Location Privacy Protection for Smartphone Users Using Quadtree Entropy Maps

Xiaoen Ju and Kang G. Shin
Department of Electrical Engineering and Computer Science
University of Michigan
jux,kgshin@eecs.umich.edu

ABSTRACT

The ever-increasing popularity of location-based services (LBSs) poses a serious threat to users' location privacy. Numerous efforts have been made to protect users' location privacy and also to limit the degradation of service quality resulting from the additional protection layer. Most existing work, however, relies on a trusted anonymization server, which can itself become one source of untrustworthiness. In this paper, we present *EMP*^{2}—a new location privacy protection scheme based on a quadtree entropy map that enables users to protect their location privacy relying only on their smartphones. EMP^2 reduces the trusted computing base (TCB) and is cost-effective, because it eliminates the requirement of a trusted server from the protection system. In addition, EMP^2 accurately estimates the uncertainty of users' intended destinations from an adversary's perspective, and can properly adjust the protection level to defend against sophisticated inference attacks based on the correlation of user queries. Our evaluation of a prototype implementation demonstrates that EMP^2 can effectively protect users' location privacy with reasonable computation time and resource consumption. EMP^2 complements the common trusted-server-based protection mechanisms and provides smartphone users a secure environment for their use of LBSs.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Privacy, Algorithm, Experimentation

Keywords

Quadtree entropy map, location privacy, location-based service

1. INTRODUCTION

Location-based services (LBSs) respond to users' queries by making use of the location information contained in the queries, providing location-aware replies and hence better-customized service with no/little human intervention. As smartphones have become increasingly popular and resource-rich, LBSs have become more feature-rich and versatile. In general, LBSs improve users' daily lives, helping them find restaurants with their favorite menus, obtain just-in-time coupons from nearby shopping centers, and track their physical fitness [12]. However, it also poses a serious threat to users' location privacy. By collecting the location information contained in the LBS queries, an adversary who has compromised the LBS server can infer sensitive privacy information about the service recipients, such as their home locations, life styles, and health conditions.

Simple anonymization techniques, such as replacing real users' names with pseudonyms or removing all the user ID-related information before sending a query to the LBS server, cannot effectively protect the users' location privacy. In effect, a number of side channels (such as timestamps in queries, spatial correlations among consecutive queries, and vehicle-speed information) as well as sophisticated target-tracking algorithms (e.g., the Multi Target Tracking (MTT) algorithm [11,21]) allow the adversary to de-anonymize the LBS queries with high statistical confidence, and thus infer the query senders' privacy information.

There have been a number of proposals to mitigate the threat without significantly degrading the quality of LBS [23]. Most existing work, however, assumes the existence of a trusted server to protect the users' location privacy via (*i*) rendering the users' queries k-anonymous [4, 8, 10, 14, 18], (*ii*) weakening the adversary's ability to link multiple pseudonyms to the same user [1], or (*iii*) hiding the real queries behind the predicted ones [17]. These approaches, albeit effective, have some common problems: the trusted server is a single point of failure, and the assumed long-term trustworthiness and considerable cost related to the server make it difficult to deploy them. Other approaches have been proposed to overcome these drawbacks by eliminating the assumption about the trusted anonymization server, including k-anonymity [15, 20], m-unobservability [3], and cryptographic transformation [9, 16, 19]. However, a protection system that (*i*) can effectively weaken the adversary's ability to infer users' intended destinations, (*ii*) only relies on a small trusted computing base, (*iii*) is cost-effective, and

(*iv*) achieves high efficiency in utilizing limited smartphone resources still remains difficult to design, implement and deploy.

In this paper, we present EMP^2—quadtree *E*ntropy *M*ap-based location *P*rivacy *P*rotection—as a realization of the aforementioned location privacy protection. EMP^2 makes use of location entropy as its privacy metric. Briefly, location entropy represents the uncertainty of the user's intended destination, given that the user is at a certain location. This information can be derived from real-world users' mobility patterns. The protection system then calculates the entropy for every location in a certain area and then generates a multi-layer entropy map in advance, where each layer corresponds to a pre-defined resolution. With such a map stored in a user's smartphone, when the user is to send queries to an LBS server, EMP^2 dynamically builds a *quadtree entropy map* (QEM) on the smartphone covering his query area, estimates the uncertainty of his intended destination related to each query location, and selects appropriate nodes as cloaking boxes for location privacy protection. In particular, EMP^2 uses only those potential destinations that are likely to be visited all over the user's query area for calculating the location entropy. This approach makes an adversary unable to reduce the possible destination sets (as well as the corresponding uncertainty) by eliminating those destinations unlikely to be visited from some locations along the sequence of queries. As a result, EMP^2 accurately estimates the uncertainty from an adversary's perspective and can thus adjust the protection level accordingly for better protection.

The key contributions of this paper are:

1. a novel privacy protection scheme that relies only on users' smartphones to achieve location privacy protection for LBS queries, thus reducing the trusted computing base and the cost related to the entire protection system;
2. a QEM-based approach that can accurately estimate the location entropy observed by an adversary and thus adjust the uncertainty level of the query locations to achieve better protection; and
3. a thorough evaluation of the proposed approach with real-world user traces, demonstrating its effectiveness and practicality.

The rest of the paper is organized as follows. Section 2 discusses the system, threat, and trust models. Section 3 presents our EMP^2 model. In Section 4, we evaluate the performance of a prototype of EMP^2 system using real-world user traces, followed by a discussion of the state-of-the-art of location privacy protection in Section 5. Section 6 concludes the paper.

2. SYSTEM, THREAT, AND TRUST MODELS

We first describe a generic LBS system model, and then present the threat model, clarifying our assumptions on the adversary's goal and capabilities. We conclude this section with a discussion on the trusted components in our proposed scheme.

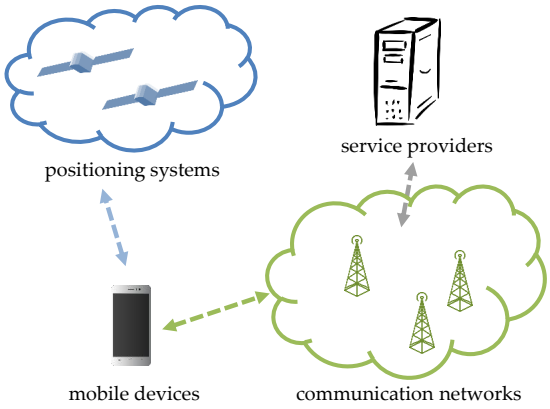


Figure 1: System model.

2.1 System Model

Figure 1 illustrates our simple and generic system model. Users utilize their mobile devices (e.g., smartphones) to send queries to LBS servers. The locations in the queries are obtained via positioning systems, e.g., Global Positioning System (GPS). The queries, as well as their responses from the LBS servers, are transmitted via communication networks, e.g., 3G networks. LBS servers are service providers, replying to queries with well-tailored responses based on the location information in the queries. Note that in our model, mobile-device users send queries wirelessly to LBS servers, without any trusted anonymization server in between.

2.2 Threat Model

We assume that the adversary's goal is to obtain the information about where a user has visited during a trip. (A trip is defined as the movement from a source to a destination.) In order for an adversary to achieve this goal, we assume that he can compromise an LBS server and access all the users' information stored in the server, such as the IP address and location information in each query. In addition, we assume that an adversary, after retrieving one trace (i.e., a sequence of locations) corresponding to a user's trip from the LBS server, can conduct reasonably sophisticated inference attacks.[1] Specifically, we assume that the user's destination is likely to be visited from each location along the trace. Resorting to this empirical fact, an adversary can conduct more accurate inference attacks by only considering the intersection of likely-to-be-visited destination sets of all location updates as the user's possible destination set. This assumption is reasonable in that, by doing so, the adversary reduces the uncertainty about the user's destination with little overhead.

Query-sending patterns also have impact on the threat to which a user is exposed. We assume that a user could begin sending queries to an LBS server at any time during his trip (i.e., not necessarily at the starting point of the trip), and

[1]How to retrieve a trace from potentially anonymous location updates has been covered by previous work [11].

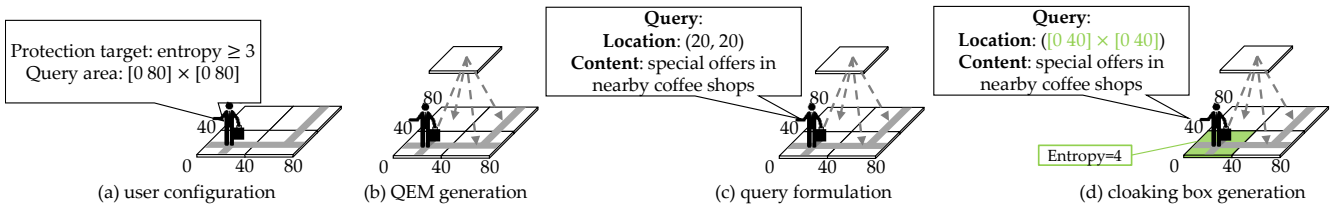


Figure 2: A high-level view of EMP^2. Step 1: configuration of EMP^2; Step 2: QEM generation; Step 3: user-query formulation; Step 4: cloaking box generation, replacement of accurate location information with cloaking box, and query sending to an LBS

could also stop at will (i.e., not necessarily after arrival at the destination). Thus, the adversary cannot simply wait until a user stops sending queries and then infer the user's destination only from a small number of points of interest (POIs)[2] in the vicinity of the location where his last query was sent.

2.3 Trust Model

Primarily, we trust users' smartphones to be secure and will not leak users' location information to LBS servers via side channels outside the control of our protection system.[3] Besides, our approach is based on entropy maps (cf. Section 3.3). Here we would like to stress that by relying on entropy maps—the *contents* provided by a map-generation server, our protection system does not introduce any *additional* trust model into the current LBS architecture. From a user's perspective, an entropy map resembles a conventional geographic map. When a user makes use of a map service such as Google map, he trusts the maps provided by the service provider. Similarly, our protection system trusts the entropy maps downloaded from the map-generation server. Generally speaking, in the LBS architecture, users trust contents provided by the service providers. The service provider, however, can still make use of the collected location information to conduct inference attacks. Trust on the contents by a server does not imply trust on the appropriate use of collected information at the server side.

3. EMP^2 METHODOLOGY

This section details the design of our EMP^2 model. First, we describe, at a high level, how EMP^2 protects users' location privacy, followed by an introduction to the location privacy metric used in EMP^2. We then discuss the concept of entropy map in general and QEM in particular. Finally, we discuss the algorithm for cloaking box generation.

3.1 A High-Level View of EMP^2

Figure 2 illustrates the high-level workflow of the EMP^2 model. Before sending queries to LBS servers, the user needs to configure EMP^2 with his privacy protection requirement (i.e., the entropy threshold) and his current query area—a predicted region that bounds the locations where he could possibly send queries. The user can designate the region by simply specifying a center and a radius. Based on the configuration, EMP^2 generates a minimum QEM that covers the area. Then for each query, EMP^2 generates a cloaking box (i.e., a rectangular area on the map) that covers the exact location of the query and also satisfies the user-specified privacy protection level. EMP^2 then replaces the location information with the cloaking box, and sends it to the server.

[2]The words destination and POI are used interchangeably in this paper.

[3]Researchers have proposed numerous approaches to securing users' handsets, and interested readers are referred to related references [2].

3.2 Location Privacy Metric

We use entropy as the location privacy metric. The general form of location entropy is widely accepted in the location privacy research community, although different researchers use different attributes to quantify it [1,3,17,26]. Similar to LISA [3], we use the conditional probability of visiting POIs from a given location to measure the location entropy.

DEFINITION 1. *The entropy of a location j is defined as*

$$e = -\sum_{i=1}^{n} p_{i|j} \log_2 p_{i|j}$$

with $p_{i|j}$ being the conditional probability of visiting the i-th POI given that the user is at location j.

This metric well represents a user's location privacy level when he sends a query from his current location to an LBS provider. This is because an adversary can hardly infer the user's privacy without the ability to relate a location to any POI. Furthermore, this metric can be easily applied to our smartphone-based protection approach, because the entropy can be calculated on the smartphone. In contrast, other well-known metrics, such as k-anonymity, usually need a trusted server [10] or the cooperation of nearby unknown users [7], thus rendering them unsuitable for our approach.

3.3 Entropy Map

In EMP^2, a quadtree covering the query region is dynamically generated using layered entropy maps.

DEFINITION 2. *An entropy map is a map with each grid containing the entropy of the location represented by the grid.*

An entropy map is stored in EMP^2 as a matrix, each element of which represents a grid on the map and consists of the location information (e.g., the local Cartesian coordinate, or longitude and latitude) as well as the conditional probability

of visiting each POI from that location. Similar usage of such a matrix can be found in [1, 3, 17, 26].

One single-layer entropy map—with each grid representing a finest-resolution grid on the map—is insufficient for protecting users' location privacy for the following two reasons. First, the entropy of the finest-resolution grid covering a user's accurate location may not exceed the entropy threshold, in which case a lower-resolution grid should be used with the hope that it could increase the entropy by covering a broader area. As a result, we need *multi-layer* maps with nodes of different resolutions. Second, multi-layer maps are necessary for defending against attacks employing the correlation among consecutive queries. By combining the information in consecutive queries, an adversary can reduce the uncertainty related to each location by considering only the POIs that are likely to be visited from all locations in the queries. To mitigate this effect, our system uses only the intersection of likely-to-be-visited POIs all over the query area for entropy calculation. However, note that the POI intersection for the finest grid layer may not be sufficient for achieving users' protection target. This necessitates the intersection calculation for lower resolution layers, which in turn requires the use of multi-layer maps.

The above reasoning justifies the need of a hierarchical multi-layer entropy map that can avoid the aforementioned drawbacks.

3.4 Quadtree Entropy Map

Quadtree [6] is a data structure in which each internal node has up to four children. It is a well-studied data structure for efficiently partitioning data in a two-dimensional space, and is widely used in geographic information and computer vision systems. Our use of quadtree for location privacy protection is inspired by Xu and Cai's work [26] which proposes the use of a pyramid structure in a trusted anonymization server to store users' footprints and generate cloaking boxes. In EMP^2, we employ quadtrees to organize entropy maps. Described below is the generation of a quadtree entropy map (QEM).

3.4.1 Trace Collection

The generation of entropy maps, or more generally, the calculation of location entropy for privacy protection, is based on trace collection. We assume that anonymized trace collection is done by some well-reputed wireless service providers (WSPs). They could select a short period of time (e.g., several days) in one or several years, during which they establish a map-generation server, encourage volunteers to contribute to the trace collection, and ask them to periodically update their accurate location information. Specifically, the collected information consists of timestamps, user identifiers, and locations. Since WSPs have long been able to generate users' traces based on their collected data and it is common for subscribers to trust these large well-reputed companies not to abuse their data, our model does not incur additional requirements on the operation of WSPs, or the trace-collection-service volunteers' trust on WSPs. In particular, note that users of the entropy map need not trust the map-generation server. In addition, since POI distribution, road connectivity, and people's travel patterns remain relatively stable, the entropy map, like a normal geographic map, need not be updated frequently. Hence, the duration, the cost, and the risk of maintaining an entropy map-generation server are significantly lower than those related to a trusted anonymization server which must always be operational.

3.4.2 Generation of Quadtree Nodes

After collecting traces, the map server generates QEM nodes and then distributes them to users' smartphones. Specifically, it pixelates a geographic map into grids according to the predefined finest resolution, and then generates one record per grid consisting of a mapping from each POI to collected traces passing through the grid and ending at that POI. The server-side mapping contains the detailed information including trace IDs for accurate counting of traces and calculation of the visiting probability. The smartphone-side mapping, on the contrary, only relates each POI with the total count of traces ending at that POI to avoid information leakage via the deployment of entropy maps. Consequently, the server-side entropy map nodes above the finest-resolution level can be generated iteratively due to the availability of the detailed information. The smartphone-side entropy map nodes are generated according to their corresponding server-side nodes at the same level.

However, storing all the nodes on the smartphone is resource-demanding. As an optimization, we propose an alternative lightweight approach, which only stores the finest-resolution layer of the map and uses a simplistic node merging method to calculate the probability of visiting POIs from lower-resolution grids. Specifically, it generates internal nodes by assigning the total of POI visit counts in the child nodes to the corresponding POI record in the parent nodes. We compare the performance and resource consumption of these two approaches in Section 4.

3.4.3 QEM Generation

On the smartphone side, when a user indicates a query area, EMP^2 sets the root node to be the center of the region, and dynamically generates a QEM covering the region in a recursive manner. Regarding the original EMP^2, all nodes for the construction of a QEM are ready for use (i.e., generated by the map-generation server) so that the tree can be easily built. As for the lightweight EMP^2 approach, the aforementioned simplistic merging method is employed to construct QEM nodes other than those belonging to the finest-resolution layer, along with the generation of the tree itself.

3.5 EMP^2 Protection Mechanism

We now describe how to use a QEM to generate cloaking boxes and protect users' location privacy. Briefly, EMP^2 first selects a candidate POI set that could be used for location entropy calculation over the query area, then generates one cloaking box for each query using the selected POI set.

3.5.1 Set of Candidate POIs

As mentioned in Section 3.1, for each query, EMP^2 generates a cloaking box that satisfies the predefined protection threshold, and uses it as the location information sent to an LBS. It is problematic, as described Section 3.3, to use all POIs for entropy calculation and to always select the finest

Algorithm 1 Candidate POI set selection

Require: QEM Q, entropy threshold e, percentage threshold p of nodes having entropy $\geq e$, likely-to-visit threshold vp_{thld}

function CandPOISelect(Q, e, p, vp_{thld})
$result = Null$
for $l = 0$ to Height(Q) **do**
  if CheckIntersec(l, e, p, vp_{thld}, $result$) = **true then**
    break
return $result$

function CheckIntersec(l, e, p, vp_{thld}, $result$)
/* calculate the POI intersection of level l */
$result = \cap_j \{i : p_{i|j} > vp_{thld}, node_j \in \text{level } l\}$
/* check whether the selection criterion is satisfied */
if *CheckRoot* **then**
  $entropy$ = GetEntropy($root$, $result$)
  if $entropy < e$ **then**
    return false
if *CheckCurrentLevel* **then**
  $pass = 0$
  for each node j at level l **do**
    $entropy$ = GetEntropy(j, $result$)
    if $entropy \geq e$ **then**
      $pass$++
  if $\frac{pass}{\text{number of nodes at level } l} < p$ **then**
    return false
return true

function GetEntropy(j, $pois$)
Normalize(j, $pois$) /*so that $\sum_i p_{i|j} = 1$ for $i \in pois$*/
return $entropy = -\sum_i p_{i|j} \log_2 p_{i|j}$ for $i \in pois$

grids reaching the protection level in the QEM as cloaking boxes. Instead, only the POIs that are likely to be visited along a sequence of query locations should be used to calculate the entropy for cloaking box generation.

Given that the locations at which a user may send queries are unpredictable, EMP^2 first calculates the per-level intersections of likely-to-visit POIs over the query area, and then selects one intersection that passes a certain candidate POI selection criterion as the candidate POI set. Algorithm 1 and Figure 3 illustrate the candidate POI set selection procedure. This procedure is executed in the initialization stage of EMP^2, before the first query is sent along the trace.

Since the candidate POI selection criterion is designed to facilitate the generation of high-quality cloaking boxes, we will first discuss our cloaking box generation approach, and then revisit the design of the POI selection criterion.

3.5.2 Cloaking Box Generation

With a selected candidate POI set, for each query, EMP^2 uses a bottom-up approach to generate a cloaking box covering the accurate location of the query. Specifically, EMP^2 first pinpoints the QEM node with the finest possible resolution that covers the location, then repeatedly checks the following two conditions from that node up to the root node, and uses the first node satisfying both conditions as the cloaking box.

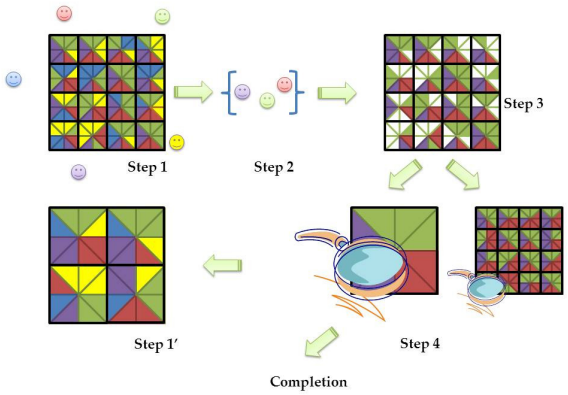


Figure 3: Candidate set selection. Step 1: Locate the current level of QEM. Each color inside a grid represents the possibility of visiting the POI (a Smiley face) with that color from the grid. Step 2: Calculate the intersection of likely-to-be-visited POIs. Step 3: Eliminate unlikely-to-be-visited POIs from QEM. Step 4: Normalize POI visiting possibility and check against selection criteria. The left branch illustrates root node checking, the right branch current level checking. Step 1': If checks fail at Step 4, then restart the whole process at a higher level at QEM. Otherwise the process completes.

Condition 1 (Subset condition). *The candidate POIs are a subset of the likely-to-visit POI set of this node (denoted as node j); that is,*

$$\forall i \in \{candidate\ POIs\}, p_{i|j} \geq vp_{thld}. \quad (1)$$

Condition 2 (Entropy condition). *The entropy of the node calculated by the candidate POIs is larger than, or equal to, the threshold value, that is,*

$$e = -\sum_{\{i|POI_i \in candidate\ POI\ set\}} p_{i|j} \log_2 p_{i|j} \geq threshold. \quad (2)$$

The rationale behind the first condition is that, since the POIs in the candidate set are those used by EMP^2 for entropy calculation, they need to be likely-to-visit POIs from the selected node. Otherwise, the adversary can eliminate those unlikely to be visited POIs and recalculate the entropy, which may diverge from EMP^2's expected value. The second condition is straightforward, because EMP^2 needs to satisfy the user's protection requirement.

3.5.3 Candidate POIs Selection Criterion

A properly selected set of candidate POIs enables EMP^2 to generate high-quality cloaking boxes that can preserve location privacy and remain small in size. Specifically, to meet the entropy condition in Section 3.5.2, the set of candidate POIs should be large enough for the entropy of a node calculated via this set to properly represent the uncertainty related to that location. Considering the subset condition, the selected set should not be too large to unduly rule out otherwise valid QEM nodes only because the former is not a subset of the latter's likely-to-visit POIs.

Two selection criteria are studied, as illustrated in function *CheckIntersec* in Algorithm 1. For the *CheckRoot* option (lines 12-15), we target the intersection selection criterion at the root node, and select an intersection if the entropy of the root calculated by this POI set exceeds the threshold. The rationale behind this option is that the user's location privacy is expected to be protected at least by using the root node as the cloaking box. This option is the least restrictive in the sense that it only sets the worst-case upper bound to the top-level node, without verifying if the nodes at the selected level could be used as cloaking boxes. This loose check also leads to a relatively small candidate POI set, which makes the subset condition easy to be satisfied but the entropy condition hard to be satisfied. Another option (lines 16-23 in Algorithm 1 with $p = 100\%$) is to target the criterion at the level corresponding to the current intersection of POIs, and select the intersection if at that level, each node can satisfy the entropy threshold. This option guarantees that the sizes of the cloaking boxes are upper-bounded by the resolution of the selected level. Such a rigorous check leads to a large candidate POI set, favored by the entropy condition but disfavored by the subset condition. Table 1 summarizes these two options. The candidate POI selection criterion used in our current implementation falls in between these two extremes, requiring at least 50% of the nodes at the selected level, as well as the root node, to satisfy the threshold (i.e., setting both *CheckRoot* and *CheckCurrentLevel* to be true and $p = 50\%$).

4. EVALUATION

This section presents a thorough evaluation of the EMP^2 system. We first discuss the real-world traces used for entropy map generation and EMP^2 performance analysis. Then, we evaluate the performance of EMP^2 and its resource consumption. Our prototype is implemented in Java. The evaluation is conducted on a Dell workstation with Intel Core2 Quad CPU 3.0 GHz and 8 GB RAM.

4.1 Real-world Traces

The real-world traces we use in the evaluation are obtained from the LiveLab project [22] of Rice University. The data available to us consists of 11 users' location logs for one month, from which we retrieve 249 traces on a 10km×10km map centered at Rice University, with the corresponding query areas completely contained in the map to avoid the border effect on the evaluation results. The entropy maps are generated as discussed in Section 3.4.2, with the finest-resolution set to 40m×40m.

4.2 Performance Evaluation

In this section, we first present the metrics that we use for performance evaluation, then compare the overall performance of EMP^2 with several alternative approaches, demonstrating that EMP^2 can achieve the best protection effect without unduly decreasing the quality of LBS.

4.2.1 Metrics

Two metrics are used to evaluate the effectiveness of EMP^2. The first metric is the protection success ratio, that is, the probability that the entropy of a cloaking box exceeds the target entropy threshold. Two types of success ratio are employed in our evaluation. On one hand, if a protection system fails to generate a cloaking box that can satisfy the target protection level, it records one protection failure. Otherwise, the protection system records a success. We call the related success ratio the *expected success ratio*, as this is the expected success ratio from the perspective of the protection system. On the other hand, after observing a complete trace, the adversary can re-calculate the entropy of each cloaking box using the intersection of likely-to-visit POIs of all the observed cloaking boxes. If the adversary-calculated entropy of a cloaking box falls below the entropy threshold, we record it as an *observed* failure. Otherwise, we record a success. This success ratio is called the *observed success ratio* since it is observed from the adversary's perspective. The two success ratios are different due to the use of different POI sets for entropy calculation: while the protection system calculates entropy at the query time, the adversary does it *a posteriori*, with information about all the cloaking boxes along a trace available to him.

The second metric is the size of cloaking box. Intuitively, the larger the cloaking box, the more irrelevant information the user receives from the LBS server, thus degrading users' experience unless an additional local filter is used. Using large boxes also unnecessarily drains the phone battery. Thus, a protection system is considered advantageous over another if the former could use smaller cloaking boxes to achieve protection similar to the latter.

4.2.2 Performance Comparison

In order to demonstrate the effectiveness of EMP^2, we compare it with several alternative protection mechanisms. We first study the effect of applying no protection at all. That is, we reveal the accurate location (i.e., the best-resolution grid, 40m×40m) associated with each query to the LBS server. This approach is used as a baseline approach which represents the intrinsic protection provided by the road paths and is referred to as "no protection" henceforth. The second approach is to fix the cloaking box size to a predefined constant. In other words, we statically pixelate the map into fixed-sized grids, and always use the grid covering the user's query-sending location as the location information for the LBS server. The rationale behind this approach is that naive information hiding (in this case hiding the least significant bits in the location data) can arguably provide some protection to users. This approach is referred to as "naive static" hereafter. Intuitively, the static nature of this approach makes it an overkill for locations of which the associated entropy is already very high, but becomes ineffective for those locations surrounded by low entropy areas and needed better protection by using larger cloaking boxes. In effect, the "no protection" approach is a special case of the "naive static" approach, with the resolution fixed at the finest resolution of the map.

While the first two approaches are static in the sense that the size of cloaking boxes is fixed *a priori*, the third approach is dynamic since it could adjust the size of cloaking box according to the user's entropy threshold. Specifically, it only assumes the existence of the best-resolution layer of the entropy map and builds a QEM as the lightweight EMP^2 approach (cf. Section 3.4). In addition, this approach does not take into consideration the correlation among the queries, and simply uses all POIs of a grid to calculate its

Table 1: POI Selection Criterion Summary

	CheckRoot	CheckCurrentLevel
Worst case guarantee	Loose (at root node)	Tight (at current level)
Candidate POI set size	Small	Large
Subset Cond. Friendly	Yes	No
Entropy Cond. Friendly	No	Yes

entropy. We call it "isolated" in the sense that it takes an isolated view of the to-be-protected query location instead of using POI intersections. The fourth approach is our original EMP^2, and the fifth is its lightweight alternative.

For the purpose of evaluation, we assume that users send queries on a regular basis. We have tuned the distance between two consecutive queries ranging from 40m to 240m, and find that it has little impact on the performance of EMP^2. Moreover, considering the modest number of traces we have, we set vp_{thld} to 0. In other words, a POI is considered likely to be visited from a location unless the conditional probability of visiting this POI from the location is 0. In reality, on an entropy map generated with a sufficient number of traces, the above conditional probability should be almost always greater than 0, and it is reasonable for the protection system and the adversary to set vp_{thld} to a positive value.

Figure 4a shows the protection effect of the five protection approaches. For all entropy thresholds, the two EMP^2 approaches achieve the best observed success ratios. Their performance is much better compared to the first two static approaches, and also better than the "isolated" dynamic approach—the strongest competitor to EMP^2—by 2.52% for the least demanding protection target (i.e., entropy=1) and by 10.12% in the most demanding case (i.e., entropy=5).

One clear trend from Figure 4a is that, for relatively low entropy thresholds like 1 and 2, all of the five approaches achieve a high success ratio. But, for more demanding entropy thresholds like 4 and 5, the success ratios of all approaches decrease. This is expected because the higher the protection requirement is, the harder it is for the protection system to meet that requirement. Another observation is that the decrease of success ratio becomes significant when the entropy threshold rises from 4 to 5. From a detailed analysis of the entropy of each trace (calculated as the average entropy of all the best-resolution grids along the trace), we found that the entropy value of over 70% of the traces lies between 4 and 5, which explains the above observation: the protection task becomes more challenging when the trace entropy is not high enough as compared to the protection threshold.

Another observation is that for all of the five entropy thresholds, the performances of the two EMP^2 approaches are very similar. Detailed comparisons show that the absolute difference between the observed success ratios of the two approaches varies from 0.003% to 1.28%, from which we conclude that the lightweight approach is a good approximation of the original EMP^2, preserving the protection quality while eliminating the considerable storage overhead.

In all cases, EMP^2 family preserves the minimum difference between the expected and observed success ratios. Figure 4b shows the relative difference between the two ratios (i.e., the ratio of the absolute difference between the two to the expected success ratio). Clearly, EMP^2 family outperforms the other three approaches in the sense that it has the most accurate estimation of the adversary-observed entropy values and can thus provide the appropriate level of protection.

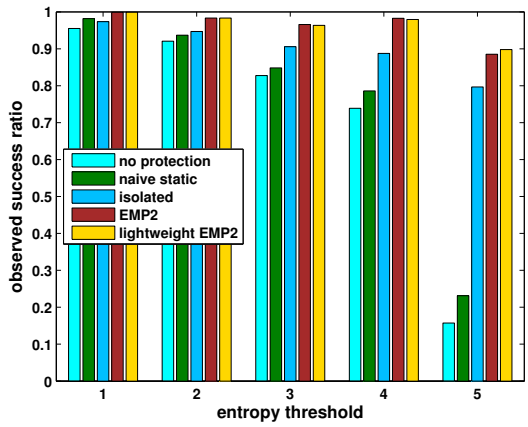
Figure 4b also shows that the differences of the first three approaches are strictly increasing with the entropy threshold. The EMP^2 family's expected success ratios are, on the other hand, always good estimates of the adversary-observed values in all cases. This phenomenon can be explained as follows. When the entropy threshold is low, the protection target can be relatively easily achieved from the protection systems' aspect and the entropy of the generated cloaking boxes is usually higher than the target. Thus, the entropy value observed by an adversary may well remain above the protection target even after eliminating the POIs not in the POI intersection along the trace. Consequently, the observed success ratio is largely preserved. Nevertheless, for a demanding protection target, the entropy value of the generated cloaking boxes is usually slightly higher than the protection target, and eliminating POIs may easily make it fall below the threshold, thus decreasing the observed success ratio. This phenomenon also justifies our use of the intersection of POIs in the design of EMP^2.

As for the size of cloaking box, Figure 4c shows the average size of cloaking boxes in different entropy protection thresholds. We can see that the "no protection" approach provides the best cloaking box size, which is straightforward since there is no protection whatsoever. The "naive static" approach also has good cloaking box size simply, because it is statically configured to be 100m×100m. The obvious drawback of this approach is its inflexibility, as is indicated at the beginning of this subsection. For the remaining three approaches, the "isolated" approach always comes with smaller boxes than EMP^2 and lightweight EMP^2. On average, the size of cloaking boxes of EMP^2 family is 1.7 times larger than that of "isolated" approach. This is a reasonable cost EMP^2 pays for better protection.

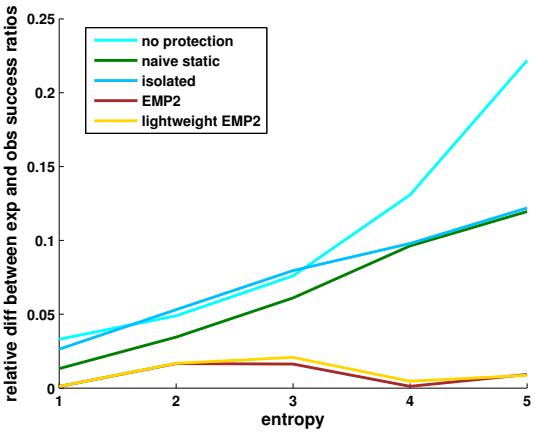
4.3 Resource Consumption

The resource consumption remains a key issue related to the practicality of smartphone-based approaches due to the limited available resources on smartphones. In our case, we measure the CPU usage, the main memory consumption, and also the storage requirement of our EMP^2 approach, and compare it with the "isolated" dynamic protection approach. The CPU and memory usage are measured three times.

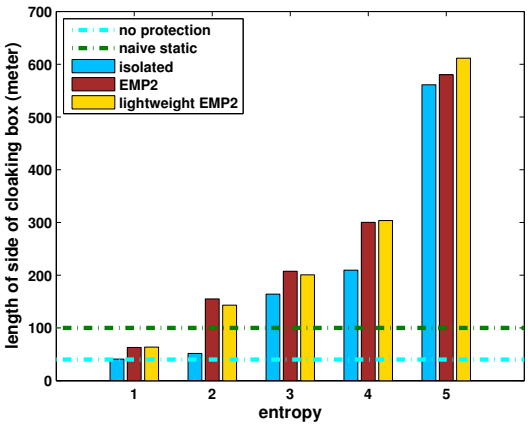
Figures 4d and 4e compare the startup time and per-query



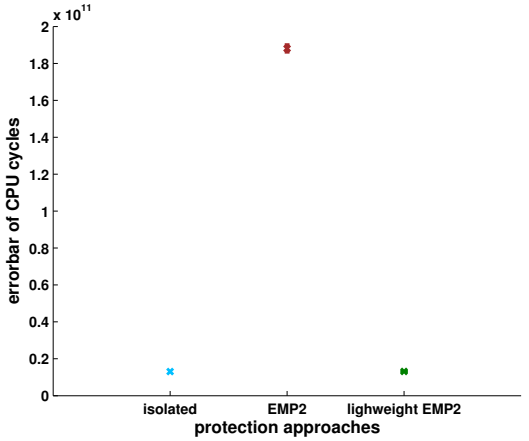
(a) Observed success ratio



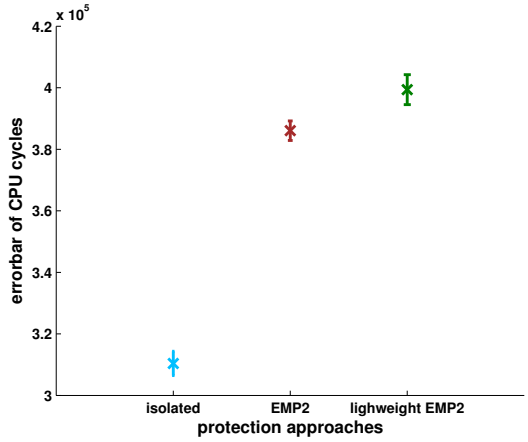
(b) Relative difference between expected and observed success ratios



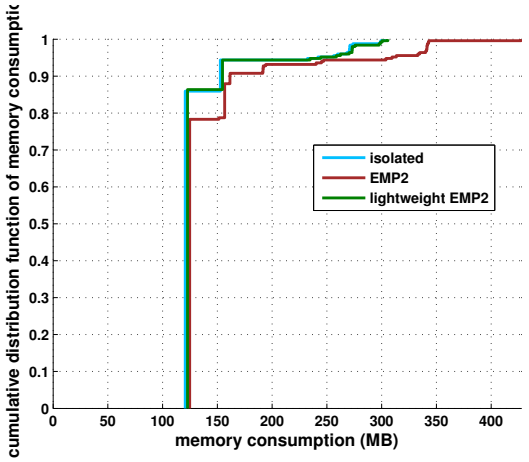
(c) Cloaking box size



(d) Startup time comparison



(e) Per-query protection time comparison



(f) Memory consumption

Figure 4: Performance and resource consumption comparisons

protection time of the isolated approach and two EMP^2 approaches. The startup time contains the time needed to build a QEM covering the user's current query area, and the time to find the proper candidate POI set in the EMP^2

case. The isolated approach and lightweight EMP^2 have a similar startup time, with lightweight EMP^2 incurring a 0.24% overhead on average. With a 1.9GHz processor commonly equipped on a modern smartphone, the CPU cycles roughly translate to a startup time of 7 seconds. We consider it tolerable since (1) this is an one-time overhead on an unoptimized prototype, and (2) users can simply start the protection system a little ahead of time so that the startup time can be hidden by the usage of other applications and will not affect the query-protection time. As for the original EMP^2, it incurs a 1436.4% overhead, taking significantly longer time to start up due to the need of searching and loading server-generated nodes into the QEM. In terms of the per-query protection time, Figure 4e shows that lightweight EMP^2 needs 3.99×10^5 cycles (i.e., 210 microseconds on a 1.9GHz processor) to protect one query, incurring a 28.67% overhead compared to the isolated approach. The additional cost of EMP^2 is due mainly to the fact that it needs to conduct an intersection operation (i.e., selecting a subset of POIs from a grid according to the candidate POI set) before each entropy calculation.

As for main memory consumption, Figure 4f demonstrates that the isolated approach and EMP^2 have very similar memory footprints. The average per-trace memory consumption is between 134MB and 142MB for the three approaches.

The major storage space consumption related to the smartphone-side privacy protection systems is caused by the entropy maps. In the case of the isolated approach and lightweight EMP^2, they both need the finest-resolution layer of the entropy map covering the 10km$\times$10km area, which takes 15MB storage space. This is a reasonable storage requirement for modern smartphones. As for the original heavyweight EMP^2, it needs to store all the multi-layer maps on the smartphone, the accumulated size of which is 745MB.

Based on the above comparisons, we conclude that the original EMP^2 approach, though accurate, incurs significant overhead in both startup time and storage space. The lightweight EMP^2 achieves a similar protection effect with reduced overhead, thus becoming effective and practical.

5. RELATED WORK

Trusted-Anonymization-Server-Based Approaches. Most existing studies on the privacy protection issue of LBS assume the existence of a trusted server residing between the smartphone users and the potentially malicious LBS servers for achieving location privacy protection. K-anonymity is one widely used metric, requiring the location region revealed to LBS servers to contain at least k users so that the query sender is indistinguishable from the remaining $k-1$ users [4, 8, 10, 18]. Beresford and Stajano [1] introduced the concept of mix zones. Since no users update their location information to LBSs in a mix zone and they are assigned new pseudonyms when they leave the zone, mix zones weaken the adversary's ability to relate the outgoing users' pseudonyms with their incoming ones. In addition, the expected distance error has also been used to quantify the accuracy at which an adversary can estimate a user's location and to guide the path confusion procedure for privacy protection [13]. CacheCloak [17] achieved privacy protection without loss of location accuracy from the LBS' perspective by caching LBS responses along the predicted paths and using the cached information to reply to users' queries. Xu and Cai proposed a feeling-based location privacy model, used entropy to measure the popularity of a region, and took a quadtree-style approach for the prevention of an adversary from relating LBS queries to specific users [26].

The major difference between the above approaches and EMP^2 is that EMP^2 does not rely on a trusted anonymization server, thus reducing the TCB and the cost related to the protection system. In addition, the privacy metric used in EMP^2 is different from those approaches. We believe that the location entropy calculated as the uncertainty about the user's intended destinations properly quantifies the uncertainty level of a location.

Non-Trusted-Server-Based Approaches. CAP [20] used a quadtree to maintain the road-density information and conducted Hilbert curve mapping and perturbation for achieving k-anonymity. Compared with the road density, the entropy map in EMP^2 better quantifies the location uncertainty because it takes the real-world users' mobility patterns into consideration.

SMILE [16] applied k-anonymity to measure and configure the users' privacy level for the use of encounter-based LBSs, protecting users' privacy by selecting the prefix length of the location hash value so as not to reveal encounter involvements to untrusted servers. While SMILE proves successful for users of encounter-based LBSs, EMP^2 targets a broader range of users, maintaining a high uncertainty level from the LBS providers' perspective.

LISA [3] used m-unobservability as its location privacy metric, weakening the adversary's ability to infer users' privacy based on the distinguishability of POIs the users visit. While LISA employs an extended Kalman filter to generate cloaking boxes, EMP^2 uses QEM for the same purpose. Moreover, EMP^2 ensures that all the POIs used for the entropy calculation of each cloaking box are likely-to-be-visited along the trace, thus achieving better protection by accurately estimating the uncertainty observed by an adversary.

Shokri *et al.* [25] formulated location privacy protection as a Stackelberg game and, given a user's service quality constraints, achieved the optimal protection mechanism against an adversary conducting the optimal inference attack, with a focus on LBSs requiring sporadic location updates. Our work, in contrast, takes into consideration the correlation among continuous location updates.

LP-Guardian [5] proposed a novel and promising protection framework, achieving fine-grained per-app location privacy protection by only using a user's smartphone. We are working towards augmenting EMP^2 with the same fine-grained support, specifically, by deploying per-app entropy maps. Such enhancement, however, imposes high pressure on storage and demands better compression algorithms.

Some recent approaches made use of the private information retrieval (PIR) for achieving stronger and provable location privacy. For example, Ghinita *et al.* [9] demonstrated the

practicality of applying computational PIR to the protection of LBS users' privacy by solving the nearest neighbor (NN) search in a theoretical setting. The advantage of this approach is that it does not disclose any spatial information, and thus prevents any type of location-based attack. Direct comparison between our approach and PIR-based ones is difficult because the privacy metric as well as the system design is completely different. In general, EMP^2 protects users' location privacy in a more efficient and cost-effective way, but with a weaker privacy standard.

Location Privacy Metric Analysis. Shokri *et al.* [24] designed a formal framework for the analysis of location privacy protection mechanisms, pointing out that the expected estimation error of the adversary is the appropriate metric for location privacy protection. Nevertheless, since destination POIs are not necessarily available at the query-sending time, estimation error may not be appropriate for privacy protection systems. For example, a user is going from one place to another, and he will stop by a coffee shop during the trip. Which coffee shop the user will visit may depend on the information from the LBS (e.g., about the coupons or the special offers of nearby coffee shops). Such on-the-fly destination selection could render the estimation error based metric difficult to be applied directly inside protection systems, but has no impact on our QEM approach. However, the unsatisfactory correlation between entropy and estimation error is worthy of further investigation. We believe this is a promising direction for novel privacy metric design.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented EMP^2—an entropy map-based location privacy protection system. It eliminates the need of a trusted anonymization server for protecting LBS users' location privacy, and generates cloaking boxes by using a dynamically-built quadtree based on layered entropy maps. EMP^2 reduces the trusted computing base and the cost of the location privacy protection system, and achieves a high protection success ratio even if the adversary acquires an entire history of users' queries from a compromised LBS server. A thorough evaluation of a prototype implementation demonstrates EMP^2's effectiveness and practicality.

We plan to study the effect of the dynamics of entropy map on the overall protection quality. Empirically, users' mobility patterns vary with the time of day. For example, office buildings are visited much more frequently during the daytime than the nighttime. By generating detailed entropy maps to capture these variations in the time frames, the protection quality of EMP^2 may be further improved.

Acknowledgments

The work reported in this paper was supported in part by NSF under Grant CNS-1114837. The authors would like to thank Lin Zhong and Clayton Shepard of Rice University for the contribution of real-world users' location data. Also, they are thankful to Zhigang Chen and Xin Hu for their valuable comments on this work.

7. REFERENCES

[1] A. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[2] A. Bose, X. Hu, K. Shin, and T. Park. Behavioral detection of malware on mobile handsets. In *MobiSys*, pages 225–238. ACM, 2008.

[3] Z. Chen, X. Hu, X. Ju, and K. G. Shin. LISA: Location information scrambler for privacy protection on smartphones. In *First IEEE Conference on Communications and Network Security (IEEE CNS 2013)*. IEEE, 2013.

[4] A. Deutsch, R. Hull, A. Vyas, and K. Zhao. Policy-aware sender anonymity in location based services. In *ICDE*, 2010.

[5] K. Fawaz and K. G. Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 239–250, New York, NY, USA, 2014. ACM.

[6] R. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[7] J. Freudiger, M. H. Manshaei, J.-P. Hubaux, and D. C. Parkes. On non-cooperative location privacy: a game-theoretic analysis. In *CCS'09*, pages 324–337. ACM, 2009.

[8] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, January 2008.

[9] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD*, pages 121–132. ACM, 2008.

[10] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*. ACM, 2003.

[11] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. *Security in Pervasive Computing*, 3450/2005:179–192, March 2005.

[12] B. Harrison and A. Dey. What have you done with location-based services lately? *IEEE Pervasive Computing*, 8(4):66–70, Oct.-Dec. 2009.

[13] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *SecureComm*, pages 194–205. IEEE, 2005.

[14] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *MobiSys*. ACM, 2008.

[15] H. Hu and J. Xu. Non-exposure location anonymity. In *ICDE*, 2009.

[16] J. Manweiler, R. Scudellari, and L. P. Cox. Smile: encounter-based trust for mobile social services. In *CCS'09*, pages 246–255. ACM, 2009.

[17] J. Meyerowitz and R. R. Choudhury. Hiding stars with fireworks: Location privacy through camouflage. In *MobiCom*. ACM, 2009.

[18] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB*, pages 763–774. ACM, September 2006.

[19] S. Papadopoulos, S. Bakiras, and D. Papadias.

Nearest neighbor search with strong location privacy. In *VLDB*, September 2010.

[20] A. Pingley, W. Yu, N. Zhang, X. Fu, and W. Zhao. Cap: A context-aware privacy protection system for location-based services. In *ICDCS*, pages 49–57. IEEE, June 2009.

[21] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

[22] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. Livelab: Measuring wireless networks and smartphone users in the field. In *HotMetrics*, 2010.

[23] K. G. Shin, X. Ju, Z. Chen, and X. Hu. Privacy protection for users of location-based services. *IEEE Wireless Communications Magazine*, 19(1):30–39, February 2012.

[24] R. Shokri, G. Theodorakopoulos, J.-Y. L. Boudec, and J.-P. Hubaux. Quantifying location privacy. In *IEEE Symposium on Security and Privacy*. IEEE, May 2011.

[25] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. L. Boudec. Protecting location privacy: Optimal strategy against localization attacks. In *CCS*. ACM, 2012.

[26] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *CCS*, pages 348–357. ACM, November 2009.