SpotME If You Can: Randomized Responses for Location Obfuscation on Mobile Phones

Daniele Quercia, Ilias Leontiadis, Liam McNamara, Cecilia Mascolo, Jon Crowcroft
Computer Laboratory, University of Cambridge, Cambridge, CB3 0FD, UK.
{name.surname}@cl.cam.ac.uk

***Abstract*—Nowadays companies increasingly aggregate location data from different sources on the Internet to offer location-based services such as estimating current road traffic conditions, and finding the best nightlife locations in a city. However, these services have also caused outcries over privacy issues. As the volume of location data being aggregated expands, the comfort of sharing one's whereabouts with the public at large will unavoidably decrease. Existing ways of aggregating location data in the privacy literature are largely centralized in that they rely on a trusted location-based service. Instead, we propose a piece of software (SpotME) that can run on a mobile phone and is able to estimate the number of people in geographic locations in a privacy-preserving way: accurate estimations are made possible in the presence of privacy-conscious users who report, in addition to their actual locations, a large number of erroneous locations. The erroneous locations are selected by a randomized response algorithm. We evaluate the accuracy of SpotME in estimating the number of people upon two very different realistic mobility traces: the mobility of vehicles in urban, suburban and rural areas, and the mobility of subway train passengers in Greater London. We find that erroneous locations have little effect on the estimations (in both traces, the error is below 18% for a situation in which more than 99% of the locations are erroneous), yet they guarantee that users cannot be localized with high probability. Also, the computational and storage overheads for a mobile phone running SpotME are negligible, and the communication overhead is limited.**

I. Introduction

If location-based services are to succeed, they will have to alleviate growing privacy concerns over collecting people's whereabouts. *Loopt.com*, a mobile social-networking service, has done so by instituting safeguards against abuse: it sends reminders to users that their location is being shared and allows them to post "erroneous" locations if they would like to hide their current location. Researchers have taken this approach one step further. They have proposed the use of aggregate location data instead of location data about single individuals [20] in order to preserve user privacy. Companies have followed suit and proposed services based on analysis of aggregate location data. TomTom, the Dutch maker of navigation devices, has refined its ability to predict road traffic congestion by analyzing the paths of millions of Vodafone subscribers [5]. The company SenseNetworks analyzes tens of millions of location estimates from mobile phones to help people find the best corner to catch a cab in New York City or get personalized recommendations of the best nightlife locations in the city [17]. The resulting situation is that data about an individual's whereabouts is shared with companies that then promise to aggregate such data.

One finds it difficult to believe that companies will discard data about individuals and keep only aggregate versions of it, not least because companies generate revenue from targeted advertising, which can be done only with data about individuals. Often companies claim to be using only "aggregate anonymous data" and then gloss over it in their "terms and conditions" [4]. Only few months ago, The Centre for Democracy & Technology, a privacy group, argued that the privacy policies of companies collecting location data are "uneven at best and inadequate at worst" [1].

We set out to explore a way of aggregating location data that suits privacy-conscious individuals and, at the same time, results in fully-working location services. As we shall see in Section II, existing approaches trust a central server (e.g., the location-based service itself, a location broker) to perform data aggregation or carry out distributed tasks other than counting people in geographic locations (e.g., distributed management of push/pull queries [15, 21]). The idea behind our work is to choose a promising data obfuscation technique, apply it to the design of a privacy-preserving way of counting people in geographic locations, and study when and how it works upon real-data on a large-scale. Researchers have long been designing distributed techniques for preserving location privacy, but the quantitative proof of how a given technique works on a large-scale has not always been available. Being based on quantitative and large-scale location data, this work adds a new dimension to the current scholarship. More importantly, our findings create an experimental basis for emerging mobile social-networking services. People count is a simple task yet supports a wide variety of services such as real-time traffic regulation, urban areas profiling, rare events detection, and crowd analysis (which is interested in modeling the behavior of crowds for predicting the use of space, planning accessibility, and planning emergency evacuations). More specifically, we make two main contributions:

1. We propose a mechanism for aggregating user locations in real-time. The idea is that users can disguise their real position using an additional set of erroneous locations (Section III). A randomized response algorithm selects these locations so that the combined responses can be statistically analyzed to estimate, for example, the actual number of people in a given location. The result is that: i) users would alleviate their privacy concerns by obfuscating their location; and ii) the accuracy of the processed data will be only marginally

affected.

2. We evaluate the extent to which the effectiveness of processing aggregate data would be affected by our proposal upon two sets of real mobility traces (Section IV): i) traces of vehicles in urban, suburban, and rural areas; and ii) real traces of the journeys made by subway passengers of all the train lines in Greater London. The results suggest that one accurately estimates the number of users in a geographical location in the presence of a large number of erroneous locations (roughly 99% of the locations are erroneous) - the error is as low as 10% for both traces (Section IV-A). Erroneous locations make it difficult to localize single individuals and, if up to 60% of car drivers maliciously inject false location information, the estimations are unaffected (Section IV-B). We also find that, if 35% of the public use SpotME, we can still estimate the number of individuals of the *whole* public (those who use SpotME and those who do not) in a given location. The communication, computational, and storage overheads on mobile phones are negligible, even when the number of erroneous locations for each user is pessimistically high (Section IV-C).

We conclude in Section V by summarizing the key experimental results. Nowadays mobile social-networking companies may go bust because they find it difficult to collect data from a critical mass of users. One way of solving this problem is to make it possible for companies to collectively share a pool of aggregate location data. We expect that SpotME users may be willing to contribute to this common pool as they now have the possibility to obfuscate their whereabouts directly on their mobile phones.

II. Existing Solutions

Users may well trust location-based services to handle their private data and, if they do so, they simply need to disclose their location from their mobile phones in a way that no unauthorized third party can access it. One way for a mobile phone to do this is to use a connection to the Internet that performs *"onion routing"* [10], whereby an Internet packet is repeatedly encrypted and forwarded from one onion router (network node) to another up to the destination. Each onion router removes a layer of encryption to uncover routing instructions. The result is that no third-party (including the onion routers) knows the origin, destination, and content of the packet.

Once a service receives location data, it stores the data along with users' privacy settings. The settings are then translated into machine-readable policies [13], and *access control mechanisms* interpret those policies to grant or deny access to location data. For example, users may want to tune their settings in a way that only their closest social contacts can know their location outside working hours. However, to regulate access in a variety of real-life situations, one would require very complex, and therefore, hard to use privacy settings [19]. That is why algorithms that automatically predict privacy preferences from a user's behavior have also been proposed [6]. More recently, researchers have been working on access control mechanisms that not only control the access to private data but make it possible to "claim the data back". Geambasu *et al.* [7] proposed a system that, by integrating cryptographic techniques with distributed hash tables (DHTs), is able to make all copies of certain data become unreadable after a user-specified time, even if one obtains a cached copy of the data.

Access control mechanisms either grant or deny access to location data. However, one may want to go beyond the simple dichotomy access granted/denied by, for example, being able to make aggregate location data publicly available. One way to do so is "*data generalization*" [2, 11, 16]. The idea is that a user sends her location to the service, which then generalizes the location with a coarser-grained spatial range. The goal is to guarantee the user's *k-anonymity*. The user is considered *k-anonymous* if her location is indistinguishable from the location of at least *(k-1)* other mobile users [9]. More recently, Hoh *et al.* proposed a decentralized approach of data generalization that relies on geographic markers in a vehicular infrastructure [12]. In push/pull mobile query systems, Kido *et al.* [15] and Shankar *et al.* [21] proposed to preserve k-anonymity by generating fake queries based on "usual patterns" extracted from user mobility, and Pingley *et al.* [18] and Gedik and Liu [8] designed a family of principled privacy-preserving techniques for query perturbation.

The approaches of onion routing, access control, and data generalization are effective privacy tools but are meant to work in specific situations. Those approaches often assume that the location-based service is trusted and that it adheres to the privacy preferences specified by users, they impose considerable computational and communication overheads on mobile phones, and they assume that users will regularly fiddle with their privacy settings. Those assumptions are reasonable for a specific class of applications and users but are not the best ones for designing mobile applications working on aggregate location data.

It thus seems that a new mechanism for collecting aggregate location data is needed. But what sort of mechanism should we use? Ideally, the mechanism should: i) conservatively distrust location-based services; ii) run seamlessly on mobile phones; and iii) require little user intervention.

III. Our Proposal: SpotME

We design a mechanism (called SpotME) that allows users to obfuscate their real location while still allowing aggregation. Next, we will describe what SpotME is (Section III-A), when it works (Section III-B), how it works (Section III-C), and study its vulnerability to attacks (Section III-D).

A. What it is

Problem Statement: How a service accurately estimates the number of people in a geographic location without individuals unequivocally revealing their true position.

By accurate estimations, we mean that the estimate of the number of people in a geographic location is close to the real number of people. SpotME alleviates mobile users' privacy concerns by allowing them to report erroneous locations in addition to their actual one. We use a technique called *'randomized response'* [22] to generate this report. This technique is used in structured survey interviews to increase the validity and reliability of self-reported behavior on sensitive issues. It has been used to ask people sensitive questions, e.g., whether they use drugs, whether they have been with a prostitute this month, or whether they have evaded paying taxes.

To see how it works, let us assume that we are interested in tax evasion. We ask an individual *whether he has evaded paying taxes*, but before he answers, we ask him to secretly flip a coin. He will then always answer "yes" if the coin came up tails, but will otherwise be truthful. So if he answers "yes", nobody but him knows why – it could be because he got tails or because he did *evade* his taxes. When applied to a large population, with a fair coin, half of those who did evade taxes will claim they did not, due to the coin toss. To find the proportion of people who actually paid taxes, we double number of the people who said "no", as half of them were forced to answer "yes". For instance, if 40% answered "no" then actually 80% of the population did not evade taxes.

More generally, let the probability of answering *forced "yes"* be p (which is the probability of a biased coin coming up tails). The proportion $\widehat{P_{no}}$ of claimed "no" answers from the survey is: $\widehat{P_{no}} = P_{no} - p \cdot P_{no}$, where P_{no} is the *true* proportion of "no". In other words, the proportion $\widehat{P_{no}}$ of "no" answers from the survey comes from the proportion of truthful "no" answers P_{no} minus the proportion of actual "no" answers that were forced "yes" ($p \cdot P_{no}$).

So, given a survey answer, the proportion of actual "no" answers can be estimated:

$$P_{no} = \frac{\widehat{P_{no}}}{1-p} = \frac{1-\widehat{P_{yes}}}{1-p} \quad (1)$$

Going back to our previous example, if 40% answered "no" (i.e., $\widehat{P_{no}} = 40\%$), then $P_{no} = \frac{40}{0.5} = 80\%$. Likewise, if a biased coin was used with $p = 0.2$, then $P_{no} = \frac{40}{0.8} = 50\%$.

Similarly, the proportion of "yes" answers from the survey comes from the proportion of truthful "yes" answers P_{yes} plus the proportion of actual "no" answers that were forced "yes": $\widehat{P_{yes}} = P_{yes} + p \cdot P_{no}$. By substituting (1) and solving for P_{yes} we obtain:

$$P_{yes} = \widehat{P_{yes}} - p \cdot \frac{1-\widehat{P_{yes}}}{1-p} = \frac{\widehat{P_{yes}} - p}{1-p} \quad (2)$$

From expression (2), one can estimate the number of true "yes" answers from the number of claimed ones. In Section III-C, we will see that, similarly, SpotME uses expression (2) to estimate the number of people in a given area on input: i) of the number of yes and no answers in the area; and ii) of a fixed p.

Symbol	Description
p	Probability of forced "yes"
l	A location in a map
k	Number of locations in a position map
m_t	Position map at time t
m'_t	Expanded position map at time t
r	Rate of update for position maps
$yes_{l,t}$	Number of devices that claim to be in l at time t
$no_{l,t}$	Number of devices that claim *not* to be in l at time t
$total_{l,t}$	Number of answers given for location l at time t
w	Window size for the moving average
u	Proportion of the public using SpotME
v	Proportion of malicious users

TABLE I
SYMBOLS AND TERMINOLOGY.

B. When it works

SpotME works under the assumption that it is possible to divide a geographic area into a set of locations. A portable device is able to compute its position and is able to place the point into the corresponding location. Without loss of generality, in our evaluation (Section IV), locations are geographic squares in the case of the vehicular traces and subway stations in the case of the subway traces and, in general, locations can be of any shape.

We do not assume that SpotME users are truthful. In our robustness analysis (Section IV-B), we will see that SpotME is robust against a large fraction of malicious individuals who inject false location information.

C. How it works

SpotME ensures that users create and send their position maps to a location-based service (*Step 1*), and that the service is able to accurately compute the number and flow of users in a location from those maps (*Step 2*):

Step 1: *Users create and send their position maps.* A portable device's *position map* is a set of k locations the device claims to be in. One of those locations is the user's true location. For each location l in the map, a randomized response algorithm is used to let the user claim to be present there or not as follows:

- Claims to be in location l with probability p.
- Or says the truth (whether it is actually in l or not) with probability $(1-p)$.

A portable device sends a new position map either at a fixed rate (e.g., every minute) or whenever it moves from one location to another. By contrast, if the device is stationary, it does not disclose anything but an acknowledgment. The acknowledgment is used to counter a simple localization attack that will become clear at the end of Section III-D. It is also used to signal that the user is still part of the system and the user's latest position map should be considered valid.

It is important to say that *the device will not just send the "yes" answers*. It will send k answers where some of them will be "yes" and some "no", and *both* "yes" and "no" answers are used to estimate the number of people in a

location.

Step 2: *A service infers number and flow of people from the maps.* The location-based service collects all the maps it receives and then processes them to infer, at each location l, the number of individuals in l. More specifically, for a location l, the service receives $total_{l,t}$ answers at time t: $yes_{l,t}$ of which are "yes" and $no_{l,t}$ of which are "no". The proportion of given "yes" answers for location l is then:

$$\widehat{P_{yes}}_{l,t} = \frac{yes_{l,t}}{total_{l,t}} \tag{3}$$

Given this sum of randomized answers, we are able to estimate the *real* number of users at location l at time t:

$$estimated_{l,t} = total_{l,t} \cdot P_{yes_{l,t}}$$

By substituting (2) and (3):

$$estimated_{l,t} = total_{l,t} \cdot \frac{\widehat{P_{yes}}_{l,t} - p}{1-p} \tag{4}$$

SpotME uses equation (4) to estimate the real number of users in location l. Clearly, this number depends on the proportion of the "yes"/"no" answers given and on the probability p. For example, if we received $total_{l,t} = 100$ answers for a location l, and 80% of them claim to be in l, then, for $p = 0.5$, the estimated number of actual users is: $estimated_{l,t} = 100 \cdot \frac{0.8-0.5}{1-0.5} = 60$.

In reality, estimations are done at discrete time steps, and short-term fluctuations may likely perturb such estimations due to the variance of the binomial distribution. The simplest way to smooth out short-term fluctuations and highlight longer-term trends is to compute the moving average of the estimations for a window w. So our estimated number of individuals in location l at time $[t-w, t]$ now becomes:

$$predicted_{l,t,w} = \frac{\sum_{i=(t-w)}^{t} estimated_{l,t}}{w} \tag{5}$$

In this expression, to make a prediction, we average the last w estimations. In our evaluation (Section IV), we find that even for very low values of w, the predictions can be effectively smoothed out and, for $w > 20$, the prediction error remains stable. Therefore we will use a window size $w = 20$.

SpotME is not only able to estimate the number of users in one location but also to infer the number of people entering/exiting the location (the population flow). It may do so by ignoring stationary users. Distinguishing between stationary and mobile users is in fact very simple - stationary users send acknowledgments and mobile ones send position maps. Then, given a location l, by excluding stationary users, the location-based service only counts recent arrivals at the location and, as such, the resulting $estimated_{l,t,w}$ (as per expression (5)) is the predicted number of individuals who have recently arrived in location l.

D. SpotME attack vulnerability

Let us assume now that the service is a malicious attacker that wants to localize a SpotME user. To see how it might do so, consider that the attacker collects three position maps m_1, m_2, and m_3 that the user generates at three consecutive time steps (Figure 1). The black locations are the locations in which the user claims to be. After receiving m_1, the attacker knows that the user is currently in one of the black locations. In the next time step, if the user moves one location, then they will be in one's of m'_1's grey or black locations. m'_1 is the expanded version of m_1 (locations adjacent to the black ones are selected) and is created by the attacker to infer where the user is likely to be next. In the next time step, the user claims to be in the black locations in m_2. The intersection of m'_1 and m_2 gives the attacker four possible locations (the black locations in m'_2). Subsequently, the intersection of m_3 and the expanded m'_2 gives a single possible position in m'_3. The user's true position has been found.

In reality, localizing the user is not as easy as we have just shown. To disguise true locations, one may select fake locations in a way that they do not affect the overall people count and appear to be plausible (e.g., locations users are likely to be in). For example, for push/pull mobile query systems, Kido *et al.* [15] and Shankar *et al.* [21] proposed to generate fake queries based on "usual patterns" extracted from user mobility. One may also select fake locations in a random way: we will evaluate the extent to which it works to select fake locations in the position map at random without any geographical constraint. Critics might rightly say that randomly-chosen fake locations would be geographically sparse, while true locations would be clustered or relatively continuous. But that is true only if the value of p is low, as it is in Figure 1 in which p is roughly 0.1 - the number of black squares per map is 6 out of 64, so $p \sim \frac{6}{64} \sim 0.1$. In our 'Evaluation' (Section IV), we will see that, for $p = 0.5$ (i.e., the user claims to be in half of the location in m_1), it is very difficult to localize the user, yet the fidelity of the results from aggregate location data remains extremely high.

Of course, if the user is stationary and were to periodically send new position maps, then the attacker would quickly localize the users. That is why, as mentioned in the previous subsection, a stationary user does not send any new position map but just an acknowledgment.

IV. Evaluation

The goal of SpotME is to make it possible for mobile users to report erroneous locations in addition to their actual one, without compromising the precision of applications that rely on aggregate location data. To ascertain the effectiveness of our proposal at meeting this goal, our evaluation ought to answer three questions:

- How effectively is the service able to estimate the number of people in a location from the position maps sent by SpotME users (Section IV-A)?
- To what extent would a malicious service be able to infer a user's location from the user's position maps? To what

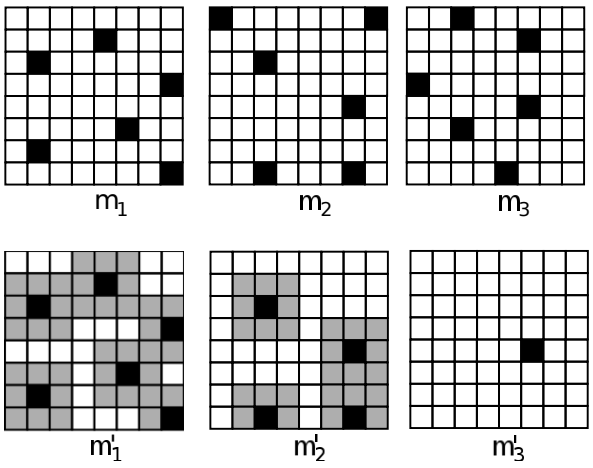

Fig. 1. A malicious service provider infers the location of a victim from the victim's position maps.

(a) Urban area (b) Suburban area (c) Rural area

Fig. 2. The mobility traces of drivers are generated for three types of area in Zurich.

extent would malicious users who inject false locations be able to compromise the estimations (Section IV-B)?
- What communication, storage, and computational overhead does SpotME impose on a mobile phone and on a location-based service that supports it (Section IV-C)?

To answer those questions, we set up experiments using movement data of different types of individuals.
The mobility traces we use are of two types:

1. Drivers of vehicles. We also use the mobility traces of drivers generated by the widely-used microscopic traffic simulator developed at ETHZ [3]. We generate the vehicular traces upon three real maps of the Zurich areas. On input of historical flows of vehicles in those areas, the traces are generated in a way that the behavior of vehicles is reproduced (e.g., stopping at traffic lights, vehicles queuing after each other, priority to the right). Since the maps reflect three types of area (urban, suburban, and rural), the resulting traces reflect scenarios with different total number of vehicles, density of vehicles, and vehicle speed:

Urban area: The number of vehicles is high (on average there are 880 vehicles at the same time), and so is the density of streets (see Figure 2(a)). The average speed is $20km/h$ and the maximum is $60km/h$.

Suburban area: The number of vehicles is medium (on average there are 420 vehicles at the same time), and so is the density of the streets (see Figure 2(b)). The average speed is $25km/h$ and the maximum speed is $60km/h$.

Rural area: The number of vehicles is low (on average there are 200 vehicles at the same time), and so is the density of streets (see Figure 2(c)). The average speed is $32km/h$ and the maximum is $70km/h$.

2. Subway passengers in London. We use the mobility traces of subway passengers of all the London subway's lines during rush hour of a weekday. For each passenger, the traces keep track of the stations the passenger enters and exits. This is possible for the 22% of the London underground passengers (which include commuters, shoppers, and tourists) who use RFID tickets to pay their fares. The number of subway stations considered is 426. The total number of passengers is 300,000, and the average number in a station each minute is 37 people with a peak of 702.

We divide the three maps in $100 \times 100m$ grid cells, and each cell is what we have called *location*. Instead, for the traces of the subway passengers, a location is a subway station.

Validation Execution. For each time step t and each location l, we compute the fraction of overestimated/underestimated individuals:

$$f_{l,t} = \frac{real_{l,t} - predicted_{l,t,w}}{real_{l,t}}$$

where $real_{l,t}$ is the actual number of individuals in location l at time t and $predicted_{l,t,w}$ is the number of individuals SpotME predicts based on the statistical analysis that we described in the previous section. Of course, the error is only defined for areas where there are more than one users.

We then compute the Root Mean Square Error (RMSE) for time t over all locations l:

$$error_t = \sqrt{\frac{\sum_{l=1}^{n} f_{l,t}^2}{n}}$$

where n is the number of locations considered (the number of data points). We repeat this on all locations for 600 time units (seconds) for the vehicular traces and for 120 time units (2 hours) for the subway traces and take the average error of the whole simulation. We use $RMSE$ because, in our case, large errors are particularly undesirable, and, since the errors for each location are squared before they are averaged, the $RMSE$ gives a relatively high weight to large errors [14].

A. Effectiveness

One would expect that the ability to estimate the number of people in a location is affected by three parameters:

1. The probability p of forced "yes". We consider the worst case in which k is maximum - a situation in which a user can potentially claim to be in all locations of the entire geographic

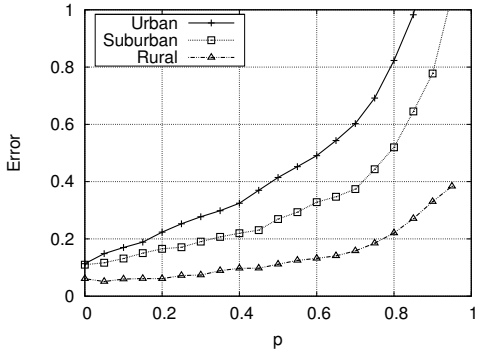

Fig. 3. The error versus p when each driver reports to be in all possible locations with probability p for the vehicular traces.

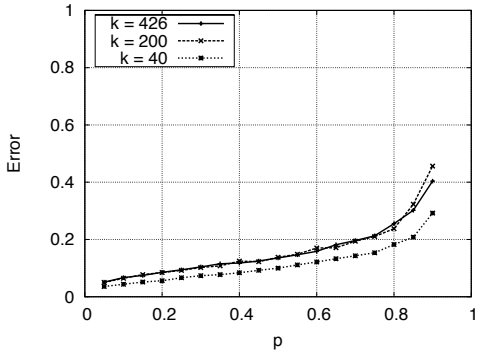

Fig. 4. The error versus p when each subway passenger reports to be in k locations (k maximum is 426).

area. That is, $k = 900$ for drivers and $k = 426$ for subway passengers. k maximum corresponds to the case of highest error. That is because the error increases with the number of individuals who claim to be in a location but are not there, and this number is maximum when everybody claims to be everywhere (k maximum). Figure 3 plots the error versus p for $k = 900$ in the three areas of the vehicular traces. As one expects, the error increases with p. However, for probability $p = 0.5$ (users can claim to be in half of the geographic area), the error is high in the urban area ($error = 41\%$) but is surprisingly low in the rural area ($error = 11\%$). The error will always be higher in dense areas (urban areas) since the average number of people who claim to be in a given location without being necessarily in it increases with the population density. Interestingly, for $p = 0$ (users claim to be in their true location), there is a residual error of 6% for the rural area and of 11% for the urban and suburban areas. That is because the moving average smooths short-term traffic fluctuations and cannot reproduce them (hence the error). Short-term traffic fluctuations are frequent in the urban area where intersections and traffic lights produce bursty vehicular traffic. However, at the price of having the residual error, we obtain a significantly reduced error for $p > 0$. If we were not to use the moving average, the error would roughly triple. For subway passengers, the error for k maximum (which in this case is 426) is reasonable and lower than that for vehicular drivers (Figure 4); up to $p = 0.6$, the error is lower than 20%. That is because the number of passengers is more than three order of magnitude higher than the number of drivers and, consequently, k's impact is greatly reduced.

2. The number k of potential locations. Since the error is unreasonably high for k maximum (especially for vehicular drivers), we explore the whole range of $k = [0, 900]$. We find that the error remains the same for $k < 50$ and becomes undesirably high for $k > 500$. Consequently, to ease explanation on how k and p affects the error, we graph the error for $k = \{50, 100, 200, 500\}$. Figure 5 shows the error versus p in the three different areas of the vehicular traces. From the three graphs, we learn that:

- High values of error are registered in the urban area because of bursty traffic and population density, as we have already found.
- For the considered k's in all the three areas, the error is low and stable for $p < 0.4$ and becomes undesirable for $p > 0.5$.

Based on the last point, we might say that, for $p < 0.5$, the error is low. The same applies for subway passengers; more specifically, the error is well below 18% for $p = 0.5\%$ (Figure 4 for $k = \{85, 42\}$). As we shall see in Section IV-B on the attacker model, for those values of p, localizing users becomes hard and, using a value of $p \in [0.4, 0.5]$, one is able to strike the balance between low error and high undetectability. Consequently, the location-based service fixes the value of p in the range $[0.4, 0.5]$, and users tune k depending on the level of privacy they would like to attain. The product $p \times k$ is the number of locations in which each user claims to be on average and expresses the level of privacy one can expect from the service. For example, if $p = 0.5$, a user claims to be, on average, in 50 locations at the same time, if she sets $k = 100$; or in 25 locations, if she sets $k = 50$. Of course, to attain very low error, the service may well choose a low value of p, say, 0.2; in that case, the users need to set a high value of k, say, at least 200.

3. The fraction u of individuals running SpotME. The previous results tell us that, by tuning the values of p and k, one is able to accurately estimate the number of SpotME users in a given location. Now, we will see that one is also able to estimate the number of people (including those *not* using SpotME) in a given location. Let us call u (uptake) the fraction of SpotME users in the population. One way to estimate the number of people $people_{l,t}$ in location l at time t is to compute:

$$people_{l,t} = \frac{predicted_{l,t,w}}{u} \tag{6}$$

where $predicted_{l,t,w}$ is the number of SpotME users predicted to be in l at time t. Expression (6) assumes that SpotME users are distributed uniformly across locations (it assumes that u is the same for all locations). In reality, we might have different penetration rates for SpotME in different areas. To test the extent to which this assumption affects the estimation of the number of people, we compute the error in the estimation versus u for the three areas and for the subway traces. In

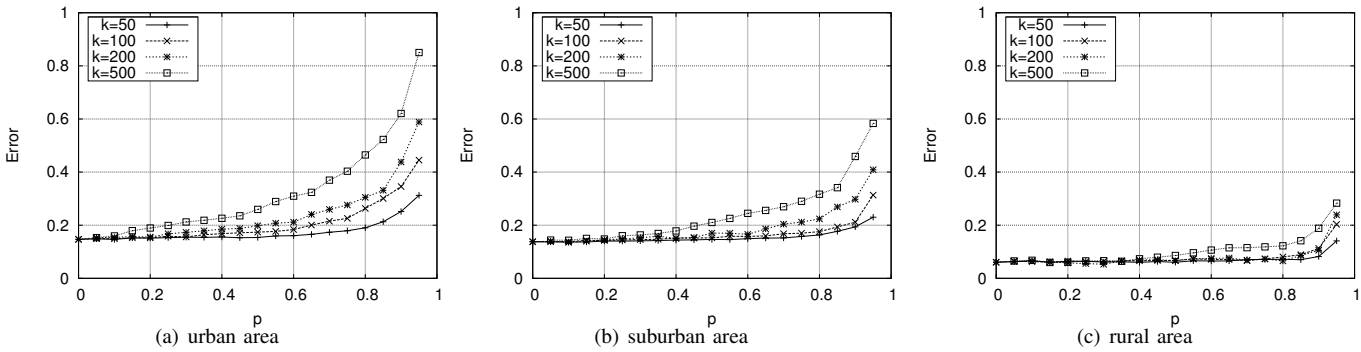

Fig. 5. The error versus p when each vehicular driver claims to be in 50,100, 200 locations at the same time with probability p. The results are for the three areas in Zurich: (a) urban; (b) suburban; and (c) rural.

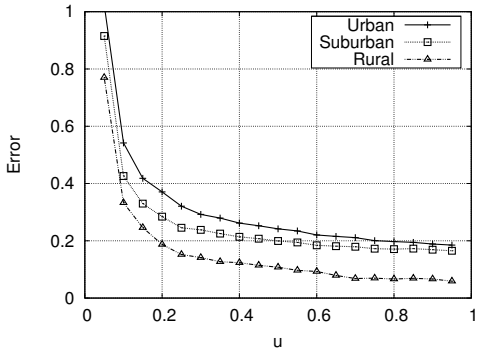

Fig. 6. The error versus SpotME's penetration rate u among vehicular drivers.

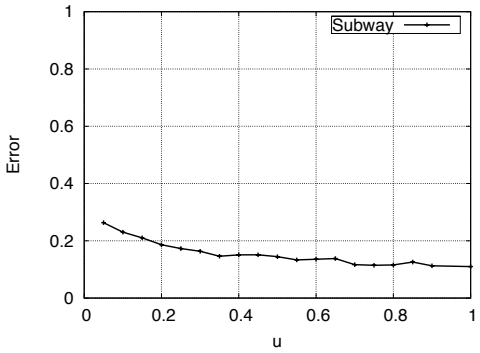

Fig. 7. The error versus SpotME's penetration rate u among subway passengers.

Figure 6, we see that, for $u = 25\%$ ($\frac{1}{4}$ of the population has installed SpotME), one can still estimate the number of people (non-SpotME users) with an error of 35% in the vehicular urban area and 19% in the rural area. For subway passengers, we also find a low error (below 20%) for a penetration rate of 20% (Figure 7). Interestingly, since the number of subway passengers is high, for penetration rates higher than 35%, the error does not significantly decrease. This would suggest that if 35% of the subway passengers of our London subway's traces were to adopt SpotME, then the number of people who are *not* necessarily using SpotME in each subway station could be accurately predicted. Clearly, the estimation of the number of SpotME's users is unaffected by the penetration rate, and the error is that we have discussed in the previous point.

B. Robustness against malicious providers and malicious users

In Section III-D, we described how a malicious service provider (attacker) could deduce the position of a victim and we claimed that the higher the victim's probability p, the more difficult localizing the victim. To measure how p impacts the attacker's ability to localize the victim, we set up a simulation in which, at each time step, the attacker collects a new position map from the victim and then selects from the map the locations in which the victim is likely to be. Being about a single position map, this simulation is not affected by mobility, so we do not use any mobility traces now. The percentage c of the selected locations (out of the total map) is a measure of the attacker's ability to localize the victim. The lower c, the more refined the localization. Figure 8 plots c versus time, for values of $p = \{0.1, 0.2, 0.3, 0.5, 0.7, 0.9\}$. We see that c tends to quickly stabilize over time. If the victim's probability p of forced "yes" is 0.1, then the attacker is able to quickly identify the location in which the victim is. By contrast, if p is 0.5, then the attacker predicts that the victim is in half of the locations and cannot refine its prediction over time. That is because the attacker guesses the victim's position by expanding the victim's map; for example, by expanding m_1 of Figure 1 into m'_1. If $p = 0.5$, then the victim claims to be in half of the possible locations (that is, half of the squares in the victim's map m_1 are black) and, to guess where the victim is, the attacker expands that map and obtain a new map m'_1 in which the victim is equally likely to be in any location of the map; for $p = 0.5$, the attacker is unable to localize the victim. In general, we might say that, for a given p, a user can expect a fixed and known level of privacy protection. This level is the product $(p \times k)$, which is the number of locations the user claims to be at the same time (on average).

In addition to a malicious provider, we might also have malicious users who try to subvert the system by injecting false position maps. Those users follow all the steps of the SpotME's algorithm expect for one: they do not report their current true locations, that is, they report locations at random.

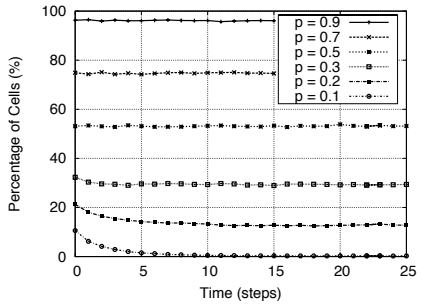

Fig. 8. The percentage of locations a mobile victim is believed to be in by a malicious attacker. As time passes by, the attacker receives a set of position maps from the victim upon which it then makes its predictions.

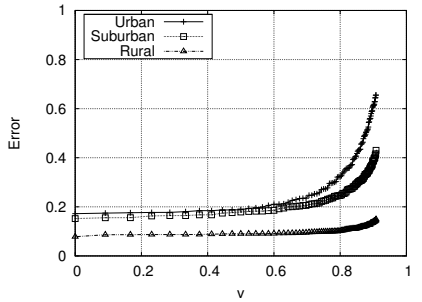

Fig. 9. The error in estimating the number of honest drivers versus the fraction v of 'injected' malicious vehicular drivers.

If many users do so, then the corresponding error in counting the number of the remaining (honest) users is expected to increase. To measure the extent to which the error increases, in our vehicular traces, we add a set of malicious users who inject random location maps and we measure the resulting error. Figure 9 plots the error versus the fraction v of vehicular drivers who are malicious. This fraction is the number of drivers who are malicious over the total number of (honest and malicious) drivers. Interestingly, up to 60% of malicious drivers, the error in estimating the number of the honest ones only slightly increases. Again, for subway passengers, the results are quantitatively similar.

C. Overheads

Storage Overhead. A device that runs SpotME stores one position map and the list of visited locations. This size of this information is very small (e.g., in our vehicular experiments it was 30KB), making such an approach feasible to be used in portable devices.

Computational Overhead. The computational overhead comes from the generation of k random numbers every time a new position map is created. Generating 2500 random numbers (worst case) on a mobile phone takes few milliseconds and, as such, SpotME is expected to run seamlessly on any modern mobile phone.

Communication Overhead. The communication overhead a mobile phone would see in using SpotME largely depends on three parameters: i) the number k of potential locations a SpotME user can report to be in, ii) the frequency which the phone sends a position map and iii) the dimensions of a location. One expects that the communication overhead increases linearly with the dimension k of the user's position map. Figure 10(a) shows that this is the case in our vehicular traces for the three types of areas (urban, suburban, and rural) and also shows that the overhead is extremely low: for $k = 1500$, a user has to only transmit 0.02 KB/sec (72 $KB/hour$).

To further decrease the communication overhead, one may have users not updating their position maps every time they change location but updating them at a fixed rate. Figure 10(b) plots the communication overhead for car drivers (KB/sec) versus the frequency of updates ($seconds$). By comparing the curves for 'dynamic' updates (maps updated at every change of location) to the curve for fixed update (maps updated at every given number of seconds), one concludes that having users updating their maps with a frequency greater than 6 seconds would result in a further decrease of communication overhead (which is already low).

An additional factor expected to impact the communication overhead is the dimension of each location in our maps. No matter whether users update their map at every change of location (dynamic update) or update them every seconds (fixed update), the larger each location, the lower the communication overhead. To see why, consider that:

- *For dynamic update:* The larger a location in the map, the more time people spend in it, the fewer the number of total changes of location overall, and the fewer the number of updates that need be sent.
- *For fixed update:* The larger each location in the map, the lower the number of total locations, the smaller each update (position map).

Figure 10(c) plots the communication overhead ($KB/second$) for car drivers versus the dimension of a location ($meters$). For both types of update, the network overhead decreases as expected, and it does so according to a power-law relationship. However, if users update at every change of location, the overhead decreases more rapidly than having users updating at a fixed rate. This suggests that the fixed rate should be set in a way that the resulting updating is less frequent than updating at every change of location, and the choice of the rate will be application-dependent.

For the subway passengers, the overall communication overhead is even lower than that for vehicular drivers. That is because the number of potential locations k is lower (at worst, it is the number of subway stations) and so is the frequency of updates (subway passengers move slower than vehicles).

V. Conclusion

SpotME makes it possible for services that rely on aggregate location data to work in the presence of privacy-conscious individuals who report erroneous locations in addition to the actual ones. Using real mobility traces in very different settings, we have seen that erroneous locations have little

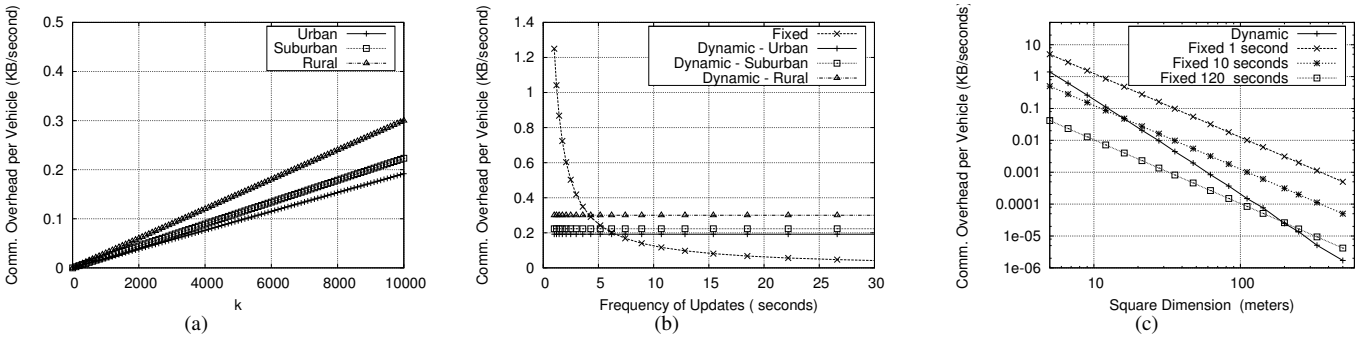

Fig. 10. The communication overhead as a function of: (a) the number k of potential locations each driver can report to be in; (b) the frequency of updates of position maps in the range $[1, 30] seconds$; and (c) the dimension of each location in the three maps of the areas in Zurich.

impact on the estimation of number of people in a geographical area and allow users to obfuscate their location. Also, SpotME scales (it entails reasonable communication overhead, and negligible computational and storage overheads on a mobile phone); is robust against injection of false location information; and is easily deployable largely because it does not require any additional infrastructure or any specialized hardware.

We are currently working on a protection mechanism that lets people signal whether they would like to be associated with the data they place on location-based services, and to be consulted about unusual uses.

Acknowledgments. We thank EPSRC for its financial support. We also thank Mirco Musolesi, Anastasios Noulas and Salvatore Scellato for their constructive feedbacks.

REFERENCES

[1] The privacy implication of commercial location-based services. *Center for Democracy & Technology*, February 2010.

[2] G. Andrienko, N. Andrienko, F. Giannotti, A. Monreale, and D. Pedreschi. Movement data anonymity through generalization. In *Proc. of the* 2^{nd} *SIGSPATIAL ACM Workshop on Security and Privacy in GIS and LBS*, 2009.

[3] R. Baumann, F. Legendre, and P. Sommer. Generic mobility simulation framework (GMSF). In *Proc. of the* 1^{st} *ACM SIGMOBILE Workshop on Mobility Models*, 2008.

[4] J. Bonneau and S. Preibusch. The Privacy Jungle: On the Market for Data Protection in Social Networks. In *The* 8^{th} *Workshop on the Economics of Information Security (WEIS)*, 2009.

[5] N. Cohn. Real-time traffic information and navigation: An operational system. *Intelligent Transportation Systems and VehicleHighway Automation*, 2009.

[6] M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In *Proc. of the* 3^{rd} *Pervasive Computing Conference (PERVASIVE)*, 2005.

[7] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy. Vanish: Increasing data privacy with self-destructing data. In *USENIX Security Symposium*, 2009.

[8] B. Gedik and L. Liu. Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. *IEEE Transactions on Mobile Computing*, 7, January.

[9] B. Gedik and L. Liu. Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. *IEEE Transactions on Mobile Computing*, 2008.

[10] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications ACM*, 42(2):39–41, 1999.

[11] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of the* 1^{st} *ACM Conference on Mobile Systems, Applications and Services (MobiSys)*, 2003.

[12] Hoh *et al.* Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. of the* 6^{th} *ACM Conference on Mobile Systems, Applications and Services (MobiSys)*, 2008.

[13] R. Hull, B. Kumar, D. Lieuwen, P. F. Patel-Schneider, A. Sahuguet, S. Varadarajan, and A. Vyas. Enabling Context-Aware and Privacy-Conscious User Data Sharing. In *Proc. of the* 5^{th} *IEEE Conference on Mobile Data Management (MDM)*, 2004.

[14] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley- Interscience, 1991.

[15] H. Kido, Yutaka, Yanagisawa, and T. Satoh. An Anonymous Communication Technique using Dummies for Location-based Services. In *Proc. of the* 2^{nd} *International Conference on Pervasive Services (ICPS)*, pages 88–97, 2005.

[16] L. Pareschi, D. Riboni, A. Agostini, and C. Bettini. Composition and Generalization of Context Data for Privacy Preservation. In *Proc. of the* 6^{th} *IEEE Conference on Pervasive Computing and Communications (PerCom)*, 2008.

[17] A. Pentland, D. Lazer, D. Brewer, and T. Heibeck. Using reality mining to improve public health and medicine. *Studies in Health Technology and Informatics*, 2009.

[18] A. Pingley, W. Yu, N. Zhang, X. Fu, and W. Zhao. CAP: A Context-Aware Privacy Protection System for Location-Based Services. In *Proceedings of the* 29^{th} *IEEE International Conference on Distributed Computing Systems*, pages 49–57, 2009.

[19] D. Riboni, L. Pareschi, and C. Bettini. Privacy in Georeferenced Context-Aware Services: A Survey. 2009.

[20] A. Sevtsuk and C. Ratti. Does urban mobility have a daily routine? learning from the aggregate data of mobile networks. *Journal of Urban Technology*, 2010.

[21] P. Shankar, V. Ganapathy, and L. Iftode. Privately querying location-based services with SybilQuery. In *Proc. of the* 11^{th} *ACM Conference on Ubiquitous computing (Ubicomp)*, pages 31–40, 2009.

[22] D. Wenliang and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of the* 9^{th} *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.