PowerSpy: Location Tracking using Mobile Device Power Analysis

Yan Michalevsky, Dan Boneh and Aaron Schulman
Computer Science Department Stanford University
yanm2,dabo@cs.stanford.edu
aschulm@stanford.edu

Gabi Nakibly
National Research and Simulation Center
Rafael Ltd.
gabin@rafael.co.il

***Abstract*—Modern mobile platforms like Android enable applications to read aggregate power usage on the phone. This information is considered harmless and reading it requires no user permission or notification. We show that by simply reading the phone's aggregate power consumption over a period of a few minutes an application can learn information about the user's location. Aggregate phone power consumption data is extremely noisy due to the multitude of components and applications simultaneously consuming power. Nevertheless, we show that by using machine learning techniques, the phone's location can be inferred. We discuss several ways in which this privacy leak can be remedied.**

I. Introduction

Our smartphones are always within reach and their location is mostly the same as our location. In effect, tracking the location of a smartphone is practically the same as tracking the location of its owner. Since users generally prefer that their location not be tracked by arbitrary 3rd parties, all mobile platforms consider the device's location as sensitive information and go to considerable lengths to protect it: applications need explicit user permission to access the phone's GPS and even reading coarse location data based on cellular and WiFi connectivity requires explicit user permission.

In this work we show that applications that want access to location data can bypass all these restrictions and covertly learn the phone's location. They can do so by analyzing the phone's power consumption over a period of time. Our work is based on the observation that the phone's location significantly affects the power consumed by the phone's cellular radio. The power consumption is affected both by the distance to the cellular base station to which the phone is currently attached (free-space path loss) and by obstacles, such as buildings and trees, between them (shadowing). The closer the phone is to the base station and the fewer obstacles between them, the less power the phone will consume. The strength of the cellular signal is a major factor affecting the power used by the cellular radio [1]. Moreover, the cellular radio is one of the most dominant power consumers on the phone [2].

Suppose an attacker measures in advance the power profile consumed by a phone as it moves along a set of known routes or in a predetermined area such as a city. We show that this enables the attacker to infer the target phone's location over those routes or areas by simply analyzing the target phone's power consumption over a period of time. This can be done with no knowledge of the base stations to which the phone is attached, as long as the attacker knows the general area in which the victim moves.

A major obstacle to our approach is that power is consumed simultaneously by many components and applications on the phone in addition to the cellular radio. A user may launch applications, listen to music, turn the screen on and off, receive a phone call, and so on. All this activity affects the phone's power consumption and results in a very noisy approximation of the cellular radio's power usage. Moreover, the cellular radio's power consumption itself depends on the phone's activity, as well as the distance to the base-station: during a voice call or data transmission the cellular radio consumes more power than when it is idle. All of these factors contribute to the phone's power consumption variability and add noise to the attacker's view of the power consumption (note that the attacker cannot tell the level of activity on the cellular radio). Nevertheless, using machine learning, we show that measuring the phone's aggregate power consumption over time completely reveals the phone's location and movement. Intuitively, the reason why all this noise does not mislead our algorithms is that the noise is not correlated with the phone's location. Therefore, a sufficiently long power measurement (several minutes) enables the learning algorithm to "see" through the noise.

In this work we use a machine learning based approach to identify the routes taken by the victim based on previously collected power consumption data. We study three types of user tracking goals:

1) **Route distinguishability:** Can an attacker tell which out of several possible routes the user is taking?
2) **Real-time motion tracking:** Assuming the user is taking a certain known route, can an attacker identify her location along the route and track the device's position on the route in real-time?
3) **New route inference:** Can an attacker identify an arbitrary (long) route taken by the user in a given area, assuming the attacker has previously measured the power profile of every short road segment in the area?

We emphasize that our approach is based on measuring the phone's aggregate power consumption and nothing else. We do not read the phone's signal strength since that data is protected on Android and iOS devices and reading it requires user per-

mission. In contrast, reading the phone's power consumption requires no special permissions and we therefore focus all our efforts on what can be learned from this data. On Android devices reading the phone's aggregate power consumption is done by repeatedly reading the following two files:

```
/sys/class/power_supply/battery/voltage_now
/sys/class/power_supply/battery/current_now
```

Over a hundred applications in the Play Store access these files. While most of these simply monitor battery usage, our work shows that all of them can also easily track the user's location.

The rest of the paper is organized as follows: We start with defining the threat model. Then we provide technical background about signal strength and power consumption, and relate it to our method. We follow with stating the underlying assumptions behind our research. The technical details of our algorithms are presented in sections V,VI and VII, followed up by presenting the results of their evaluation. We discuss future research directions related to our work, suggest possible defenses against our attack, and finally discuss related work.

II. Threat Models

We assume a malicious application has been installed on the victim's device and runs in the background while the victim is tracked. The malicious application has neither permission to access the GPS, nor other location providers (e.g. cellular or WiFi network). The application has no permission to access the identity of the currently attached or visible cellular base stations or SSID of the WiFi networks.

We only assume permission for network connectivity and access to the power data[1]. These are very common permissions for an application and are unlikely to raise suspicion on the part of the victim. To date there are 179 applications submitted to the Google Play application market that access voltage and current data. We assume most of them are either providing diagnostics or using it to profile the application's power consumption. We do not assume the application can measure the power consumed by the cellular radio, but only the total power consumed by phone. The attacker needs the network connectivity to leak out the power measurements[2], as well as to generate low rate traffic in order to prevent the cellular radio from going into low power state, thereby accentuating the power consumption profile[3].

As noted above, we assume the attacker has prior knowledge of the area or routes through which the victim travels. This knowledge allows the attacker to learn in advance the power consumption profiles of these routes or area. We assume the victim moves by some means of transportation, like a car or a bus, while she is tracked. Our scheme is of no use to locate a victim that stands still.

[1]Available without special permissions on Android.

[2]Network connectivity is necessary for reporting measurements back to the attacker for real-time tracking, but not necessary for inference of past activities. If the data can be leaked in another way, the attacker can learn about routes taken by the device owner in the past.

[3]Although the attack might work even without it.

The focus of this work is on location identification techniques for a limited number of routes or a predetermined area, and it remains to be seen whether it can scale for an attack using a database with a large number of routes, and having no prior knowledge about the victim. We focus on the case of tracking certain users with some sort of daily routine. For example, a mobile device holder can drive to the same place via several possible routes and we want to know which one has she taken. Or there might be several locations the person visits as part of her daily routine, forming several possible routes. This approach could be further scaled if we could use additional information to somewhat limit the pool of possible routes prior to applying our method[4].

III. Background

In this section we provide technical background on the relation between a phone's location and its cellular power consumption. We start with a description of how location has is related to signal strength, then we describe how signal strength is related to power consumption. Finally, we present examples of this phenomenon, and we demonstrate how obtaining access to power measurements could leak information about a phone's location.

A. Location affects signal strength and power consumption

Distance to the base station is the primary factor that determines a phone's signal strength. The reason for this is, for signals propagating in free space, the signal's power loss is proportional to the square of the distance it travels over [3]. Signal strength is not only determined by path loss, it is also affected by objects in the signal path, such as trees and buildings, that attenuate the signal. Finally, signal strength also depends on multi-path interference caused by objects that reflect the radio signal back to the phone through various paths having different lengths.

In wireless communication theory signal strength is often modeled as random variation (e.g., log-normal shadowing [3]) to simulate many different environments[5]. However, in one location signal strength can be fairly consistent as base stations, attenuators, and reflectors are mostly stationary.

A phone's received signal strength to its base station affects its cellular modem power consumption. Namely, phone cellular modems consume less instantaneous power when transmitting and receiving at high signal strength compared to low signal strength. Schulman et. al. [1] observed this phenomenon on several different cellular devices operating on different cellular protocols. They showed that communication at a poor signal location can result in a device power draw that is 50% higher than at a good signal location.

The primary reason for this phenomenon is the phone's power amplifier used for transmission which increases its

[4]For instance, we could use the last WiFi access point the user was connected to prior to driving to understand which are the routes that could be possibly taken.

[5]Parameters of the model can be calibrated to better match a specific environment of interest.

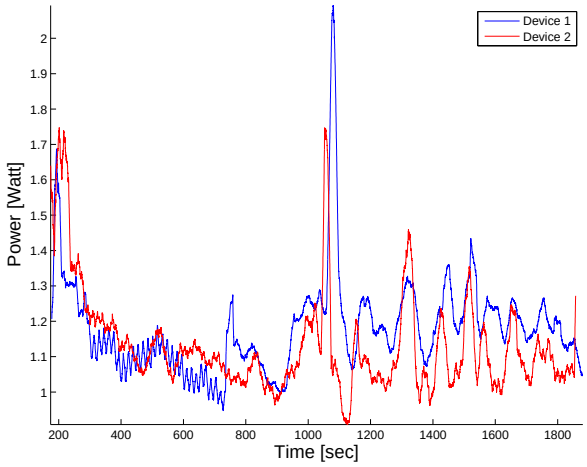


Fig. 2: For two phones of the same model, power variations on the same drive are similar.

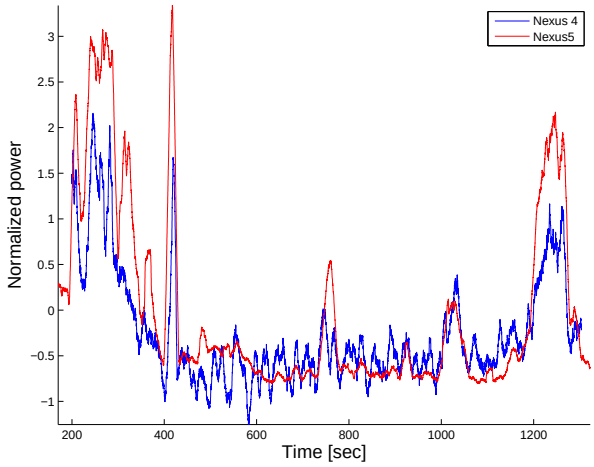


Fig. 3: For two different phone models, power variations on the same drive are similar.

gain as signal strength drops [3]. This effect also occurs when a phone is only receiving packets. The reason for this is cellular protocols which require constant transmission of channel quality and acknowledgments to base stations.

B. Power consumption can reveal location

The following results from driving experiments demonstrate the potential of leaking location from power measurements.

We first demonstrate that signal strength in each location on a drive can be static over the course of several days. We collected signal strength measurements from a smartphone once, and again several days later. In Figure 1 we plot the signal strength observed on these two drives. In this figure it is apparent that (1) the segments of the drive where signal strength is high (green) and low (red) are in the same locations across both days, and (2) that the progression of signal strength along the drive appears to be a unique irregular pattern.

Next, we demonstrate that just like signal strength, power measurements of a smartphone, while it communicates, can reveal a stable, unique pattern for a particular drive. Unlike signal strength, power measurements are less likely to be stable across drives because power depends on how the cellular modem reacts to changing signal strength: a small difference in signal strength between two drives may put the cellular modem in a mode that has a large difference in power consumption. For example, a small difference in signal strength may cause a phone to hand-off to a different cellular base station and stay attached to it for some time (Section III-C).

Figure 2 shows power measurements for two Nexus 4 phones in the same vehicle, transmitting packets over their cellular link, while driving on the same path. The power consumption variations of the Nexus 4 phones are similar, indicating that power measurements can be mostly stable across devices.

Finally, we demonstrate that power measurements could be stable across different models of smartphones. This stability would allow an attacker to obtain a reference power measurement for a drive without using the same phone as the victim's. We recorded power measurements, while transmitting packets over cellular, using two different smartphone models (Nexus 4 and Nexus 5) during the same ride, and we aligned the power samples, according to absolute time.

The results presented in Figure 3 indicate that there is similarity between different models that could allow one model to be used as a reference for another. This experiment serves as a proof of concept: we leave further evaluation of such an attack scenario, where the attacker and victim use different phone models, to future work. In this paper, we assume that the attacker can obtain reference power measurements using the same phone model as the victim.

C. Hysteresis

A phone attaches to the base station having the strongest signal. Therefore, one might expect that the base station to which a phone is attached and the signal strength will be the same in one location. Nonetheless, it is shown in [1] that signal strength can be significantly different at a location based on how the device arrived there, for example, the direction of arrival. This is due to the hysteresis algorithm used to decide when to hand-off to a new base station. A phone hands-off from its base station only when its received signal strength dips below the signal strength from the next base station by more than a given threshold [4]. Thus, two phones that reside in the same location can be attached to two different base stations.

Hysteresis has two implications for determining a victim's location from power measurements: (1) an attacker can only use the same direction of travel as a reference power measurement, and (2) it will complicate inferring new routes from power measurements collected from individual road segments (Section VII).

Fig. 1: Signal strength profiles measured on two different days are stable (The maps were smudged to prevent unblinding and will be put with full details in the final version).

IV. Assumptions and limitations

Exploring the limits of our attack, i.e. establishing the minimal necessary conditions for it to work, is beyond the scope of this work. For this reason, we state the assumptions on which we rely in our methods.

We assume there is enough variability in power consumption along a route to exhibit unique features. Lack of variability may be due to high density of cellular antennas that flatten the signal strength profile. We also assume that enough communication is occurring for the signal strength to have an effect on power consumption. This is a reasonable assumption, since background synchronization of data happens frequently in smartphone devices. Moreover, the driver might be using navigation software or streaming music. However, at this stage, it is difficult to determine how inconsistent phone usage across different rides will affect our attacks.

Identifying which route the user has taken involves understanding which power measurements collected from her mobile device are associated with driving activity. Other works, such as [5], address this question by using data from other sensors that require no permissions to access them (gyroscopes and accelerometers). We do not deal with the details of it in this paper, and assume we are capable of identifying driving activity.

There might be events occurring while driving, such as an incoming phone call, that have a significant effect on power consumption. Figure 4 shows the power profile of a device at rest with a phone call occurring between 50-90 seconds (the part marked in red). The peak immediately after the phone call is caused by using the mobile device to terminate the phone call and turn off the display. We can see that this event appears prominently in the power profile and can develop techniques to cope with such transient effects by identifying and truncating peaks that stand out in the profile. In addition, smoothing the profile by a moving average should mitigate these transient effects.

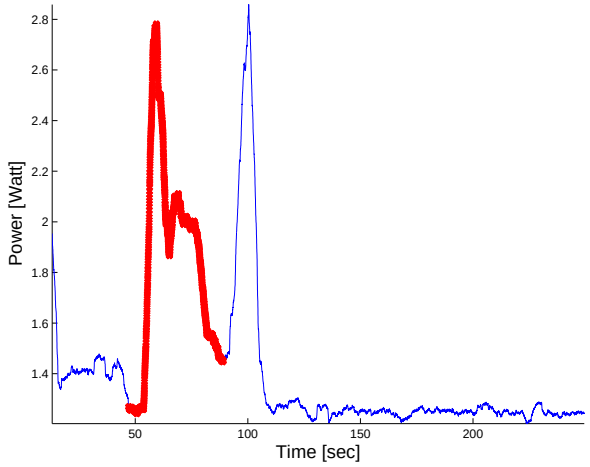


Fig. 4: Power profile with a phone call occurring between 50-90 seconds. Profile region during phone call is marked in red.

V. Route distinguishability

The first problem is one of classification. We have collected power profiles associated with known routes and want to classify new samples based on this training set. Each power profile is basically a time series which needs to be compared to other time series. A score is assigned after each comparison, and based on these scores we select the most likely matching route. Because different rides along the same route can vary in speed at different locations along the ride, and because routes having the same label can vary slightly at certain points (especially before getting to a highway and after exiting it), we need to compare profile features that can vary in time and length and allow for a certain amount of difference. We also have to compensate for different baselines in power consumption due to constant components that depend on the running applications and on differences in device models.

We use a classification method based on Dynamic Time Warping (DTW) [6], an algorithm for measuring similarity between temporal sequences that are misaligned and vary in time or speed. We compute the DTW distance[6] between the new power profile and all reference profiles associated with known routes, selecting the known route that yields the minimal distance. More formally, if the reference profiles are given by sequences $\{X\}_{i=1}^{n}$, and the unclassified profile is given by sequence Y, we choose the route i such that

$$i = \underset{i}{\operatorname{argmin}}\ \mathrm{DTW}(Y, X_i)$$

which is equivalent to 1-NN classification given DTW metric.

Because the profiles might have different baselines and variability, we perform the following normalization for each profile prior to computing the DTW distance: we calculate the mean and subtract it, and divide the result by the standard deviation. We also apply some preprocessing in the form of smoothing the profiles using a moving average (MA) filter in order to reduce noise and obtain the general power consumption trend, and we downsample by a factor of 10 to reduce computational complexity.

VI. Mobile device tracking

In this setting we assume we know that a mobile user is taking a certain route and our objective is to track the mobile device while it is moving. There is no assumption of the starting point along the route, meaning, in probabilistic terms, that our prior on the initial location is uniform. We have reference power profiles collected in advance for that route, and we constantly receive new power measurements from an application installed on the mobile device. Our goal is to localize the device along the route, and continue tracking it as it moves using the real-time observations and training profiles.

A. Tracking using Dynamic Time Warping

This approach is similar to that of route distinguishability, but we use only the measurements collected up to this point, which comprise a sub-sequence of the entire route profile. We use the *Subsequence* DTW algorithm [6], rather than the classic DTW, to search a sub-sequence in a larger sequence, and return a distance measure as well as the corresponding start and end offsets.

We search for the sequence of measurements we have accumulated since the beginning of the drive in all our reference profiles and select the profile that yields the minimal DTW distance. The location estimate corresponds to the location associated with the end offset returned by the algorithm.

B. Improved tracking using a motion model

While the previous approach alone may yield mistakes in location estimation due to a match of the measurements to an incorrect location, we can further improve the estimation by imposing rules based on a sensible motion model. We first need to know when we are "locked" on the target. For this purpose we define a similarity threshold so that if the minimal DTW distance is above this threshold, we are in a *locked* state. Once we are locked on the target, we perform a simple sanity check at each iteration: "Has the target displaced by more than X?"

If the sanity check does not pass we consider the estimation unlikely to be accurate, and simply output the previous estimate as the new estimate location. If the similarity is below the threshold, we switch to an *unlocked* state, and stop performing this sanity check until we are "locked" again. Algorithm 1 presents this logic as pseudocode.

Algorithm 1 Improved tracking using a simple motion model

```
locked ← false                          ▷ Are we locked on the target?
while target moving do
    loc[i], score ← estimateLocation()
    d ← getDistance(loc[i], loc[i − 1])
    if locked and d > MAX_DISP then
        loc[i] ← loc[i − 1]             ▷ Reuse previous estimate
    end if
    if score > THRESHOLD then
        locked ← true
    end if
end while
```

VII. Inference of new routes

In Section V we addressed the problem of identifying the route traversed by the phone, assuming the potential routes are known in advance. This assumption allowed us to train our algorithm specifically for the potential routes. As previously mentioned, there are indeed many real-world scenarios where it is applicable. Nevertheless, we set out to tackle a broader tracking problem in this section, where the future potential routes are not explicitly known. We assume that the area in which the mobile device owner moves is known, however the number of all possible routes in that area may be too large to practically pre-record each one. Such an area can be, for instance, a university campus, a neighborhood, a small town or a highway network.

We address this problem by pre-recording the power profiles of all the road segments within the given area. Each possible route a mobile device may take is a concatenation of some subset of these road segments. Given a power profile of the tracked device, we will reconstruct the unknown route using the reference power profiles corresponding to the road segments. Note that, due to the hysteresis of hand-offs between cellular base stations, a power consumption is not only dependent on the traveled road segment, but also on the previous road segment the device came from.

In the following section we formalize this problem and present our algorithm for solving it.

[6]In fact we compute a normalized DTW distance, as we have to compensate for difference in lengths of different routes - a longer route might yield larger DTW distance despite being more similar to the tested sequence.

A. Formal Model

We formalize the problem described above as a hidden Markov model (HMM) [7]. Let I denote the set of intersections in an area in which we wish to track a mobile device. A road segment is given by an ordered pair of intersections (x, y), defined to be a continuous road between intersection x and intersection y. We denote the set of road segments as R.

We assume that once a device starts to traverse a road segment it does not change the direction of its movement until it reaches the end of the segment. We define a state for each road segment. We say that the tracked device is in state s_{xy} if the device is currently traversing a road segment (x, y), where $x, y \in I$. We denote the route of the tracked device as a (Q, T), where

$$Q = \{q_1 = s_{x_1x_2}, q_2 = s_{x_2x_3}, ...\}$$
$$T = \{t_1, t_2, ...\}$$

For such a route the device has traversed from x_i to x_{i+1} during time interval $[t_{i-1}, t_i]$ ($t_0 = 0, t_{i-1} < t_i \ \forall i > 0$).

Let $A = \{a_{xyz} | \forall x, y, z \in I\}$ be the state transition probability distribution, where

$$a_{xyz} = p\{q_{i+1} = s_{yz} | q_i = s_{xy}\} \quad (1)$$

Note that $a_{xyz} = 0$ if there is no road between intersections x and y or no road between intersections y and z. A traversal of the device over a road segment yields a power consumption profile of length equal to the duration of that movement. We denote a power consumption profile as an observation o. Let B be the probability distribution of yielding a given power profile while the device traversed a given segment. As noted above, due to the hysteresis of hand-offs between cellular base stations, this probability depends on the previous segment the device traversed. Finally, let $\Pi = \{\pi_{xy}\}$ be the initial state distribution, where π_{xy} is the probability that the device initially traversed segment (x, y). If there is no road segment between intersections x and y, then $\pi_{xy} = 0$. In our model we treat this initial state as the state of the device *before* the start of the observed power profile. We need to take this state into account due to the hysteresis effect. Note that an HMM is characterized by A, B, and Π.

The route inference problem is defined as follows. Given an observation of a power profile O over time interval $[0, t_{\max}]$, and given a model A, B and Π, we need to find a route (Q, T) such that $p\{(Q, T)|O\}$ is maximized. In the following we denote the part of O which begins at time t' and ends at time t'' by $O_{[t',t'']}$. Note that $O = O_{[0,t_{\max}]}$. We consider the time interval $[0, t_{\max}]$ as having a discrete resolution of τ.

In the following we describe a method to solve the above problem based on a particle filter. The performance of the algorithm will be examined in the next section.

B. Particle Filter

A particle filter [8] is a method that estimates the state of a HMM at each step based on observations up to that step. The estimation is done using a Monte Carlo approximation where a set of samples (particles) is generated at each step that approximate the probability distribution of the states at the corresponding steps. A comprehensive introduction to particle filters and their relation to general state-space models is provided in [9].

We implement the particle filter as follows. We denote $O^r = \{o^r_{xyz}\}$, where o^r_{xyz} is a power profile prerecorded over segment (y, z) while the segment (x, y) had been traversed just before it. We use a discrete time resolution $\tau = 3$ seconds. We denote $\Delta^{yz}_{\min}$ and $\Delta^{yz}_{\max}$ to be the minimum and maximum time durations to traverse road segment (y, z), respectively. We assume these bounds can be derived from prerecordings of the segments. At each iteration i we have a sample set of N routes $P_i = \{(Q, T)\}$. The initial set of routes P_0 are chosen according to Π. At each step, we execute the following algorithm:

Algorithm 2 Particle filter for new routes estimation

for all route p in P **do**
  $t_{\text{end}} \leftarrow$ end time of p
  $(x, y) \leftarrow$ last segment of p
  $z \leftarrow$ next intersection to traverse (distributed by A)
  $W_p \leftarrow \min_{\substack{t \in [\Delta^{yz}_{\min}, \Delta^{yz}_{\max}] \\ o^r_{xyz} \in O^r_{xyz}}} \left\{\text{DTW}(O_{[t_{\text{end}}, t_{\text{end}}+t]}, o^r_{xyz})\right\}$
  $p \leftarrow p||(y, z)$
  Update the end time of p
end for
Resample P according to the weights W_p

At each iteration, we append a new segment, chosen according to the prior A, to each possible route (represented by a particle). Then, the traversal time of the new segment is chosen so that it will have a minimal DTW distance to the respective time interval of the tracked power profile. We take this minimal distance as the weight of the new route. After normalizing the weights of all routes, a resampling phase takes place. N routes are chosen from the existing set of routes according to the particle weights distribution[7]. The new resampled set of routes is the input to the next iteration of the particle filter. The total number of iterations should not exceed an upper bound on the number of segments that the tracked device can traverse. Note however that a route may exhaust the examined power profile before the last iteration (namely, the end time of that route reached $t_{\max}$). In such a case we do not update the route in all subsequent iterations (this case is not described in Algorithm 2 to facilitate fluency of exposition).

Before calculating the DTW distance of a pair of power profiles the profiles are preprocessed to remove as much noise as possible. We first normalize the power profile by subtracting its mean and dividing by the standard deviation of all values included in that profile. Then, we zeroed out all power values below a threshold percentile. This last step allowed us to

[7]Note that the resampling of the new routes can have repetitions. Namely, the same route can be chosen more than one time

focus only on the peaks in power consumption where the radio's power consumption is dominant while ignoring the lower power values for which the radio's power has a lesser effect. The percentile threshold we use in this paper is 90%.

C. Choosing the best route

Upon its completion, the particle filter outputs a set of N routes of various lengths. Let us denote this set by P_{final}. This set exhibit an estimate of the distribution of routes given the power profile of the tracked device. To select the best estimate route the simple approach is to choose the route that appears the most number of times in P_{final} as it has the highest probability to occur. Nonetheless, since a route is composed of multiple segments chosen at separate steps, at each step the weight of a route is determined solely based on the last segment added to the route. Therefore, in P_{final} there is a bias in favor of routes ending with segments that were given higher weights, while the weights of the initial segments have a diminishing effect on the route distribution with every new iteration.

To counter this bias, we choose another estimate route using a procedure we call *iterative majority vote*. This procedure ranks the routes based on the prevalence of their prefixes. At each iteration i the procedure calculates – Prefix[i] – a list of prefixes of length i ranked by their prevalence out of the all routes that has a prefix in Prefix[i-1]. Prefix[i][n] denotes the prefix of rank n. The operation $p||j$ – where p is a route and j is an intersection – denotes the appendage of j to p. At each iteration i the procedure detailed in Algorithm 3. In the following we denote RoutePrefixed(R, p) to be the subset of routes out of the set R having p as their prefix.

Algorithm 3 Iterative majority vote

$I' \leftarrow I$
while not all prefixes found **do**
  Prf $\leftarrow$ next prefix from Prefix[i].
  Find $j \in I'$ that maximizes
    RoutePrefixed(RoutePrefixed(P_{final}, Prf), Prf$||j$)
  if no such j is found **then**
    $I' = I$
    continue loop
  end if
  Prefix$[i+1] \leftarrow$ Prefix$[i+1] \cup \{$Prf$||j\}$
  $I' = I' - \{j\}$
end while

At each iteration i we rank the prefixes based on the ranks of prefixes of the previous iteration. Namely, prefixes which are extensions of a shorter prefix having a higher rank in a previous iteration will always get higher ranking over prefixes which are extensions of a lower rank prefix. At each iteration the procedure first finds the most common prefixes of length $i+1$ which start with the most common prefix of length i found in the previous iteration and ranks them according to their prevalence. Then the procedure looks for common prefixes of length $i+1$ that start with the second most common prefix of length i found in the previous iteration, and so on until all prefixes of length $i+1$ are found. The intuition of this procedure is as follows. The procedure gives preference to routes traversing segments that commonly traversed by other routes. Such segments received a high score during the steps that they were chosen. Since we can not pick the most common segments separately from each step (a continuous route probably will not emerge), we iteratively pick the most common segment out of the routes that are prefixed with the segments that were already chosen.

VIII. Experiments

A. Data collection

Our experiments required collecting real power consumption data from smartphone devices along different routes. We developed the PowerSpy android application[8] that collects various measurements including signal strength, voltage, current, GPS coordinates, temperature, state of discharge (battery level) and cell identifier. The recordings were performed using Nexus 4 mobile devices.

B. Route distinguishability

To evaluate the first algorithm for distinguishing routes we recorded reference profiles for several different routes. We used a dataset of 43 profiles for 4 different routes[9] about 19 kilometers each. Driving in different directions along the same roads (from point A to B vs. from point B to A) is considered two different routes. We perform a leave-one-out cross validation, each time using one of the profiles for testing. Figure 5 is a confusion matrix, which shows a high success rate in classifying the routes. The achieved successful classification rate in this case was 93%. Adding another three distinct routes to the set (now having 7 distinct routes and 51 power profiles in total) resulted in 90.2% correct classification (Figure 6).

For another dataset of 13 profiles we got perfect classification, distinguishing two different directions along the same road.

We also evaluated the algorithm on a dataset of 18 profiles for 2 different routes of about 20 kilometers, collected in a completely different area[10] with higher cell density and obtained somewhat lower correct classification rate of of 78%, which is nevertheless significantly better than a random guess. We attribute the decrease in correct classification to higher cell density, resulting in more monotonous power profiles.

C. Mobile device tracking

We evaluate the algorithm for mobile device tracking using a set of 10 training profiles and an additional test profile. The evaluation simulates the conditions of real-time tracking by serially feeding samples to the algorithm as if they are received from an application installed on the device. We calculate

[8]To be released on the Google Play market and as open source after unblinding.

[9]While there might be some differences in the routes, we would still label them the same if they are similar enough (taking the same highway).

[10]Different country and a different cellular provider.

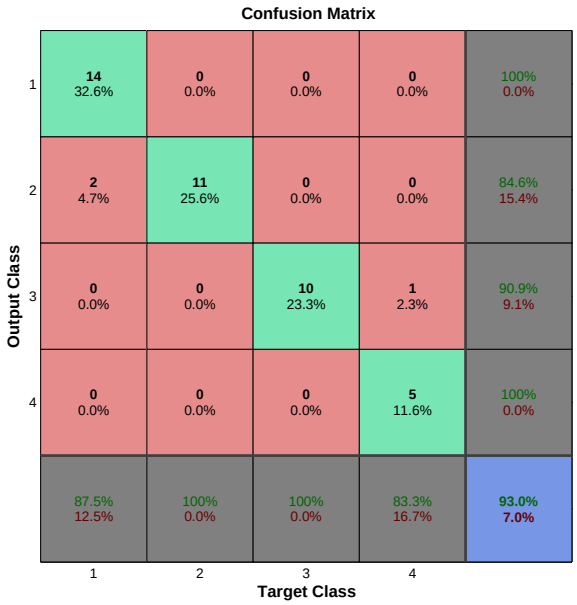


Fig. 5: Confusion matrix for classification with 4 possible routes of 19 kilometers each.

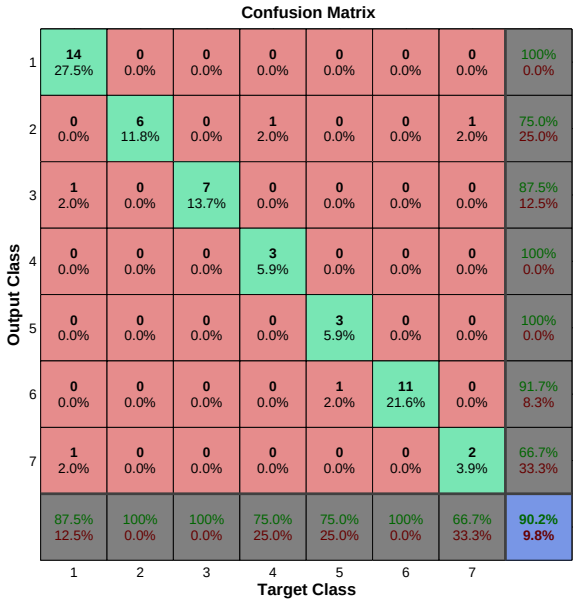


Fig. 6: Confusion matrix for classification with 7 possible routes.

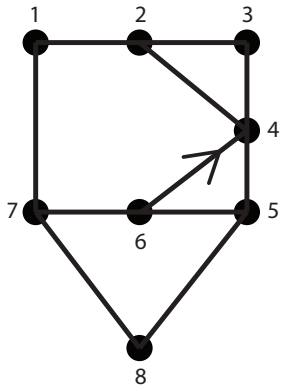


Fig. 8: Area for route inference. It is depicted schematically to serve the anonymity of this submission. The area in full detail will be included in the final version.

the estimation error, i.e. the distance between the estimated coordinates and the true location of the mobile device at each step of the simulation. We are interested in the *convergence time*, i.e. the number of samples it takes until the location estimation is close enough to the true location, as well as in the distribution of the estimation errors given by a histogram of the absolute values of the distances.

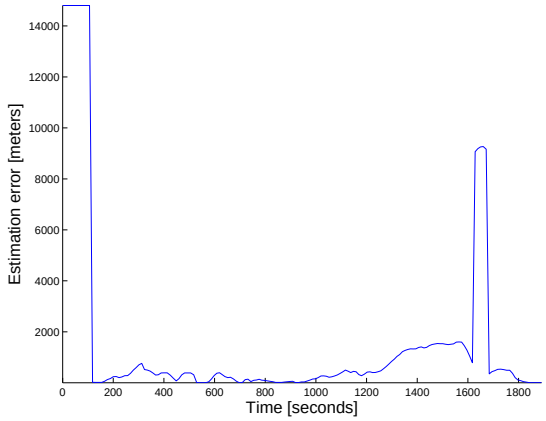
Figure 7 illustrates the performance of our tracking algorithm for one of the routes, which was about 19 kilometers long. At the beginning, when there are very few power samples, the location estimation is extremely inaccurate, but after two minutes we lock on the true location. We have a quite precise estimation up to some point after 20 minutes, where it starts to slightly diverge. This part is on a highway and with increase of velocity we have an increase in the estimation error. Around 26 minutes (in figure 7a) we have a large estimation error, but as we mentioned earlier, these kind of errors are quite easy to prevent by imposing a simple motion model. The histogram shows that most of the errors are small compared to the length of the route. 80% of the estimation errors are less than 1 km.

We also tested the improved tracking algorithm explained in VI-B. Figure 7d presents the estimation error over time, and we can see that the big errors towards the end of the route that appeared in 7a are not present in this case.

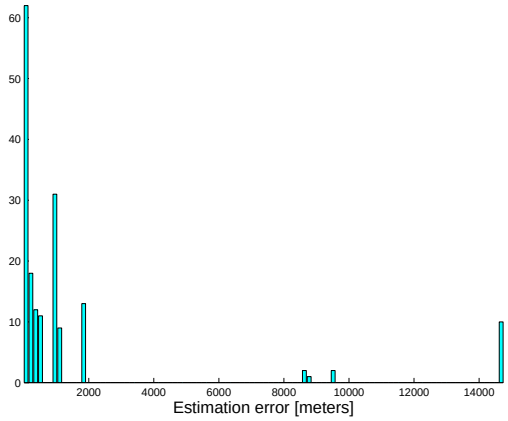
D. Inference of new routes

1) Setup: For the evaluation of the particle filter presented in Section VII we considered an area schematically depicted in Figure 8. The area has 8 intersections having 23 road segments [11]. The average length of a road segment is about 400 meters. The average travel time over the segments is around 90 seconds. The area is located in the center of a medium-sized city. Traffic congestion in this area varies across segments and time of day. For each power recording, the track traversed at least one congested segment. Most of the 8 intersections have

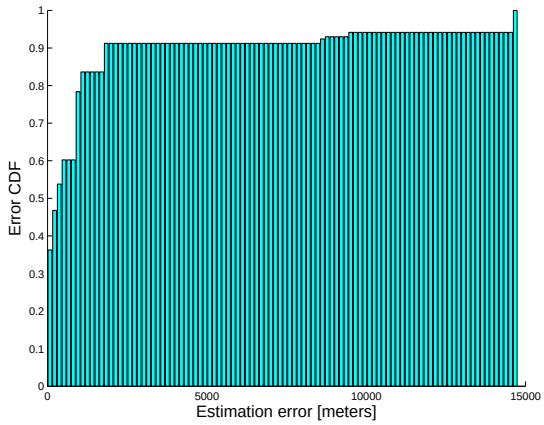
[11]One of the segments is a one way street as depicted in Figure 8.



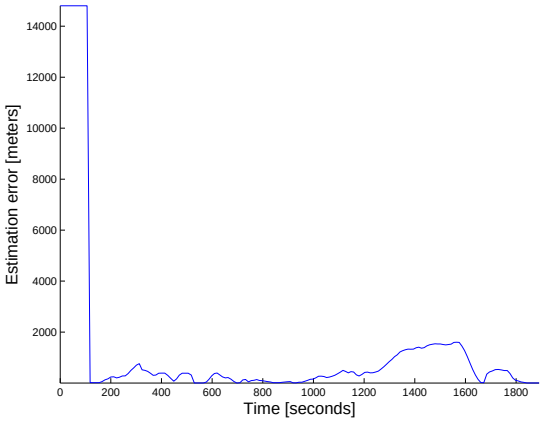
(a) Convergence to true location



(b) Error distribution



(c) Error cumulative distribution



(d) Location estimation error for improved tracking algorithm

Fig. 7: Location estimation error for online tracking

traffic lights, and about a quarter of the segments pass through them.

We had two pre-recording sessions of all segments. Each road segment was entered from every possible direction to account for the hysteresis effects. In total each pre-recording session produced 42 segments recording.

We set the following parameters of the HMM (as they are defined in Section VII-A):

1) A – This set defines the transition probabilities between the road segments. We set these probabilities to be uniformly distributed over all possible transitions. Namely, $a_{xyz} = \{1/|I_y| \; |I_y = \{w|(y,w) \in R, w \neq x\}\}$.
2) B – This set defines the distribution of power profile observations over each state. These probabilities depend on the road segments and their location relative to the nearby based stations. We do not need an explicit formulation of these probabilities to employ the particle filter. The likelihood of a a power profile to be associated with a road segment is estimated by the DTW distance of the power profile to prerecorded power profiles of that segment.
3) Π – This set defines the initial state distribution. We assume that the starting intersection of the tracked device is known. This applies to scenarios where the tracking begins from well-known locations, such as the user's home, office, or another location the attacker knows in advance.

For testing, we used two Nexus 4 phones (different from the one used for the prerecordings). Each phone was used to record the power profile of a different route. The two routes combined cover almost all of the road segments in the area. Table I details the routes. The recordings where done on different days.

As noted, we can only measure the aggregate power consumption which can be significantly affected by applications that continuously run. To have a better sense of the affects of these applications the two phones were run with different number of background applications. Phone #1 has a relatively modest number of applications which included (beyond the default Android apps): Email (corporate account), Gmail, and

Phone #1	8-5-6-7-1-2-3-4-5-6-4-3-2-1-7-8
Phone #2	7-1-2-3-4-5-8-7-6-5-4-2-1-7-8

TABLE I: Test Routes

Google Calender. Phone #2 has a much higher number of application which included on top of the applications of phone #1: Facebook, Twitter, Skype, Waze, and WhatsApp. All those applications periodically send and receive traffic.

For each of the two tracks we derived all possible sub-tracks having 2 to 7 road segments. We estimated each such sub-track. In total we estimated 88 sub-tracks. For each sub-track we employed Algorithms 2 and 3 to get two best estimates for the sub-track.

Table II summarizes the results of the route estimations for each of the two phones. For each route we have two alternatives for estimated route (1) the most frequent route in the particle set as output by Algorithm 2; (2) the route output by Algorithm 3. For each alternative we calculated the Levenshtein distance between it and the true route. The Levenshtein distance is a standard metric for measuring the difference between two sequences [10]. It equals the minimum number of updates required in order to change one sequence to the next. In this context, we treat a route as a sequence of intersections. The distance is normalized by the length of the longer route of the two. For each estimate we also note whether it is an exact fit with the true route (i.e., zero distance). The average distance and percentage of exact fits are calculated for each type of estimated route. We also calculate these metrics for both estimates combined while taking into account for each track the best of the two estimates. To benchmark the results we note in Table II the performance of a random estimation algorithm which simply outputs a random, albeit feasible, route.

The results in Table II show that the performance of the most frequent route output by the particle filter is comparable to the performance of the best estimate output by Algorithm 3. However, their combined performance is significantly better than either estimates alone. This result tells us that Algorithm 3 extracts significant amount of information from the routes output by the particle filter beyond the information gleaned from the most frequent route.

For Phone #1 the combined route estimates were able to exactly identify the true track for around 2/3 of scenarios. While the average distance was only 0.15, namely on average only around 1/7 of the estimated route is different than the true route. For Phone #2 which run many applications the route estimates are less accurate. Only 1/5 of routes are identified exactly, while on average 2/5 of the estimated route is different than the true route. This shows that the number of running applications can have a significant effect on the accuracy of the estimated route. Nonetheless, even in this case the percentage of exact fits are considerably better than a random guess (20% vs. 5%), while the average distance also presents a markable improvement (0.4 vs. 0.62).

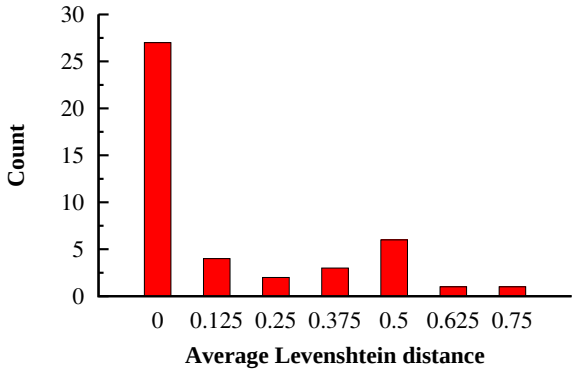


Fig. 9: Histogram of the Levenshtein distances for the estimated routes.

Figure 9 depicts the histogram of distances of the combined route estimates for phone #1. It can be clearly seen that in most cases we have an exact fit with the true route. When estimation errors occur, the distance to the true route is roughly uniformly distributed, indicating that there are some estimates with very low distances.

To have a better sense of the distance metric used to evaluate the quality of the estimated routes Figure 10 depicts three cases of estimation errors and their corresponding distance values in increasing order. It can be seen that even estimation error having relatively high distances can have a significant amount of information regarding the true route.

IX. Future directions

In this section we discuss ideas for further research, improvements, and additions to our method.

A. Power consumption inference

While new (yet very common) smartphone models contain an internal ampere-meter and provide access to current data, other models (for instance Galaxy S III) supply voltage but not current measurements. Therefore on these models we cannot directly calculate the power consumption. V-edge [11] proposes using voltage dynamics to model a mobile device's power consumption. That and any other similar technique would extend our method and make it applicable to additional smartphone models.

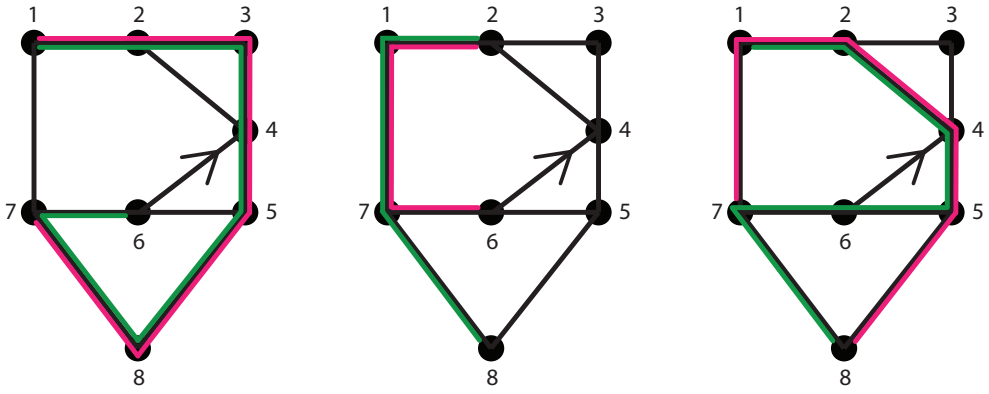
Ref. [12] presents PowerTutor, an application that estimates power consumption by different components of the smartphone device based on voltage and state of discharge measurements. Isolating the power consumed by the cellular connectivity will improve our method by eliminating the noise introduced by other components such as audio/Bluetooth/WiFi etc. that do not directly depend on the route.

B. State of Discharge (SOD)

The time derivative of the State-of-Discharge (the battery level) is basically a very coarse indicator of power consumption. While it seemed to be too inaccurate for our purpose, there is a chance that extracting better features from it or

	Average distance				Exact fits			
	random	most frequent	Alg. 3	combined	random	most frequent	Alg. 3	combined
Phone #1	0.62	0.35	0.27	0.15	5%	45%	45%	65%
Phone #2		0.52	0.57	0.40		16%	13%	20%

TABLE II: Summary of route inference results



(a) Distance = 0.125 (b) Distance = 0.25 (c) Distance = 0.43

Fig. 10: Examples of estimation errors and their corresponding distances. The true route is green and the estimated route is red. Schematic graphs are used to serve anonimity of this submission. Plots with actual coordinates on top of real map will be included in the final version.

having few possible routes may render distinguishing routes based on SOD profiles feasible. Putting it to the test is even more interesting given the HTML 5 Battery API that enables obtaining certain battery statistics from a web-page via JavaScript. Our findings demonstrate how future increases in the sampling resolution of the battery stats may turn this API even more dangerous, allowing web-based attacks.

C. LTE

Our evaluation was done for a 3G network. Testing how our method applies to LTE is important due to its growing deployment. While we expect our method to work just the same, it requires confirmation.

D. Choice of reference routes

Successful classification depends among other factors on good matching between the power profile we want to classify and the reference power profiles. Optimal matching might be a matter of month, time of day, traffic on the road, and more. We can possibly improve our classification if we tag the reference profiles with those associated conditions and select reference profiles matching the current conditions when trying to distinguish a route. That of course requires collecting many reference profiles.

E. Collecting a massive dataset

Collecting a massive dataset of power profiles associated with GPS coordinates is a feasible task given vendors' capability to legally collect analytics about users' use of their smartphones. Obtaining such big dataset will enable us to better understand how well our approach can scale and whether it can be used with much less prior knowledge about the users.

X. Defenses

A. Non-defenses

One might think that by adding noise or limiting the sampling rate or the resolution of the voltage and current measurements one could protect location privacy. However, our method does not rely on high sampling frequency or resolution. In fact, our method works well with profiles much coarser than what we can directly get from the raw power data, and for the route distinguishing task we actually performed smoothing and downsampling of the data yet obtained good results. Our method also works well with signal strength, which is provided with much lower resoltion and sampling frequency[12].

B. Risky combination of power data and network access

One way of reporting voltage and current measurements to the attacker is via a network connection to the attacker's server. Warning the user of this risky combination may somewhat raise the bar for this attack. There are of course other ways to leak this information. For instance, a malicious application disguised as a diagnostic software can access power data and log it to a file, without attempting to make a network connection, while another, seemingly unrelated, application reads the data from that file and sends it over the network.

[12]In fact, since it reflects more directly the environmental conditions, signal strength data can provide even better route identification and tracking. We did not focus on signal strength since accessing it requires access permissions and has already drawn research attention to it as useful for localization.

C. Secure hardware design

The problem with access to total power consumption is that it leaks the power consumed by the transceiver circuitry and communication related tasks that indicate signal strength. While power measurements can be useful for profiling applications, in many cases, examining the power consumed by the processors executing the software logic might be enough. We therefore suggest that supplying only measurements of the power consumed by the processors (excluding the power consumed by the TX/RX chain) could be a reasonable trade-off between functionality and privacy.

D. Requiring superuser privileges

A simple yet effective prevention may be requiring superuser privileges (or being root) to access power supply data on the phone. Thus, developers and power-users can install diagnostic software or run a version of their application that collects power data on a rooted phone, whereas the release version of the software excludes this functionality. This would of course prevent the collection of anonymous performance statistics from the install-base, but as we have shown, such data can indicate much more than performance.

E. Power consumption as a coarse location indicator

Same as the cell identifier is defined as a coarse location indicator, and requires appropriate permissions to be accessed, power consumption data can also be defined as one. The user will then be aware, when installing applications that access voltage and current data, of the application's potential capabilities, and the risk potentially posed to her privacy.

This defense may actually be the most consistent with the current security policies of smartphone operating systems like Android and iOS, and their current permission schemes.

XI. Related work

Power analysis has shown to be a powerful tool to leak information from a system in various contexts. The most well-known one is the recovery of an encryption key from a cryptographic system [13].

Prior work has established the relationship between signal strength and power consumption in smartphones [1], [14]. Further, Bartendr [1] demonstrated that paths of signal strength measurements are stable across several drives.

Geolocation Techniques [15] covers some of the GSM metrics based techniques for localization of a mobile device. Most works on estimation of location in the absence of GPS readings focus on two main approaches based on signal strength values (either for cellular or for WiFi): 1) fingerprinting - where a pre-recorded radio map of the area of interest is leveraged to infer locations through best matching. It creates a search index of radio fingerprints to latitude/longitude coordinates. This approach is closest to our work. 2) propagation based - in which signal strength values are used to calculate distances to base stations (or access points) with known locations through the computation of the path loss.

All fingerprint localization works (e.g., [16]–[19] require at least signal strength information and base station ID or WiFi network name (SSID). Our work does not rely on the signal strength but rather on power consumption, furthermore it does not rely on base station (cell) ID, which is acknowledged as a coarse location indicator for mobile devices.

A. Abusing smartphone sensors

An emerging line of work shows that phone sensors can be used in unexpected ways that can lead to unintended consequenes.

SurroundSense [20] demonstrates how ambient sound and light can be used for mobile phone localization, and although it focuses on legitimate use-cases, the same methods could be leveraged for breaching privacy.

AccelPrint [21] is an attempt to fingerprint smartphones by tracking imperfections in their accelerometer measurements. Fingerprinting of mobile devices by the charactersitics of their loudspeakers is proposed in [22], [23].

Lukas et. al. [24] proposed a method for digital camera fingerprinting by pattern noise present in the images. [25] enhances the method enabling identification of not only the model but particular cameras. Applied to smartphones it could give away a particular mobile device.

Bojinov et. al. [26] showed that various sensors on smartphones can be used to identify a mobile device by its unique hardware characteristics. The *Gyrophone* study [27] demonstrated that gyroscopes on smartphones can be used for eavesdropping on parts of a conversation in the vicinity of the phone and identifying the speakers. In this paper we show that same is true of the phone's power usage meter: it can be used in an unintended way to track the phone's location.

This line of research suggests that providing applications with unrestriced access to sensors, can potentially result in a security breach and compromise sensitive information.

XII. Conclusion

We showed that applications that read the phone's amperemeter can gain information about the location of a mobile device without accessing the GPS or any other coarse location indicators. Our approach enables known route identification, real-time tracking, and identification of a new route by only analyzing the phone's power consumption. We evaluated our methods on real-world data collected from popular smartphones that have a significant mobile market share, and demonstrated their effectiveness. We believe that with more data our approach can be further improved and be used to obtain even more information on the phone's location. We therefore conclude by suggesting several possible defenses. More generally, our work suggests that more security modeling needs to be done before giving 3rd party applications direct access to sensors.

References

[1] A. Schulman, V. Navda, and R. Ramjee, "Bartendr: a practical approach to energy-aware cellular data scheduling," *... conference on Mobile ...*, 2010. [Online]. Available: http://www.stanford.edu/~aschulm/docs/mobicom10-bartendr.pdfhttp://dl.acm.org/citati

[2] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *MobiSys*, 2012.

[3] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[4] G. P. Pollini, "Trends in handover design," *Communications Magazine, IEEE*, vol. 34, no. 3, pp. 82–90, 1996.

[5] P. Mohan, V. N. V. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *. . . of the 6th ACM conference on . . . .* New York, New York, USA: ACM Press, Nov. 2008, p. 323.

[6] M. Müller, *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. [Online]. Available: http://www.springerlink.com/index/10.1007/978-3-540-74048-3

[7] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, 1989.

[8] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, 2002.

[9] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the kalman filter," *IEEE AEROSPACE AND ELECTRONIC SYSTEMS MAGAZINE*, vol. 19, no. 7, pp. 37–38, 2004.

[10] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," in *Soviet physics doklady*, vol. 10, 1966, p. 707.

[11] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: fast self-constructive power modeling of smartphones based on battery voltage dynamics," *Presented as part of the 10th USENIX . . .*, 2013. [Online]. Available: https://www.usenix.org/system/files/conference/nsdi13/nsdi13-final135.pdf

[12] L. Zhang, B. Tiwana, Z. Qian, and Z. Wang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," *Proceedings of the . . .*, 2010.

[13] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology – CRYPTO'99*. Springer, 1999, pp. 388–397.

[14] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *USENIX Annual Technical Conference*, 2010.

[15] C. Gentile, N. Alsindi, R. Raulefs, and C. Teolis, *Geolocation Techniques*. New York, NY: Springer New York, 2013. [Online]. Available: http://www.springerlink.com/index/10.1007/978-1-4614-1836-8

[16] K. Muthukrishnan, B. J. van der Zwaag, and P. Havinga, "Inferring motion and location using WLAN RSSI," in *Mobile Entity Localization and Tracking in GPS-less Environnments*. Springer, 2009, pp. 163–182.

[17] J. Krumm and E. Horvitz, "Locadio: Inferring motion and location from wi-fi signal strengths." in *MobiQuitous*, 2004, pp. 4–13.

[18] T. Sohn, A. Varshavsky, A. LaMarca, M. Y. Chen, T. Choudhury, I. Smith, S. Consolvo, J. Hightower, W. G. Griswold, and E. De Lara, "Mobility detection using everyday gsm traces," in *UbiComp 2006: Ubiquitous Computing*. Springer, 2006, pp. 212–224.

[19] R. W. Ouyang, A.-S. Wong, C.-T. Lea, and V. Y. Zhang, "Received signal strength-based wireless localization via semidefinite programming," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6.

[20] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 261–272.

[21] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "Accelprint: Imperfections of accelerometers make smartphones trackable," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[22] W. B. Clarkson and E. W. Felten, "Breaking assumptions: distinguishing between seemingly identical items using cheap sensors," Tech. Rep., 2012.

[23] A. Das and N. Borisov, "Poster: Fingerprinting smartphones through speaker," in *Poster at the IEEE Security and Privacy Symposium*, 2014.

[24] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *Information Forensics and Security, IEEE Transactions on*, vol. 1, no. 2, pp. 205–214, 2006.

[25] C.-T. Li, "Source camera identification using enhanced sensor pattern noise," *Information Forensics and Security, IEEE Transactions on*, vol. 5, no. 2, pp. 280–287, 2010.

[26] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile device identification via sensor fingerprinting," *arXiv preprint arXiv:1408.1416*, 2014.

[27] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing speech from gyroscope signals," in *Proc. 23rd USENIX Security Symposium (SEC14), USENIX Association*, 2014.