pi\sigma^2\right)^{-\frac{N}{2}} \exp\left(-\frac{\|\boldsymbol{\alpha} - \boldsymbol{\theta}\|^2}{2\sigma^2}\right)$$

where $\|\cdot\|$ indicates the L_2 norm of a vector. As $\left(2\pi\sigma^2\right)^{-\frac{N}{2}}$ is constant for a fixed N and σ, maximizing $P(\boldsymbol{\alpha}|\boldsymbol{\theta})$ is now equivalent to minimizing $\|\boldsymbol{\alpha} - \boldsymbol{\theta}\|$. Therefore, the adversary obtains the optimal solution as stated in Theorem 1.

Theorem 1. *Given graph G and a turn angle trace $\boldsymbol{\alpha}$ with normally distributed noise, the optimal route tracking solution is $\boldsymbol{\theta}^* = \arg\max_{\theta \in G} \|\boldsymbol{\alpha} - \boldsymbol{\theta}\|$.*

Based on Theorem 1, our search algorithm (Algorithm 1) aims at finding $\boldsymbol{\theta}$ that minimizes $\|\boldsymbol{\alpha} - \boldsymbol{\theta}\|$. The main idea is to maintain a list of potential vertices (i.e., road segments) from which we develop the possible routes. The algorithm takes as input the graph $G = (V, E)$ and a sequence $(\alpha_1, \ldots, \alpha_N)$. The search consists of N rounds corresponding to a trace of N intersections. While the algorithm is similar to trellis codes decoding techniques in which paths are built up, maintained or eliminated according to a metric, our search is improved by filtering routes based on specific selection rules and keeping only top candidate routes after a number of iterations.

The algorithm starts by considering all vertices of the graph as potential starting points (initialization $U_0 \leftarrow V$). In each

Input: $G = (V, E)$, $\alpha_1, \ldots, \alpha_N$
Output: U_N
1 Initialization: $U_0 \leftarrow V$; $U_1 \leftarrow \emptyset$; $\ldots U_N \leftarrow \emptyset$;
2 **for** $k = 1$ **to** N **do**
3 **for** $u \in U_{k-1}$ **do**
4 **for** $v \in V$ *such that* $(u, v) \in E$ **do**
5 **if** $\text{filter}(u, v, \alpha_k)$ *passed* **then**
6 $v.score \leftarrow u.score + \text{scoring}(u, v, \alpha_k)$;
7 $v.prev \leftarrow u$;
8 $U_k \leftarrow U_k \cup \{v\}$;
9 **end**
10 **end**
11 **end**
12 $U_k \leftarrow \text{pick\_top}(U_k)$;
13 **end**

Algorithm 1: Search Algorithm

k-th round, we build a new list U_k of potential vertices as follows. For each vertex $u \in U_{k-1}$, we explore all its outgoing edges (u, v). During this traversal (line 4 – 10), filtering is applied (line 5) to eliminate such vertices/segments whose corresponding map data deviates too much from the actual sensor data. In this basic algorithm, the filter checks if the turn angle (i.e., the edge weight) between the current vertex u and the candidate vertex v is within a specific range of the actual turn α_k. Specifically, an edge (u, v) passes the filter, only if $|\theta(u, v) - \alpha_k| \leq \gamma$, in which case v is put into U_k as a candidate for the next search iteration (line 8). The threshold γ depends on the quality of sensor data and is evaluated in Section V. We note that when a vertex v does not satisfy the filtering rules, it simply means v is not used as a starting point in the next iteration, but v may appear again if other starting points connecting to v satisfy the conditions.

At the same time when filtering is passed, the edge (u, v) is also evaluated for the likelihood to match the actual trace by the scoring function (line 6). The score for each k-th turn is computed by

$$\text{scoring}(u, v, \alpha_k) = d(\alpha_k, \theta(u, v)) = |\alpha_k - \theta(u, v)|, \quad (1)$$

where we compute the angle distance based on L_1 norm instead of L_2 norm for two main reasons: (a) computing L_1 norm requires less overhead; (b) in practice, we observe that L_1-based matching generally outperforms L_2-based, because gyroscope errors are usually small (cf. Section V-A), allowing L_1-based estimation to better overcome sparse large errors, while L_2 norm tends to amplify such errors. The score of every route is initialized to 0 (line 1) and evolves to $\sum_{k=1}^{N} d(\theta(u, v), \alpha_k)$ after N iterations. When updating the score, we additionally store the previous vertex ($v.prev$) of the candidate in order to trace back the full route (without storing the whole route) at the end of the search. We also note that as the list of candidates is developed through each iteration with non-negative metric, finding the actual route with loops is possible, because loops simply increase the score and are treated as regular routes (i.e., the search will terminate).

Since routes with lower score have higher matching probability $P(\boldsymbol{\alpha}|\boldsymbol{\theta})$, we only keep the top K candidates at the end of every iteration by calling $\mathrm{pick\_top}$ function (line 12). It is noted that depending on attack configuration, $\mathrm{pick\_top}$ may shorten the list of candidates only after some specific round. At the end of the search, based on U_N and previous vertex information stored for each candidate, the outcome $\mathcal{P} = \{\hat{\mathcal{R}}_1, \ldots, \hat{\mathcal{R}}_K\}$ is appropriately produced and returned.

Effect of Filtering and Top Selection: While scoring gradually distinguishes routes from each other, filtering can immediately eliminate a route at early stage, which will not be recovered later. There is a trade-off when determining the filtering thresholds. A tight rule can reduce the search time but may result in pruning more good routes due to early errors, whereas loose criteria reduces false elimination rate but increases running time and memory consumption. Similarly, selecting top candidates after some specific iterations can decrease the search time yet potentially removes good candidates that are bad at early stages. We leave the rigorous analysis of such parameters as future work. Instead, based on simulations and real driving experiments, we select appropriate parameters with respect to both attack performance and computation constraints such as memory and timing requirements. Using such parameters, we can verify that filtering and top candidates selection can actually improve the attack efficiency.

B. Advanced Algorithm & Scoring Metrics

While Algorithm 1 illustrates the main idea of our search technique, it essentially represents a baseline attack, because it relies only on the sequence of observed turn angles as the single input source to the algorithm. We now incorporate, into the basic search algorithm, the curvature of the undertaken route and the travel time between turns.

Curve Similarity: We define the curvature of the route as a sequence of angles between intersections. Consider the victim's travel between the k-th and $(k+1)$-th intersections and let $T_k\delta$ (δ is the sampling period, and $T_k = 1, 2, \ldots$) be the victim's travel time for that distance. The curvature is then expressed by $\boldsymbol{C}_k = (\alpha_{k,1}, \ldots, \alpha_{k,T_k})$, where $\alpha_{k,i}$ are instantaneous directions at sampling time $i\delta$ on the k-th curve.

In order to match the sampled curvature with a candidate curve, we assume that the vehicle movement along the curve is at constant speed. On one hand, this simplifies the estimation and greatly decreases the computation burden for each route. Since on the other hand, no available data can provide sufficient accuracy of the instantaneous vehicular velocity, finding the best curve fit is challenging. However, our evaluation shows that curve matching with constant speed assumption considerably improves the attack performance. Specifically, we compute the angle sequence on each candidate curve as follows. For a candidate segment corresponding to a vertex u (which is either straight or curvy), we divide it into T_k equal-length sub-segments and consider each sub-segment as a straight line, then we find the orientations of sub-segments based on their geographic coordinates. Therewith, we obtain $\boldsymbol{\vartheta}_u = \vartheta(u) = (\vartheta_{u,1}, \ldots, \vartheta_{u,T_k})$ as the curvature of u, where $\vartheta_{u,i}$ is the orientation of the i-th sub-segment.

Our goal is to maximize the probability $P(\boldsymbol{\vartheta}_u|\boldsymbol{C}_k)$ of matching a candidate curve $\boldsymbol{\vartheta}_u$ given the victim's curve $\boldsymbol{C}_k$ observed by the adversary. As discussed previously in Section IV-A, due to the assumption of victim route equiprobability, we instead search for such $\boldsymbol{\vartheta}_u$ that maximizes

$$\begin{aligned} P(\boldsymbol{C}_k|\boldsymbol{\vartheta}_u) &= P(\boldsymbol{n} = \boldsymbol{C}_k - \boldsymbol{\vartheta}_u) \\ &= \left(2\pi\sigma^2\right)^{-\frac{T_k}{2}} \exp\left(-\frac{\|\boldsymbol{C}_k - \boldsymbol{\vartheta}_u\|^2}{2\sigma^2}\right) \end{aligned}$$

where $\boldsymbol{n} \leftarrow \mathcal{N}(0, \sigma)$ is the normally distributed random vector approximating the gyroscope noise. We determine the curve similarity by

$$d(\boldsymbol{C}_k, \boldsymbol{\vartheta}_u) = \frac{1}{T_k} \sum_{i=1}^{T_k} |\alpha_{k,i} - \vartheta_{k,i}|.$$

We note that the curve similarity, different from turn scoring in Equation (1), is normalized to mitigate the effect of bias scoring due to error accumulation on long curves (large T_k).

Travel Time Similarity: The tracking of the actual route based on turn angles and curvature information so far does not take into account the time scale of the victim's travel on each road segment. To incorporate this information in the attack, we extract from Nokia's HERE map [13] the maximum allowed speed for every road in the geographic area $\mathcal{G}$ and compute the minimum time required to travel from one intersection to another along each road segment. Let $t_k \in \mathcal{D}$ be the actual time spent by the victim to travel from the k-th to the $(k+1)$-th intersection, and $\tau(u, v)$ be the minimum required time (computed from speed limit) for traveling from the last intersection to the current intersection (u, v) on the candidate route. The metric for the travel time similarity is computed by

$$d(t_k, \tau(u, v)) = |t_k - \tau(u, v)|.$$

Final Scoring Function: By incorporating the likelihood of the turn angles, the curvature, and the travel time along the search route, our final scoring function becomes $\mathrm{scoring}(u, v, \alpha_k, t_k, \boldsymbol{C}_k)$ and is computed as

$$\omega_A d(\alpha_k, \theta(u, v)) + \omega_T d(t_k, \tau(u, v)) + \omega_C d(\boldsymbol{C}_k, \boldsymbol{\vartheta}_u) \quad (2)$$

where different weights $\omega_A, \omega_T, \omega_C$ can be selected dependently on the geographic area.

C. Filtering Rules

We extend the filtering rules in Algorithm 1 by exploiting the magnetometer and the phone's system time to quickly exclude bad routes during the search.

1) Heading Check: At the time of each turn at an intersection, we extract the heading of the vehicle from the magnetometer sensor sample and check that the next segment's direction should be close to the heading direction after turning. In practice, we observe that since the magnetometer may be influenced by an external magnetic field, the heading derived from the magnetometer is not always accurate.

In order to exploit this information properly, we first verify the magnetometer data to be reliable based on the magnitude of the heading vector, which essentially depends[3] on the specific geographic area $\mathcal{G}$. Specifically, the reliability is established if $M_l \leq \|m_t\| \leq M_h$, where $m_t \in \mathcal{D}$ is the magnetometer vector, and M_l, M_h are lower and upper bounds that depend on $\mathcal{G}$. Only after the reliability is assured, the orientation check is performed. Specifically, with h_k denoting the heading vector (obtained after calibrating and rotating magnetometer vectors m_t, cf. Section IV-D) of the vehicle after turning at the k-th intersection between u and v, and $\vartheta_{v,1}$ be the orientation of the first sub-segment of segment v. The heading check is satisfied, if $|h_k - \vartheta_{v,1}| \leq \phi_m$, where ϕ_m is the magnetometer error threshold. Note that in case of unreliable magnetometer data, the check is not performed but v is not eliminated.

2) Travel Time Check: Due to the maximum speed limit on each road, the travel time cannot be arbitrarily small. Our idea for pruning impossible routes is as follows. Given the actual travel time duration $t_k \in \mathcal{D}$ between the k-th and $(k+1)$-th intersections, the maximum distance traveled by the vehicle is $L_k \leq L_{\max} = \beta V_{\max} t_k$, where $V_{\max}$ is the regulated speed limit, and $\beta \geq 1$ is the over-speeding ratio that can be reached by the vehicle. Consequently, during the search we only keep such candidate routes that are not longer than $L_{\max}$. To reduce the computation overhead, we instead precompute $t_v = \frac{L_v}{V_{\max}}$ for each candidate road segment v of length L_v, and our timing rule becomes $t_k \geq \frac{t_v}{\beta}$, i.e., $L_v \leq L_{\max}$. We emphasize that in realistic scenarios, since the vehicle may drive at any speed below the limit or may get stuck in the traffic for an unpredictable duration, the travel distance can be arbitrarily small. Therefore, no non-zero lower bound on segment length is established.

D. Sensor Data Processing

A big challenge in implementing this attack is extracting accurate route information from noisy sensor data. Along with the external factors discussed before (e.g., potholes, bumps, road slopes, magnetic field and driver behavior), some internal misconfiguration may also introduce errors in the data.

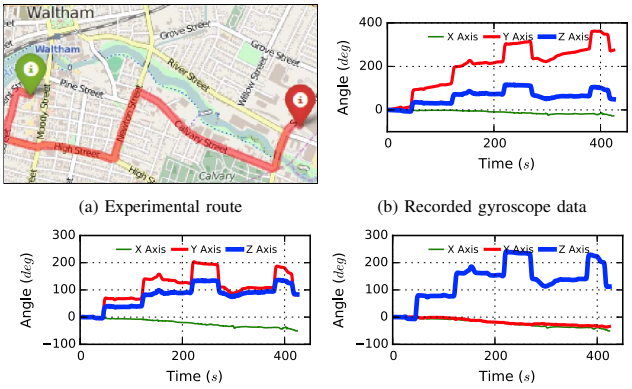
Axis Misalignment: Sensor x, y and z axes may not have perfect orthogonal alignment. This causes a bias in the sensor values which can be defined as the deviation from the expected x, y and z values when the device is at rest. The bias can typically be removed by subtracting them from the reported x, y and z sensor values.

Thermal Noise: The sensor's x, y and z axes values may also vary with the device/sensor temperature. Some Operating Systems compensate for this noise by pre-filtering the data, but at the cost of reduced accuracy.

Given these errors, we decompose the sensor data processing into error compensation and trace extraction tasks.

1) Error Compensation: Error compensation consists of a calibration phase followed by rotation of the data. Note that while our discussion focuses on gyroscope data, similar tasks can be performed for accelerometer.

[3]Heading vector's magnitude is higher for Temperate than Tropical cities.



(a) Experimental route (b) Recorded gyroscope data

(c) Calibrated gyroscope data (d) Rotated gyroscope data

Fig. 4: Error compensation steps for gyroscope data.

Calibration: The gyroscope sensor bias and vehicle vibration result in angle drift, i.e., the values change linearly[4] in time even at idle. An example of experimental route is shown in Figure 4a. As gyroscope data is reported as a sequence of angle change between sampling periods, we integrate them over time to obtain the relative (with respect to the initial recording) angle sequence in x, y, z axes depicted in Figure 4b, which shows a large positive drift in the y axis. To compensate for the drift, we assume the vehicle is at parked state in the calibration phase (we note that this is only required once for subsequent attacks). The drift vector is estimated as $\overline{\Delta\alpha} = \mathbf{E}[\Delta\alpha/\Delta t]$, the expected angle change rate. The calibration is then performed by subtracting $\overline{\Delta\alpha}$ from the angle sequence (Figure 4c). Note that complete removal of drift is a difficult task and would require more computation-expensive mechanisms, e.g., *Sensor Fusion* algorithms.

Rotation: Recall that a victim can place her smartphone in any orientation in the vehicle. To simplify the attack computation, we rotate the sensor data to a reference coordinate system, where the x axis points from left to right of the driver, the y axis aligns with the heading direction of the vehicle, and the z axis points upward perpendicularly to the Earth surface. After rotation, the x and y values are then used to measure $pitch$ and $roll$ respectively, while turn angle information is indicated in the z axis (Figure 4d).

2) Trace Extraction: In the reference coordinate system, we use the z values of gyroscope data to extract the victim's turn angles at intersections and curves between them, while acceleration vectors are used to improve the search performance by detecting vehicle's idle states.

Turn and Curve Detection: Based on z values of gyroscope vectors after rotation, left and right turns are distinguished according to positive and negative angle changes. Our idea for identifying intersections is illustrated in Figure 4d, where left turns are identified by an increasing slope within a short period of time and right turns correspond to decreasing slope. More precisely, let z_i be the gyroscope value on the z

[4]Our observation suggests linear model well approximate the angle drift.

TABLE I: Default parameters used in evaluation.

Parameter	Value
Scoring weights	$\omega_A = 2.5, \omega_T = 0.1, \omega_C = 2.5$
Turn/curve detection threshold	$\phi_{g1} = 1^\circ, \phi_{g2} = 10^\circ, \phi_{g3} = 30^\circ$
Turn angle filtering threshold	$\gamma = 60^\circ$
Heading filtering threshold	$\phi_m = 90^\circ$
Travel time filtering threshold	$\beta = 1.5$
Noise distribution	$\mu = 0.003, \sigma = 7.54$
Sampling period	$\delta = 100\,\text{ms}$
Top candidates limit	$K = 5000$, for iterations $k \geq 2$

axis at time $i\delta$ in the rotated angle trace. An intersection is found if it satisfies *all* the following conditions:

1) *Start turn:* The angle change between time $i\delta$ and $(i+1)\delta$ is higher than a threshold ϕ_{g1}, i.e., $|z_{i+1} - z_i| > \phi_{g1}$, which captures the event that the vehicle is starting to make a turn or enter a curve.
2) *Large deviation:* The largest deviation on a slope under investigation must be greater than a threshold ϕ_{g2}, i.e., $\max_{i \in \text{slope}} |z_{i+1} - z_i| > \phi_{g2}$. This distinguishes the real turn from a slight curve on the route.
3) *Large turn angle:* If the difference between the first and the last angle on a slope is greater than ϕ_{g3}, i.e., $|z_{i+n} - z_i| > \phi_{g3}$, the slope is recognized as a real turn, and the value $\alpha_k = z_{i+n} - z_i$ is the turn angle for the corresponding k-th intersection.

A curve is recognized if the first condition is met, but the other two conditions do not hold at the same time. In other cases, the road segment under investigation is considered a straight segment. The parameters ϕ_{g1}, ϕ_{g2}, and ϕ_{g3} are configured accordingly to the geographic area.

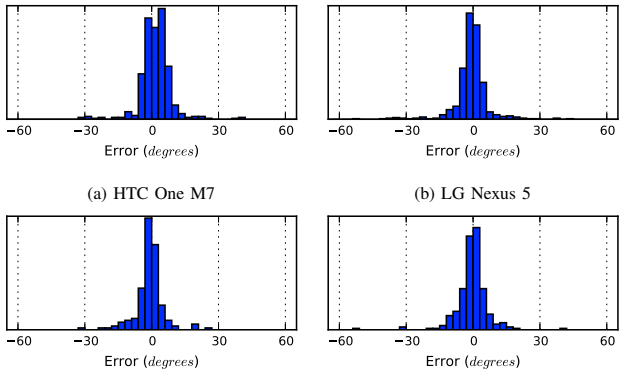
Idle State Detection: Despite the limited accuracy of the accelerometer to reveal the precise instantaneous vehicular speed, we can still exploit it to differentiate an idle state (e.g., vehicle stops at traffic lights) from movement on a straight road. In both cases, the gyroscope does not expose large enough variations for detecting angle changes with adequate accuracy. However, with accelerometer, the former case results in nearly zero magnitudes of acceleration vectors, while the values are considerably larger with higher fluctuations for the latter case. With idle states detected, we can better estimate the actual non-idle time and improve the attack performance.

V. Evaluation

In this section, we evaluate the attack efficiency based on simulations and real driving experiments. First, we justify the accuracy of gyroscope sensor and present our selection criteria for cities chosen for evaluation. Subsequently, we present our simulation and real driving results with a discussion on attack performance and the implications on user privacy. The attack parameters with default values are given in Table I.

A. Accuracy of Gyroscope

While the accelerometer and magnetometer accuracy depend heavily on the environment rendering them more suitable for filtering improbable routes with relaxed rules, the gyroscope sensor is less impacted by the environment. Therefore, it is important to first justify the accuracy of gyroscope data.



(a) HTC One M7 (b) LG Nexus 5

(c) LG Nexus 5X (d) Samsung S6

Fig. 5: Gyroscope noise distributions measured in real driving experiments for different smartphones.

For this justification, we measure the accuracy based on real driving experiments as follows. We use 4 smartphones of different brands and models, and take total 70 driving routes in both Boston and Waltham (Massachusetts, USA). To assess the gyroscope errors, we extract the truth turn angles θ_i of taken routes from OpenStreetMap, then for each θ_i, we obtain the gyroscope angle α_i (after sensor data processing phase) and compute turn errors $e_i = \alpha_i - \theta_i$. As observed from Figure 5 showing histogram of e_i, the error distribution for each phone closely follows a *normal distribution* with more than 95% of errors below 10°. Table II indicates almost equal noise standard deviation of each device. For all routes combined for 4 phones, the mean μ and standard deviation σ values are 0.003 and 7.54, respectively.

TABLE II: List of phones tested for accuracy along with the number of turns, and the gyroscope noise's mean and standard deviation.

Phone	No. Turns N	Mean μ	Std. dev. σ
HTC One M7	482	1.73°	7.07°
LG Nexus 5	618	-0.77°	7.89°
LG Nexus 5X	170	-1.12°	6.40°
Samsung S6	238	-0.57°	7.51°

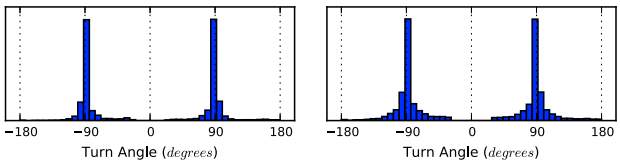
B. Selection of Cities

To assess the attack's impact on diverse cities of the world, we identified 11 cities for simulations based on their size, density and road structure. Table III summarizes their attack-related characteristics such as the graph size (number of vertices $|V|$ and edges $|E|$) and distribution of turn angles at intersections (mean μ_{turn} and standard deviation σ_{turn}).

Big cities such as Atlanta, Boston, London, Madrid, Paris, and Rome create larger graphs than the rest according to our construction method. While Manhattan is quite populated, it has the smallest graph in our set, because our graph only contains maximal-length segments. Nevertheless Manhattan is dominated by long east-west and north-south roads, many of which are parallel. Despite having similar graph size as Manhattan, Concord and Waltham are attributed to a larger standard

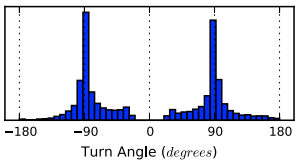
TABLE III: List of cities used for evaluation with their characteristics: graph size ($|V|, |E|$) and turn angle distribution ($\mu_{\text{turn}}, \sigma_{\text{turn}}$).

City	|V|	|E|	Mean μ_{turn}	Std Dev σ_{turn}
Atlanta, GA, USA	10529	25557	88.73°	17.58°
Berlin, Germany	4708	19752	88.21°	19.87°
Boston, MA, USA	8010	22149	89.69°	20.52°
Concord, MA, USA	3049	6467	88.13°	29.58°
London, UK	9468	21968	87.83°	20.38°
Madrid, Spain	10012	30144	86.41°	25.13°
Manhattan, NY, USA	1033	3699	89.23°	17.81°
Paris, France	6744	11204	86.35°	26.26°
Rome, Italy	9408	20577	85.98°	26.15°
Sunnyvale, CA, USA	5592	12302	88.59°	16.00°
Waltham, MA, USA	3366	9437	88.93°	20.53°

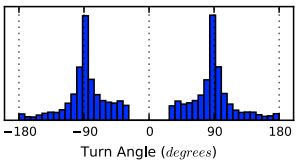


(a) Sunnyvale

(b) Boston



(c) Rome



(d) Concord

Fig. 6: Distribution of intersection turn angles in selected cities.

deviation σ_{turn}. The top cities of grid-like road structure are Atlanta, Sunnyvale, and Manhattan with low values of σ_{turn}. Boston, Berlin, and London have more spread out turns, but not as much as Paris and Rome. Figure 6 shows the turn angle distributions for some selected cities, where we observe that the majority of intersections in Sunnyvale are $90°$ while Boston, Rome, and Concord have more unique turns.

C. Creation of Simulated Routes

For each selected city, we test the feasibility of the attack by running the system on simulated routes. In case of Boston and Waltham, we also collect 70 driving experiments used for experimental evaluation described in Section V-E. Both sets of simulated and real routes are converted to the same format for compatibility, in which the user's route is represented as a sequence $\mathcal{U} = ((h_1, \alpha_1, t_1, \boldsymbol{C}_1), \ldots, (h_N, \alpha_N, t_N, \boldsymbol{C}_N))$. The heading vector h_i represents the direction of vehicle right before entering an intersection with turn angle α_i, whereas t_i and $\boldsymbol{C}_i$ are the time duration and curvature of the travel between the previous intersection and the next one.

Route Generation: Based on the constructed graph $G = (V, E)$ for a selected city $\mathcal{G}$, each simulated route is created by first randomly choosing a route length $N \leftarrow \{4, \ldots, 11\}$, then the route is formed by adding N random connected segments that satisfy (a) turn angle constraint: $30° \leq |\alpha_i| \leq 150°$, (b) travel time constraint: $t_i \geq 10\,\text{s}$. Note that as these segments are maximal-length, the system may choose connections that are large distances apart for larger segments. In our simulations, the generated routes are between $\approx 0.5\,\text{km}$ and $\approx 48.15\,\text{km}$ with an average length of $\approx 7.15\,\text{km}$.

Noise Adding: To simulate realistic scenarios, we add various levels of noise to the route's characteristics. The magnetometer noise n_m is added to h_i by a *uniform distribution* such that $-90° \leq n_m \leq 90°$. To mimic the travel time in practice, we add *uniform distributed* noise n_t to t_i such that $\frac{t_i}{\beta} \leq t_i + n_t \leq \frac{t_i}{\beta'}$, where β is the over-speeding ratio, and β' is the lower bound speed ratio which attempts to model the slow driver or traffic jam. While β is fixed to 1.5, β' is varied depending on simulation scenarios defined shortly below. The gyroscope noise is finally added to both turn angles α_i and curvature $\boldsymbol{C}_i$ according to a normal distribution $\mathcal{N}(\mu, \sigma)$ with $\mu = 0.003$ (obtained from Section V-A). We note that the noise margin with simulated magnetometer and travel time is relatively higher than in reality; for instance, the magnetometer error is found to be only around $60°$ for our devices, while in practice drivers rarely exceed 15% (i.e., $\beta = 1.15$) of speed limit (e.g., $75\,\text{mph}$ over the limit $65\,\text{mph}$ in Boston).

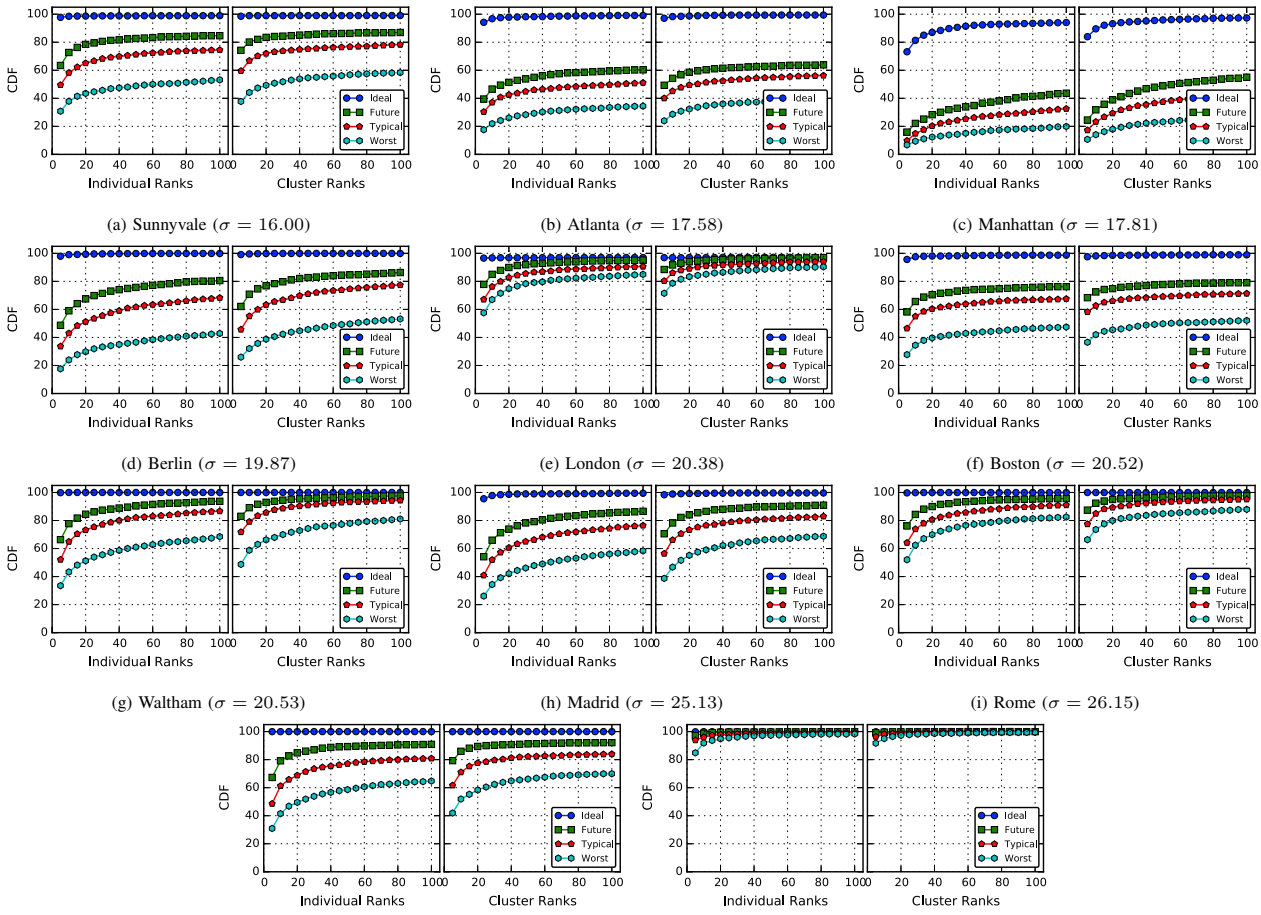
Simulation Scenarios: To understand the attack performance under various environments, our simulation evaluation is performed and reported for different scenarios, in which several noise parameters are adjusted from the above settings.

- **Ideal**: noise-free scenario (upper bound performance).
- **Worst**: $\sigma = 10, \beta' = 0.1$. In this scenario, we consider heavy traffic and old smartphones with less accuracy.
- **Typical**: $\sigma = 8, \beta' = 0.5$. In this scenario, we consider moderate traffic and current smartphones. Note that, $\sigma = 8$ is slightly higher than the experimental value $\sigma = 7.54$, implying a slightly harder attack.
- **Future**: $\sigma = 6, \beta' = 0.5$. In this scenario, we consider moderate traffic and future smartphones equipped with more accurate sensors as MEMs technology progresses.

D. Simulation Results

We evaluate the potential of the attack for all cities in Table III using the 4 different scenarios specified in Section V-C. In total, there are 44 test cases and for each, we generate a new set of 2000 simulated routes. We use the same scoring weights $\omega_A = 2.5, \omega_T = 0.1, \omega_C = 2.5$ for every city. These weights are selected as they are relatively good for all cities, and our main simulation goal is to evaluate the attack using the same configuration for different city profiles. Other parameters used for the attack are specified in Table I. The attack outcome is evaluated according to both *individual rank* and *cluster rank*. For the latter metric, we choose the proximity threshold $\Delta = 500\,\text{meters}$, which typically covers a few house blocks or apartment buildings.

Figure 7 shows the *Cumulative Distribution Function (CDF)* of individual and cluster ranks (i.e., P^{idv} and P^{clt}) produced by the attack. For the *Typical* scenario, we see that the system is able to find more than 50% (resp. 60%) of exact routes (resp. clusters of routes) in the top 10 results for all cities except for Atlanta, Berlin, and Manhattan. Even in the *Worst* scenario, more than 35% (resp. 40%) of exact routes (resp. clusters)



(a) Sunnyvale ($\sigma = 16.00$) (b) Atlanta ($\sigma = 17.58$) (c) Manhattan ($\sigma = 17.81$)

(d) Berlin ($\sigma = 19.87$) (e) London ($\sigma = 20.38$) (f) Boston ($\sigma = 20.52$)

(g) Waltham ($\sigma = 20.53$) (h) Madrid ($\sigma = 25.13$) (i) Rome ($\sigma = 26.15$)

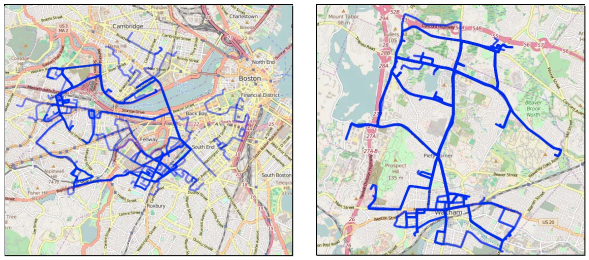
(j) Paris ($\sigma = 26.26$) (k) Concord ($\sigma = 29.58$)

Fig. 7: Attack performance on simulated routes for various cities. Graphs are arranged in ascending order of turn distribution σ.

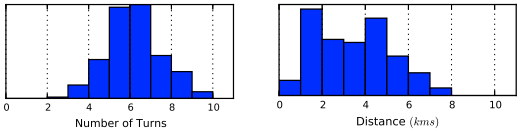
are discovered in the top 10 results. In case of cluster rank, we examine the results in more details (excluded in this paper due to lack of space) and find that each cluster comprises a relatively small set of routes (approximately 1-20 routes per cluster). This explains why cluster ranks are only slightly better than individual ranks.

Among cities having low σ_{turn} (less unique turns) in the top row of Figure 7, Manhattan results in lower ranking than Atlanta and Sunnyvale even when it has a higher σ_{turn} and smaller graph size (lower $|V|$ and $|E|$). This can be attributed to two factors: (1) Manhattan has mostly straight roads reducing the curvature impact on scoring, and (2) most roads are parallel rendering heading filters ineffective. Atlanta and Sunnyvale, on the other hand, have more curvy roads that do not run in parallel. Atlanta has lower ranking than Sunnyvale, because it has a lot more segments and connections that significantly increase the search space and inversely affect the results. Berlin, like others in this group, has more 90° turns and straighter roads, and its reported results are in between Atlanta's and Sunnyvale's.

In the middle and bottom rows of Figure 7, since the cities have high value of σ_{turn}, the turn angle impact on scoring is high (especially very high for Rome, Paris and Concord, cf. Table III). Attack for Concord is most successful, because the high number of curvy roads and unique turns helps diversify the route's score, and the small graph size significantly reduces the search space. Paris creates somewhat more difficulty for the adversary than both Rome and London even though it has a higher σ_{turn} and lower $|V|$ and $|E|$. This can be explained by the fact that many internal roads in Paris are straight, reducing the curvature impact on scoring. Madrid, like Paris, also has a lot of straight roads, but due to high $|V|$, it results in slightly lower rankings than Paris. The attack seems easy in Rome and London thanks to the high variations in curvature in both cities. Boston has lower ranking than London even when it is similar in turn distributions and graph size. This is mainly because Boston has several grid-like residential areas such as South Boston and Back Bay that create much confusions for routes passing through such areas. Waltham's road structure is very similar to Boston's except that it is much smaller, which becomes the main factor for increasing the attack performance.



(a) Traveled routes in Boston (b) Traveled routes in Waltham



(c) Turns Distribution (d) Distance Distribution

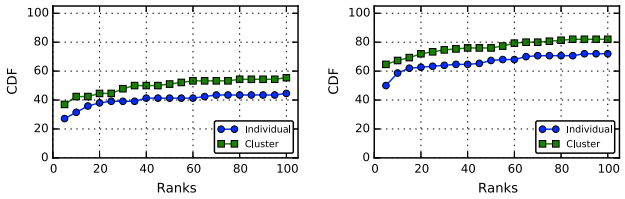
Fig. 8: Real experiments statistics: (a-b): GPS traces of all traveled routes; (c-d): Turn and Distance distributions for all routes combined.

E. Real Driving Experimental Results

To measure the attack efficiency in actuality, we carried out real driving experiments in Boston and Waltham. For each city, over 70 different routes were taken. These routes emulated mostly realistic scenarios, e.g., traveling between residential areas, shopping stores, office, or city centers. There were 4 drivers participating in the experiments, who were instructed to (1) place the phone anywhere but in fixed position during collection, (2) idle at least 10 seconds before driving, and (3) drive within the city limit and take a minimum of 3 turns on their routes. These requirements allow us to model typical realistic scenarios, in which the victim, after putting her phone in a stable position (cup holder, mount, etc.), may take a few seconds before starting to drive to check for her safety, such as tying her seatbelt, and adjusting the seat, mirrors, or lights. In this initial study, we did not consider situations when the vehicle starts by reversing. We emphasize that given the limited resources, we aimed to obtain a data-set as diverse as possible, therefore we did not request the drivers to repeat the same routes. Still, all routes consist of total $\approx 980\,\mathrm{km}$, including driving in both peak and off-peak hours. Scoring weights $(\omega_A, \omega_T, \omega_C)$ were fine-tuned based on road characteristics: $(2.5, 0.1, 3)$ for Boston, and $(2.25, 0.1, 2.5)$ for Waltham. Both cities (especially Boston) have more unique curves than turns attributing to the higher ω_C. Waltham has typically less traffic than Boston, therefore, we assign lower ω_A and ω_C to increase impact of ω_T.

Figure 8 shows the distribution of turns made on all routes, total traveled distances, and GPS traces. Note that GPS is used only for ground truth comparison. The shortest route taken was $\approx 0.75\,\mathrm{km}$, the longest $\approx 7.25\,\mathrm{km}$. Additionally, 4 more routes were taken to consider scenarios of driving in a circle, taking many turns (≥ 20), and traveling longer distances ($\geq 20\,\mathrm{km}$). These routes were also used to test the system's stability.

Figure 9 shows the attack in terms of both individual and cluster ranks. The reported results are a worst-case scenario



(a) Boston (b) Waltham

Fig. 9: Attack performance on real driving experiments.

with no a priori information on the user's routes. We see that roughly 50% of routes in Waltham and roughly 30% of routes in Boston are in the top 5 individual ranks. When top 1 is considered (i.e., exact route), the success probability reduces to 38% for Waltham, and 13% for Boston, respectively. The gap between individual and cluster ranks is about 10%, which is almost similar to simulations. The number of routes per cluster is around 2-3 for most top ranked clusters. The performance for both cities lies between the simulation's *Typical* and *Worst* scenarios. However, the results for Boston are closer to the *Worst* scenario, while Waltham's are much like the *Typical*. The main reason for this difference is the traffic in Boston that caused more variations in estimating non-idle time than Waltham. The small gap between real and simulation results shows that our simulation framework may serve as an effective model for studying the attack in a larger scale where experiments are limited.

F. Feasibility of the Attack

The colluding server was setup inside a Linux Virtual Machine (VM) on a Dell PowerEdge R710 server. The VM has 2x4 cores with 16 threads running at 2.93 GHz, with 32 GB of RAM. The attack is written in Python and run using *PyPy*, a fast Python JIT compiler. We measure the feasibility of attack in terms of execution time for processing data and searching routes. The search time specifically depends on the route length and graph size.

Data Processing: The longest experimental route (approximately 45 minutes) in our set requires $\approx 1.4\,\mathrm{s}$ to process the sensor data and produce a trace of heading, turns, curves, and timestamps, while an average route takes $0.1 - 0.2\mathrm{s}$.

Route Search: For the largest city in our set, Atlanta, the search for each route takes about 2.2 s. For Concord, the smallest one, each route takes about 0.4 s. We use 15 threads to parallelize the search on multiple routes, and 1 remaining thread for control and management. The simulation of 88000 routes takes $\approx 21\,\mathrm{hours}$ to complete ($\approx 0.85\,\mathrm{s}$ per route).

While not a formal benchmark, it still implies that the attack is practical (e.g., less than 4 seconds for a long route in Atlanta). With adequate resources, an adversary can handle millions of routes fairly quickly.

G. Impact of Algorithm Parameters and Assumptions

In this subsection, we study the attack performance under various conditions such as when calibration is not performed,

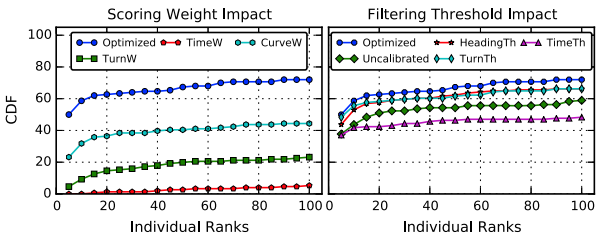


Fig. 10: Impact of parameters and calibration on Waltham experiments.

or the algorithm parameters are not carefully selected. We use the real driving experiments from Waltham in this investigation and re-perform evaluation changing one parameter at a time to better understand the impact of individual parameters. For comparison, the performance achieved with parameters optimized in Section V-E is referred to as the *Optimized* test case (cf. Table IV and Figure 10).

TABLE IV: Test cases for impact of parameters and calibration.

Test case	Parameter settings
Optimized	As in Section V-E
TurnW	As *Optimized*, except $\omega_T = 0, \omega_C = 0$
TimeW	As *Optimized*, except $\omega_A = 0, \omega_C = 0$
CurveW	As *Optimized*, except $\omega_A = 0, \omega_T = 0$
HeadingTh	As *Optimized*, except $\phi_m = 30^\circ$
TimeTh	As *Optimized*, except $\beta = 1.0$
TurnTh	As *Optimized*, except $\gamma = 20^\circ$
Uncalibrated	*Optimized* without calibration

Scoring Weights: To justify the impact of each scoring weight, we ignore the other weights by setting them to zero in the scoring function, cf. Equation (2). Figure 10 shows that curvature is the most useful factor for success probability, while travel time only slightly increases the performance. This is not only applicable to Waltham, but also to cities that have numerous roads with unique curvature. The travel time varies more due to external factors such as traffic or unknown speed, making it less impactful. Hence, weights must be selected based on the target area to maximize the attack success.

Filtering Thresholds: Filtering allows quick elimination of bad routes, however, it can also falsely remove good routes. To see the performance impact from over-filtering, we reduce the thresholds for turn, heading, and time as specified in Table IV. We observe several interesting facts from Figure 10. First, tighter heading and turn thresholds only slightly decrease performance, which implies that the sensors have small noise margin. Therefore, stricter rules can be applied to speed up the search if execution time is of high priority. On the other hand, stricter travel time threshold results in considerably lower performance, which reveals that over-speeding is a common practice in real driving.

Calibration: Recall that for the real driving experiments, drivers were instructed to stay idle for at least $10\,\mathrm{s}$ before driving. While this allows for easy calibration, an alternative calibration method can be used, in which we first detect idle time (based on accelerometer) and then compute the gyroscope drift during that state. This enables calibration whenever the vehicle is idle (e.g., stopping at traffic lights) and the parking assumption can be relaxed. In Figure 10, however, we show that even without calibration, the performance does not decrease significantly. In fact, the individual ranks drop only by $10\% - 15\%$ in comparison with *Optimized* which implies calibration is an optional rather than a required operation.

Route Equiprobability: We emphasize that the reported results in this work are based on the worst-case assumption of no a priori information of the victim's travel history. Knowing the starting or ending point would improve the accuracy. On the other hand, such travel history information can be built up over time to improve the attack. We plan to study such extensions in future work.

Fixed Position: Our assumption of fixed phone position is realistic in various scenarios (e.g., many states in the USA prohibit hand-held use). However, if users interact with their phones, we describe an idea (we did not implement it) that can help increase possibility of distinguishing between a real turn and a change in phone's orientation due to user interaction. Our idea is based on the observation that human interaction (e.g., touching, holding in hand) induces high variations in sensor data in *all* 3 dimensions for a short duration. Note that if the variations are low, the attack is barely affected and there is no need for detection. When such events are detected, we simply ignore the sensor data, and later, re-perform rotation to reflect the phone's new position. In practice, however, more complex algorithms would be required to deal with noise and unknown human behaviors, which can be studied in the future.

Detection of Vehicle Start: In this work, we assume that it is feasible to determine when a user enters their vehicle. This can be done a posteriori with the app continuously recording (and storing a window of few minutes) and using techniques similar to Android step detection [14] to detect when the user stops walking and steps into the vehicle.

Reversing: In this work, we assume only forward motion of drivers. While reversing can be detected using the accelerometer, a more complex problem may arise when turning is performed at the same time as reversing (e.g., making a U-turn or pulling out of a parallel parking spot). This increases the search space, and our algorithm would have to be extended to roll back to previous states along all candidate routes.

Known City: Knowledge about the victim's city can be obtained in several ways. For instance, the app can detect the city based on IP address when the victim is connected to Wi-Fi or cellular networks. Additionally, an adversary with access to the victim's social network can find the victim's city, frequently visited places, and even route patterns. A powerful adversary can also run the attack on multiple geographic areas in parallel. These techniques can be combined together to devise an effective attack.

VI. Countermeasures

Access to motion sensors is granted without permissions or any notifications to the user as they are still underestimated as a source of privacy leakage. Several detection and protection

mechanisms can be used to mitigate this attack, for example, when installing an app, permissions to the sensors must be explicitly requested by the app. Also, like location, a notification (with app name) should be displayed to the user when sensors are accessed. To deal with attacks that also require access to sensors for other activity, more complex mechanisms are required such as closely monitoring the Internet traffic and energy consumption, or generating adequate artificial noise in the data before providing it to the app. While the above make the attack more difficult, effective protection mechanisms are beyond the scope of this work and considered an open problem. The mechanisms discussed should be implemented in the OS to ensure prevention globally, however, they can also be implemented using dynamic instrumentation tools like *ddi* [15] or recently, using app sandboxing tools like Boxify [16].

VII. Related Work

Smartphone privacy attacks have recently attracted significant interest. They typically fall into one of the three categories. Some attacks use cellular signals, GPS, Wi-Fi, Bluetooth, NFC, Wi-Fi Direct and other radio communications mechanisms (henceforth, we will refer to them as wireless location support systems or WLSS). Sensor centric attacks use native smartphone sensors such as the gyroscope, accelerometer and magnetometer as data sources with no WLSS involvement. The hybrid cases are where the victim makes available, albeit to a limited community and on a limited basis, her location. These attacks use WLSS and sensor data integration. Fawaz et al. [1] reported that 85% of surveyed users expressed concern about conveying location information. Some countermeasures emerged in the form of location privacy protection mechanisms or LPPMs. These services obfuscate location information by modifying precision or performing location transformation. As they attempt to deflect WLSS centric threats, LPPMs remain ineffective in mitigating our threat. As of this publication and to our knowledge, no service exists to address our proposed threat.

A. WLSS Based Attacks

WLSS based attacks typically require either apps installed on a smartphone with appropriate permissions or significant presence within the network infrastructure. We do not address the former as the user consciously forfeited some degree of position anonymity. The infrastructure attack involves taking over some of the infrastructure components or injecting signature probes and are subject to detection by conventional means (i.e. IDS or IPS solutions). WLSS attacks provide accuracies near 90% when attempting path identification.

In Qian et al. [17], the authors attempt targeted cellular DoS attacks. Of relevance is identifying the specific smartphone location as a precursor to the attack. The attack seeks to gain IP identification using techniques like active probes and fingerprints. By measuring promotion delay and Round Trip Time (RTT), cellphone localization is achieved with granularity to the Location Area Code (LAC)/Radio Network Controller (RNC) range. Its effectiveness is limited due to measurement tuning needs and RNC sharing observed among smaller cities. This expands the geographical area cross section from which to identify the user. As with WLSS attacks, introducing network probes may enable detection.

Kune et al. [18] describe location determination via leakage from lower level Global System for Mobile Communications (GSM) broadcasts, in particular, a victim's temporary identifier. For this attack to work, the attacker must initiate a Paging Control Channel (PCCH) paging request targeting the victim and passively listen for broadcast PCCH messages. Although relatively simple, it places the attacker as an active network participant which risks detection. It also requires a priori knowledge of the victim's telephone number. Position resolution was observed to within 1 km^2.

Bindschaedler et al. [19] use a group of 802.11 access ports to eavesdrop on proximate target smartphones in order to evaluate mixing zone effectiveness. Data collection includes device time, location, device identifier and content. Although victims may attempt to hide via a mix-zone network where MAC addresses are synchronously changing (assuming sufficient group membership), tracking can be achieved. This attack requires collusion of multiple APs and Wi-Fi or equivalent communications mechanisms. This may be impractical to set up exclusive of the most sophisticated attackers.

B. Hybrid Attacks

There are a number of works [20–26] that combine WLSS data with motion/inertial sensors to infer user location, mode of transit, orientation and behavior. Of those surveyed, we find best case accuracies near 80%. Although positional accuracy benefits offered by these mechanisms are interesting, these attacks generally require obtaining a 'fix' via WLSS functionality prior to leveraging sensor data. This exposes the attacker to WLSS discovery mechanisms.

Zhang et al. [27] developed the SensTrack system which identifies turning points using a smartphone's accelerometer to determine speed, distance, and orientation. Additionally, they use sensors with adaptive Wi-Fi and GPS switching to address location contexts where GPS is less effective (i.e. indoor locations). Their system achieved prediction errors of nominally 3.128 meters versus 5 for good GPS signal strength. This approach assumes some location predetermination using GPS for initial reference position. Furthermore, the short distances within a building do not offer the challenges one realizes in the spatial-temporal context of driving a vehicle.

C. Sensor Only Attacks

The following attacks are most representative of our approach as they rely entirely on zero-permission sensor sources.

Han et al. [28] suggested a method of location inference using the accelerometer and magnetometer. Leveraging a probabilistic dead reckoning method called Probabilistic Inertial Navigation (ProbIN), they mapped probability of displacement to probability of motion. Training data associates sensor data with map truth. Resolution is observed approaching 200 meters, the length of a typical city block. Their small sample

size limited the experimental path length range to between 1 km and 9.7 km. Although claiming better accuracy than achievable using Wi-Fi or cellular techniques, their approach greatly depends on acquiring training data which may present a resource challenge (i.e. time and labor) in large scale scenarios.

In Nawaz et al. [29], the authors demonstrate that a smartphone's accelerometer and gyroscope can be used to identify 'significant' journeys independent of phone orientation and traffic. This is because gyroscope signatures obtained from multiple journeys of the same route exhibit similar patterns that differ only in amplitude and time compression or expansion. They apply Dynamic Time Warping to calculate the distance between various journeys and use a k-medoids clustering approach to cluster similar routes together. A route is labeled as significant if it is traveled more times than a predefined threshold. They test this technique for two cities using 43 real driving experiments and showed that the routes were accurately clustered in 8 clusters defined for the two cities. Grid road networks are addressed in a different manner. Here, they depend on turn count as a uniqueness metric and suggest that their technique is effective for reasonably long routes because such routes exhibit a unique sequence of turns even when individual turns are similar.

In Zhou et al. [30], the authors describe a novel technique that analyzes verbal directions provided by a GPS based navigation app. Using a second zero-permissions app, they measure speaker on/off times controlled by the navigation app. The attacker can infer which course a driver took due to the duration of these audible driving instructions. Permission for speaker usage is not required as of this writing. Associating talk time to an off-board synthesized instruction driving set yields a 30% false positive rate over a small sample size (7 out of 10 correct). This approach requires the use of a voice enabled navigation system. Furthermore, it assumes that the navigation app is trustworthy.

Michalevsky et al. [10] introduce a power based scheme that distinguishes a user route from a set of possible routes in real-time. Furthermore, they attempt to infer new routes by constructing projected route power profiles that are aggregated from shorter, known segment power profiles, all using 3G networks. With a 'modest' number of applications running, they achieve accurate results in 2/3 of the scenarios while the results degrade to an accuracy of 1/5 with additional active applications such as Facebook and Skype. In addition, they are limited by the need to provide data to the learning machine which itself limits scalability in obtaining training data.

D. Behavior Analysis

This research area involves determining user modality from smartphone sensors. For example, ergonomic/activity identification is discussed in [31]. The authors use learned data from walking, jogging, climbing stairs, sitting, and standing to ascertain user activity. They identified and collected data for 43 features from a 29 person sample set. Raw data was evaluated using the WEKA data mining tool suite to develop decision tree, logistic and regression and multilayer neural network models. Excluding motions associated with moving up and down stairs, the method can identify activity nearly 90% of the time. Although of a single modality and reasonably well suited for human activity identification, is has limited ability to ascertain paths with much less start and stop points.

Lee and Mase [32] studied the feasibility of detecting user behavior such as sitting, standing, walking on level ground, going up or down a stairway as well as determining the number of steps taken to infer a person's location in an indoor environment. They developed a system using the accelerometer and gyroscope sensors to measure the forward and upward acceleration and angle of the user's legs. In addition, the compass is used to determine the direction of movement. The phone is mounted on different body locations and a dead-reckoning method is applied to estimate the user's physical location. The authors show that their system efficiently calculated the number of steps and location for eight individuals, using a predefined database of selected locations in an office environment. They claim a high recognition ratio of 91.8% for ten unique location transitions.

E. Other Works of Interest

Two additional works are noteworthy. They include a pattern matching/machine vision approach to path traversal tracking and a framework to measure the effectiveness of the attack.

In terms of matching shapes, patterns and contours, there are numerous examples in the literature. We identify one here for this discussion. Kupeev et al. [33] decomposed shape contours in terms of segments for purposes of determining similarity of contours. They were able to analyze 24 shape distances with 32 unique quantized rotation angles against one another. The error rate appeared to be less than 10%. Of importance is the limited use of this technique observed in the location privacy space. This approach's weaknesses are similar to other contour matching solutions in that the subtle differences in road contours may not be distinguishable between similar yet geographically separate roads.

In Shokri et al. [34], the authors suggest a framework for scoring location privacy protection mechanisms. Here, they define a triad taxonomy of accuracy, certainty and correctness where the later represents the metric that determines the privacy of user. To our knowledge, this is the first significant attempt at establishing an evaluation framework. Although not utilized in this work, it provides a foundation for evaluating in the future, our results when compared with truth.

VIII. Conclusion

We modeled the problem of tracking vehicular users as the problem of identifying the most likely route on a graph derived from the city's roads public database. The performance results of our algorithms, both simulations and experimental, indicate that in most cities a significant number of users are vulnerable to tracking by seemingly innocuous applications that do not request permissions to any sensitive information. We believe that this calls for rigorous methods and tools to mitigate side-channel attacks making use of mobile phones sensors.

REFERENCES

[1] K. Fawaz and K. G. Shin, "Location privacy protection for smartphone users," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. ACM, 2014, pp. 239–250.

[2] Senate Judiciary Committee, "S.2171 - Location Privacy Protection Act of 2014," https://www.congress.gov/bill/113th-congress/senate-bill/2171, 2014.

[3] Euclid Analytics, "Privacy statement," http://euclidanalytics.com/privacy/statement/, accessed: May, 2015.

[4] S. Datoo, "This recycling bin is following you," http://qz.com/112873/this-recycling-bin-is-following-you/, Quartz, August 2013, accessed: May, 2015.

[5] Z. M. Seward and S. Datoo, "City of london halts recycling bins tracking phones of passers-by," http://qz.com/114174/city-of-london-halts-recycling-bins-tracking-phones-of-passers-by/, Quartz, August 2013, accessed: May, 2015.

[6] L. Hutchinson, "iOS 8 to stymie trackers and marketers with mac address randomization," http://arstechnica.com/apple/2014/06/ios8-to-stymie-trackers-and-marketers-with-mac-address-randomization/, June 2014, accessed: May, 2015.

[7] A. Cassola, W. Robertson, E. Kirda, and G. Noubir, "A practical, targeted, and stealthy attack against wpa enterprise authentication," in *Proceedings of the 20th Annual Network & Distributed System Security Symposium, NDSS'13*, 2013.

[8] FTC, "Android flashlight app developer settles FTC charges it deceived consumers," https://www.ftc.gov/news-events/press-releases/2013/12/android-flashlight-app-developer-settles-ftc-charges-it-deceived, December 2013, accessed: November, 2015.

[9] S. Narain, A. Sanatinia, and G. Noubir, "Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning," in *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*, 2014.

[10] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis," in *Proceedings of the 24th USENIX Conference on Security Symposium*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 785–800.

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

[12] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed., 2004.

[13] Nokia, "HERE Map," https://maps.here.com/.

[14] Android SDK, "Step detection," http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_STEP_DETECTOR.

[15] Collin R. Mulliner, "Dynamic Dalvik Instrumentation Framework for Android," https://github.com/crmulliner/ddi.

[16] M. Backes, S. Bugiel, C. Hammer, O. Schranz, and P. von Styp-Rekowsky, "Boxify: Full-fledged app sandboxing for stock android," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015, pp. 691–706.

[17] Z. Qian, Z. Wang, Q. Xu, Z. M. Mao, M. Zhang, and Y.-M. Wang, "You can run, but you can't hide: Exposing network location for targeted DoS attacks in cellular networks," in *Proceedings of the 19th Annual Network & Distributed System Security Symposium*, Feb. 2012.

[18] D. F. Kune, J. Koelndorfer, N. Hopper, and Y. Kim, "Location leaks over the GSM air interface," in *Proceedings of the 19th Annual Network & Distributed System Security Symposium*, Feb. 2012.

[19] L. Bindschaedler, M. Jadliwala, I. Bilogrevic, I. Aad, P. Ginzboorg, V. Niemi, and J.-P. Hubaux, "Track me if you can: On the effectiveness of context-based identifier changes in deployed mobile networks." in *NDSS*. The Internet Society, 2012.

[20] N. Marmasse and C. Schmandt, "A user-centered location model," *Personal and Ubiquitous Computing*, vol. 6, no. 5-6, pp. 318–321, 2002.

[21] D. Patterson, L. Liao, D. Fox, and H. Kautz, "Inferring high-level behavior from low-level sensors," in *UbiComp 2003: Ubiquitous Computing*, ser. Lecture Notes in Computer Science, A. Dey, A. Schmidt, and J. McCarthy, Eds. Springer Berlin Heidelberg, 2003, vol. 2864, pp. 73–89.

[22] D. Ashbrook and T. Starner, "Using GPS to learn significant locations and predict movement across multiple users," *Personal Ubiquitous Comput.*, vol. 7, no. 5, pp. 275–286, Oct. 2003.

[23] D. Patterson, L. Liao, K. Gajos, M. Collier, N. Livic, K. Olson, S. Wang, D. Fox, and H. Kautz, "Opportunity knocks: A system to provide cognitive assistance with transportation services," in *UbiComp 2004: Ubiquitous Computing*, ser. Lecture Notes in Computer Science, N. Davies, E. Mynatt, and I. Siio, Eds. Springer Berlin Heidelberg, 2004, vol. 3205, pp. 433–450.

[24] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello, "Extracting places from traces of locations," in *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, ser. WMASH '04. ACM, 2004, pp. 110–118.

[25] K. Laasonen, M. Raento, and H. Toivonen, "Adaptive on-device location recognition," in *Pervasive Computing*, ser. Lecture Notes in Computer Science, A. Ferscha and F. Mattern, Eds. Springer Berlin Heidelberg, 2004, vol. 3001, pp. 287–304.

[26] L. Liao, D. J. Patterson, D. Fox, and H. Kautz, "Learning and inferring transportation routines," *Artificial Intelligence*, vol. 171, no. 5-6, pp. 311–331, Apr. 2007.

[27] L. Zhang, J. Liu, H. Jiang, and Y. Guan, "Senstrack: Energy-efficient location tracking with smartphone sensors," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3775–3784, Oct 2013.

[28] J. Han, E. Owusu, L. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, Jan 2012, pp. 1–9.

[29] S. Nawaz and C. Mascolo, "Mining users' significant driving routes with low-power sensors," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '14. ACM, 2014, pp. 236–250.

[30] X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt, "Identity, location, disease and more: Inferring your secrets from android public resources," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. ACM, 2013, pp. 1017–1028.

[31] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *SIGKDD Explor. Newsl.*, vol. 12, no. 2, pp. 74–82, Mar. 2011.

[32] S.-W. Lee and K. Mase, "Activity and location recognition using wearable sensors," *Pervasive Computing, IEEE*, vol. 1, no. 3, pp. 24–32, July 2002.

[33] K. Kupeev and H. Wolfson, "On shape similarity," in *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1, Oct 1994, pp. 227–231 vol.1.

[34] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *Security and Privacy (SP), 2011 IEEE Symposium on*, May 2011, pp. 247–262.