This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

User-Defined Privacy Grid System for Continuous Location-Based Services

Roman Schlegel, *Member, IEEE,* Chi-Yin Chow, *Member, IEEE,* Qiong Huang, *Member, IEEE,* and Duncan S. Wong, *Member, IEEE*

Abstract—Location-based services (LBS) require users to continuously report their location to a potentially untrusted server to obtain services based on their location, which can expose them to privacy risks. Unfortunately, existing privacy-preserving techniques for LBS have several limitations, such as requiring a fully-trusted third party, offering limited privacy guarantees and incurring high communication overhead. In this paper, we propose a user-defined privacy grid system called *dynamic grid system* (DGS); the first holistic system that fulfills four essential requirements for privacy-preserving *snapshot* and *continuous* LBS. (1) The system only requires a semi-trusted third party, responsible for carrying out simple matching operations correctly. This semi-trusted third party does not have any information about a user's location. (2) Secure snapshot and continuous location privacy is guaranteed under our defined adversary models. (3) The communication cost for the user does not depend on the user's desired privacy level, it only depends on the number of relevant points of interest in the vicinity of the user. (4) Although we only focus on range and k-nearest-neighbor queries in this work, our system can be easily extended to support other spatial queries without changing the algorithms run by the semi-trusted third party and the database server, provided the required search area of a spatial query can be abstracted into spatial regions. Experimental results show that our DGS is more efficient than the state-of-the-art privacy-preserving technique for continuous LBS.

Index Terms—Dynamic grid systems, location privacy, location-based services, spatio-temporal query processing, cryptography

1 Introduction

In today's world of mobility and ever-present Internet connectivity, an increasing number of people use location-based services (LBS) to request information relevant to their current locations from a variety of service providers. This can be the search for nearby points of interest (POIs) (e.g., restaurants and hotels), location-aware advertising by companies, traffic information tailored to the highway and direction a user is traveling and so forth. The use of LBS, however, can reveal much more about a person to potentially untrustworthy service providers than many people would be willing to disclose. By tracking the requests of a person it is possible to build a movement profile which can reveal information about a user's work (office location), medical records (visit to specialist clinics), political views (attending political events), etc.

Nevertheless, LBS can be very valuable and as such users should be able to make use of them without having to give up their *location privacy*. A number of approaches have recently been proposed for preserving the user location privacy in LBS. In general, these approaches can be classified into two main categories. (1) *Fully-trusted third party* (TTP). The most popular privacy-preserving techniques require a TTP to be placed between the user and the service provider to hide the user's location information from the service provider (e.g., [1]–[8]). The main task of the third party is keeping track of the exact location of all users and blurring a querying user's location into a cloaked area that includes $k-1$ other users to achieve k-anonymity. This TTP model has three drawbacks. (a) All users have to continuously report their exact location to the third party, even though they do not subscribe to any LBS. (b) As the third party knows the exact location of every user, it becomes an attractive target for attackers. (c) The k-anonymity-based techniques only achieve low regional location privacy because cloaking a region to include k users in practice usually results in small cloaking areas. (2) *Private information retrieval (PIR) or oblivious transfer (OT).* Although PIR or OT techniques do not require a third party, they incur a much higher communication overhead between the user and the service provider, requiring the transmission of much more information than the user actually needs (e.g., [9]–[11]).

Only a few privacy-preserving techniques have been proposed for *continuous* LBS [2], [7]. These techniques rely on a TTP to continuously expand a cloaked area to include the initially assigned k users. These techniques not only inherit the drawbacks of the TTP model, but they also have other limitations. (1) *Inefficiency.* Continuously expanding cloaked areas substantially increases the query processing overhead. (2) *Privacy leakage.* Since the database server receives a set of consecutive cloaked areas of a user at different timestamps, the correlation among the cloaked areas would provide useful information for inferring the user's location. (3) *Service termination.* A user has to terminate the service when users initially assigned to her cloaked area leave the system.

In this paper, we propose a user-defined privacy grid system called *dynamic grid system* (DGS) to provide privacy-preserving *snapshot* and *continuous* LBS. The main idea is to place a *semi-trusted* third party, termed *query server* (QS), between the user and the service provider (SP). QS only needs to be semi-trusted because it will not collect/store or even have access to any user location information. *Semi-trusted* in this context means that while

• *R. Schlegel is with Corporate Research, ABB Switzerland Ltd., Baden-Dättwil, Switzerland. *C.-Y. Chow and *D. S. Wong are with the Department of Computer Science, City University of Hong Kong, Hong Kong. *Q. Huang is with the College of Informatics, South China Agricultural University, China. *Authors are ordered alphabetically. E-mail: rs@ione.ch, chiychow@cityu.edu.hk, csqhuang-c@my.cityu.edu.hk, and duncan@cityu.edu.hk*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

QS will try to determine the location of a user, it still correctly carries out the simple matching operations required in the protocol, i.e., it does not modify or drop messages or create new messages. An *untrusted* QS would arbitrarily modify and drop messages as well as inject fake messages, which is why our system depends on a *semi-trusted* QS.

The main idea of our DGS. In DGS, a querying user first determines a *query area*, where the user is comfortable to reveal the fact that she is somewhere within this query area. The query area is divided into equal-sized grid cells based on the dynamic grid structure specified by the user. Then, the user encrypts a query that includes the information of the query area and the dynamic grid structure, and encrypts the identity of each grid cell intersecting the required search area of the spatial query to produce a set of *encrypted identifiers*. Next, the user sends a request including (1) the encrypted query and (2) the encrypted identifiers to QS, which is a semi-trusted party located between the user and SP. QS stores the encrypted identifiers and forwards the encrypted query to SP specified by the user. SP decrypts the query and selects the POIs within the query area from its database. For each selected POI, SP encrypts its information, using the dynamic grid structure specified by the user to find a grid cell covering the POI, and encrypts the cell identity to produce the encrypted identifier for that POI. The encrypted POIs with their corresponding encrypted identifiers are returned to QS. QS stores the set of encrypted POIs and only returns to the user a subset of encrypted POIs whose corresponding identifiers match any one of the encrypted identifiers initially sent by the user. After the user receives the encrypted POIs, she decrypts them to get their exact locations and computes a query answer.

Because the user is continuously roaming she might need information about POIs located in other grid cells (within the query area) that have not been requested from QS before. The user therefore simply sends the encrypted identifiers of the required grid cells to QS. Since QS previously stored the POIs within the query area together with their encrypted identifiers, it does not need to enlist SP for help. QS simply returns the required POIs whose encrypted identifiers match any one of the newly required encrypted identifiers to the user. After the user received the encrypted POIs from QS, she can evaluate the query locally. When the user unregisters a query with QS, QS removes the stored encrypted POIs and their encrypted identifiers. In addition, when the required search area of a query intersects the space outside the current query area, the user unregisters the query with QS and re-issues a new query with a new query area.

Contributions. Our DGS has the following key features: (1) *No TTP.* Our DGS only requires a *semi-trusted* query server (QS) (i.e., trusted to correctly run the protocol) located between users and service providers. (2) *Secure location privacy.* DGS ensures that QS and other users are unable to infer any information about a querying user's location, and the service provider SP can only deduce that the user is somewhere within the user-specified query area, as long as QS and SP do not collude. (3) *Low communication overhead.* The communication cost of DGS for the user does not depend on the user-specified query area size. It only depends on the number of POIs in the grid cells overlapping with a query's required search area. (4) *Extensibility to various spatial queries.* DGS is applicable to various types of spatial queries without changing the algorithms carried out by QS or SP if their answers can be abstracted into spatial regions, e.g., reverse-NN queries [12] and density queries [13].

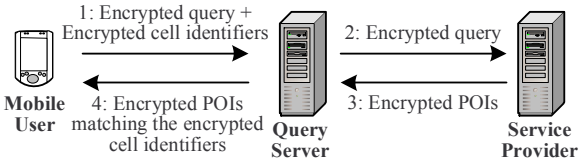


Fig. 1. System architecture of our DGS

The rest of this paper is organized as follows. Section 2 presents the system model of our DGS. Section 3 describes the query processing algorithms designed for DGS. Section 4 shows that DGS is secure and preserves user location privacy. Section 5 shows experimental results. Section 6 highlights related work. Finally, Section 7 concludes the paper.

2 System Architecture

Fig. 1 depicts the system architecture of our dynamic grid system (DGS) designed to provide privacy-preserving continuous LBS for mobile users. Our system consists of three main entities, *service providers*, *query servers* and *mobile users*. We will describe the main entities and their interactions, and then present the two spatial queries, i.e., range and k-nearest-neighbor (NN) queries, supported by our system.

Service providers (SP). Our system supports any number of independent service providers. Each SP is a spatial database management system that stores the location information of a particular type of *static* POIs, e.g., restaurants or hotels, or the store location information of a particular company, e.g., Starbucks or McDonald's. The spatial database uses an existing spatial index (e.g., R-tree or grid structure) to index POIs and answer range queries (i.e., retrieve the POIs located in a certain area). As depicted in Fig. 1, SP does not communicate with mobile users directly, but it provides services for them indirectly through the query server (QS).

Mobile users. Each mobile user is equipped with a GPS-enabled device that determines the user's location in the form (x_u, y_u). The user can obtain snapshot or continuous LBS from our system by issuing a spatial query to a particular SP through QS. Our system helps the user select a query area for the spatial query, such that the user is willing to reveal to SP the fact that the user is located in the given area. Then, a grid structure is created and is embedded inside an encrypted query that is forwarded to SP, it will not reveal any information about the query area to QS itself. In addition, the communication cost for the user in DGS does not depend on the query area size. This is one of the key features that distinguishes DGS from the existing techniques based on the fully-trusted third party model.

When specifying the query area for a query, the user will typically consider several factors. (1) The user specifies a minimum privacy level, e.g., city level. For a snapshot spatial query, the query area would be the minimum bounding rectangle of the city in which the user is located. If better privacy is required, the user can choose the state level as the minimum privacy level (or even larger, if desired). The size of the query area has no performance implications whatsoever on the user, and a user can freely choose the query area to suit her own privacy requirements. For continuous spatial queries, the user again first chooses a query area representing the minimum privacy level required, but also takes into account possible movement within the time period t for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

the query (e.g., 30 minutes). If movement at the maximum legal speed could lead the user outside of the minimum privacy level query area within the query time t, the user enlarges the query area correspondingly. This enlargement can be made generously, as a larger query area does not make the query more expensive for the user, neither in terms of communication nor computational cost. (2) The user can also generate a query area using a desired k-anonymity level as a guideline. Using a table with population densities for different areas, a user can look-up the population density of the current area, and use this to calculate the query area size such that the expected number of users within the query area correlates with the desired k-anonymity level. Considering that this is an approximation for the corresponding k-anonymity, the resulting query area can be taken as a lower-bound and the final query area size calculated as the lower-bound times a safety margin factor. The idea of using such density maps has been used for LBS [14] and health data [15]. (3) Alternatively, the user can specify a query area based on how far she wants to travel, e.g., if the user wants to find restaurants within the downtown area, she sets the downtown area as the query area.

A larger query area does have an impact on QS and SP in terms of workload and communication cost, but the bottleneck is considered to be between the user and QS, and the load on this link and on the user does not depend on the query area size. A system parameter can be defined to limit the maximum query area size or number of objects returned to a user, in order to not overload the client-side application.

Query servers (QS). QS is a semi-trusted party placed between the mobile user and SP. Similar to the most popular infrastructure in existing privacy-preserving techniques for LBS, QS can be maintained by a telecom operator [16]. The control/data flows of our DGS are as follows (Fig. 1):

1) The mobile user sends a request that includes (a) the identity of a user-specified SP, (b) an *encrypted query* (which includes information about the user-defined dynamic grid structure), and (c) a set of *encrypted identifiers* (which are calculated based on the user-defined dynamic grid structure) to QS.
2) QS stores the encrypted identifiers and forwards the encrypted query to the user-specified SP.
3) SP decrypts the query and finds a proper set of POIs from its database. It then encrypts the POIs and their corresponding identifiers based on the dynamic grid structure specified by the user and sends them to QS.
4) QS returns to the user every encrypted POI whose encrypted identifier matches one of the encrypted identifiers initially sent by the user. The user decrypts the received POIs to construct a candidate answer set, and then performs a simple filtering process to prune false positives to compute an exact query answer.

We assume that there is a secure channel between the user and QS. This assumption is necessary as SP might be able to learn the user's location if it can eavesdrop on the communication between the user and QS. The secure channel can easily be established using standard techniques such as identity-based encryption [17], key exchange protocols (e.g. MQV [18]) or SSL/WTLS. We also note that the data privacy of SP is protected with regards to QS, as QS only receives encrypted information about the POIs, and only the client can decrypt the POIs. QS hence does not learn any information about the POIs (e.g., name, address, ratings, etc.).

We additionally note that the work-load of the QS depends to some extent on the query area chosen by a user, i.e., a large query area would lead to the SP sending a comparatively larger amount of data to the QS. If this is an issue, a QS could provide different tiers of services, some of which could be paid, that would allow for different privacy requirements of users (i.e., different maximum query area sizes). Basic economical principles could be applied to make this worthwhile both for users and QS service providers.

Supported spatial queries. DGS supports the two most popular spatial queries, i.e., range and k-NN queries, while preserving the user's location privacy. The mobile user registers a continuous range query with our system to keep track of the POIs within a user-specified distance, *Range*, of the user's current location (x_u, y_u) for a certain time period, e.g., "*Continuously send me the restaurants within one mile of my current location for the next one hour*". The mobile user can also issue a continuous k-NN query to find the k-nearest POIs to the user's current location (x_u, y_u) for a specific time period, e.g., "*Continuously send me the five nearest restaurants to my current location for the next 30 minutes*".

Since a snapshot query is just the initial answer of the continuous one, DGS also supports snapshot range and k-NN queries. Although we only focus on range and k-NN queries in this work, DGS is applicable to other continuous spatial queries if the query answer can be abstracted into spatial regions. For example, our system can be extended to support reverse-NN queries [12] and density queries [13] because recent research efforts have shown that the answer of these queries can be maintained by monitoring a region.

2.1 Adversarial Models

We now discuss adversarial models regarding QS and SP, and then present the formal security proof of our DGS in Section 4. A malicious QS or SP will try to break a user's privacy by working with the data available to them within the described protocol. We do not consider QS or SP with access to external information not directly related to the protocol.

2.1.1 User Anonymity

As described above, both QS and SP will try to de-anonymize a user by using the information contained in the protocol (although they still faithfully follow the protocol itself). While QS does not have any information about a user that would allow it to narrow down the list of users that would fit a specific query, SP has access to the plaintext query of a user. This query, however, only contains the query region and the grid parameters, and with the information available, QS can therefore do no better than establish that the user is somewhere within the query region (see also Section 4).

One other concern regarding the de-anonymization of users is that if for example the services of SP are paid services, then SP might for example be able to link a query with a billing record and at least establish the presence of a user in a query area. While in this paper we consider it acceptable that a user can be located to be within a query region by QS (after all, the user can freely choose the query area and hence choose it such that her personal privacy requirements are met), there is other research which would allow to prevent the linking of a query area to a specific user through billing records, for example the work by Yau and An [19]. So even if the SP requires the authentication of users to provide a (paid) service, the service can be provided while protecting the anonymity of the user. However, no matter in which way the SP

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

provides the service, the privacy guarantees will always be better than TTP, as a TTP always knows the exact location of the users, while in our system neither QS nor SP know the exact location of a user. Regarding paid services and QS, in such a case QS does not have any information to narrow down the geographic location of a user, even if it is being used as a paid service and can link queries to billing records.

Regarding the de-anonymization of users, we also note that the type of POI in a query sent to SP or the density of POIs per cell in the query area, do not provide QS with any meaningful information that could be used to reduce the anonymity set of a user. Specifically, there is no correlation between the density of POIs in a cell and the actual location of a user, as the user launches a query without a-priori knowledge of the density of POIs, and hence the density of POIs in a cell cannot be used to make deductions about the possible location of a user in the query area.

A natural choice for the role of QS is the network service provider of a user. Even though the network service provider can typically locate a user down to an individual cellular network cell already, taking on the role of QS does not provide it with any additional information about the user, such as the actual query area. There is also no requirement for the queries of a user to correspond to the user's actual location, and the network service provider does not have any information available that would allow it to infer either case. To summarize, a network service provider already knows the location of its users, and serving as a QS does not provide it with any additional information about its users. Alternatively, QS services could also be provided by volunteers (e.g., like many of the nodes in the Tor network), by ad-supported services, or even by services that charge a modest fee.

2.1.2 Other Attacks

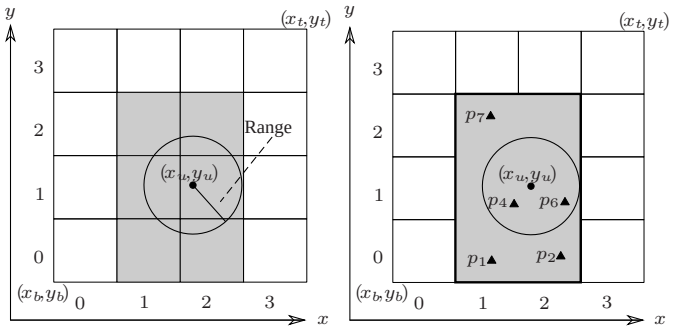
In this subsection we discuss a few other attacks and explain how they relate to our proposed system.

IP localization. One possible attack involves QS trying to determine the position of a user through IP localization (i.e., using a database which can map IP addresses to locations). Because of how mobile phone networks are setup (considering that our system is aimed at mobile users using mobile phone networks), however, mobile phones cannot be located with useful accuracy, as shown by Balakrishnan et al. [20]. Even so, if IP localization is a concern, solutions at the network level can hide the originating IP, for example by using an anonymizing software such as Tor [21].

Timing attacks. Another set of attacks might use timing information if QS can observe the traffic close to the originating user. However, if QS can observe such traffic, the location privacy of the user is very likely already compromised, even without timing attacks. Furthermore, we consider this to be out of the scope of our work.

Query server as client. QS might try to also act as a client in an attempt to gain some information which could help to localize a user. QS has no information to launch such an attack, however, not even an approximate location of the user. Also, the number of POIs returned to a client does not allow QS to make any inferences, because it knows neither the query area nor the grid parameters. A large number of POIs could either mean a dense region or a large query area, depending on the grid parameters, which are unknown to QS.

Network traffic fingerprinting. An attack as described by Bissias et al. [22] which makes inferences based on the statistics of encrypted network connections is not applicable to our system. The attack as described in the paper is equivalent to determining which QS a user is using. This information does not need to be secret and communication with QS is highly uniform across different query servers (unlike website traffic), very likely making them for all practical purposes indistinguishable.



(a) Dynamic grid structure (b) Answer computation

Fig. 2. Example of range query processing in DGS

Commuter problem. Another attack that can identify the home and the office location of a user through location traces is described by Golle et al. [23]. This attack is not applicable to our system, because no plaintext locations are ever transmitted in our system and no inferences can be made.

We exclude side-channel attacks in general from the security analysis as being out of the scope of this paper. Many of the side-channel attacks mentioned above are fundamental to network communications, and as such neither limited to nor a consequence of the design of our proposed protocol.

3 Dynamic Grid System (DGS)

In this section, we will describe how our DGS supports privacy-preserving continuous range and k-NN queries. This section is organized as follows: Section 3.1 describes the details of our DGS for processing continuous range queries and incrementally maintaining their answers, and Section 3.2 extends DGS to support k-NN queries.

3.1 Range Queries

Our DGS has two main phases for privacy-preserving continuous range query processing. The first phase finds an initial (or a snapshot) answer for a range query (Section 3.1.1), and the second phase incrementally maintains the query answer based on the user's location updates (Section 3.1.2).

3.1.1 Range Query Processing

As described in Section 2, a continuous range query is defined as keeping track of the POIs within a user-specified distance $Range$ of the user's current location (x_u, y_u) for a certain time period. In general, the privacy-preserving range query processing protocol has six main steps.

Step 1. Dynamic grid structure (by the user). The idea of this step is to construct a dynamic grid structure specified by the user. A querying user first specifies a query area, where the user is comfortable to reveal the fact that she is located somewhere within that query area. The query area is assumed to be a rectangular area, represented by the coordinates of its bottom-left vertex (x_b, y_b) and top-right vertex (x_t, y_t). Notice that the user is not necessarily

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

required to be at the center of the query area. Instead, its location can be anywhere in the area. However, our system can also support irregular spatial regions, e.g., the boundary of a city or a county, by using a minimum bounding rectangle to model the irregular spatial region as a rectangular area. The query area is divided into $m \times m$ equal-sized grid cells to construct a dynamic grid structure, where m is a user-specified parameter. Each grid cell is identified by (c, r), where c is the column index from left to right and r is the row index from bottom to top, respectively, with $0 \leq c, r < m$. Given the coordinates of the bottom-left vertex of a grid cell, (x_c, y_c), the grid cell identity can be computed by $(c, r) = \left(\left\lfloor \frac{x_c - x_b}{(x_t - x_b)/m} \right\rfloor, \left\lfloor \frac{y_c - y_b}{(y_t - y_b)/m} \right\rfloor\right)$. Fig. 2 gives a running example for privacy-preserving range query processing, where the querying user is located in the cell $(2, 1)$, $m = 4$, and the circle with a radius of the range distance $Range$ specified by the user constitutes the query region of the range query.

Step 2. Request generation (by the user). In this step, the querying user generates a request that includes (1) a query for a SP specified by the querying user and (2) a set of encrypted identifiers, S_e, for a QS. The user first selects a random key K and derives three distinct keys:

$$(\mathrm{HK}, \mathrm{EK}, \mathrm{MK}) \leftarrow \mathrm{KDF}(K) \tag{1}$$

where $\mathrm{KDF}(\cdot)$ is a key derivation function ([24]). Then, the user sets query and S_e as follows:

(1) *Query generation.* An encrypted query for a specific SP is prepared as:

$$\mathsf{query} \leftarrow \mathsf{IBE.Enc}_{SP}(\text{POI-type}, K, m, (x_b, y_b), (x_t, y_t)) \tag{2}$$

where $\mathsf{IBE.Enc}_{SP}(\cdot)$ is Identity-Based Encryption (IBE) under the identity of SP (details of IBE are in [17]). In the encrypted query, POI-type specifies the type of POIs, K is the random key selected by the user, and the personalized dynamic grid structure is specified by m, (x_b, y_b), and (x_t, y_t).

(2) *Encrypted identifier generation.* Given the query region of the range query, the user selects a set of grid cells S_c in the dynamic grid structure that intersect the query region, i.e., a circle centered at the user's current location (x_u, y_u) with a radius of $Range$. For each selected grid cell i in S_c, its identity (c_i, r_i) is encrypted to generate an encrypted identifier:

$$h_i \leftarrow H(c_i, r_i) \tag{3}$$

$$C_i \leftarrow \mathsf{SE.Enc}_{\mathrm{HK}}(h_i) \tag{4}$$

where $H(\cdot)$ is a collision-resistant hash function and $\mathsf{SE.Enc}_{key}(\cdot)$ a symmetric encryption algorithm (for example AES-based) under key key. After encrypting all the grid cells in S_c, the user generates a set of encrypted identifiers S_e. It is important to note that the user will make sure that the identifiers in S_e are ordered randomly. Finally, the user produces a request as below and sends it to QS:

$$\mathsf{request} \leftarrow \langle SP, \mathsf{query}, S_e \rangle \tag{5}$$

In the running example (Fig. 2a), the range query region, which is represented by a circle, intersects six grid cells, i.e., $(1, 0)$, $(2, 0)$, $(1, 1)$, $(2, 1)$, $(1, 2)$, and $(2, 2)$ (represented by shaded cells), which make up the set of grid cells S_c, and thus, the user has to encrypt each identity of these grid cells and produce six encrypted identifiers in S_e.

Step 3. Request processing (by QS). When QS receives the request from the user, it simply stores the set of encrypted identifiers S_e and forwards the encrypted query to SP specified by the user.

Step 4. Query processing (by SP). SP decrypts the request to retrieve the POI-type, the random key K selected by the user in the request generation step (Step 2), and the query area defined by m, (x_b, y_b), and (x_t, y_t). SP then selects a set of n_p POIs that match the required POI-type within the user specified query area from its database. For each selected POI j with a location (x_j, y_j)) $(1 \leq j \leq n_p)$, SP computes the identity of the grid cell in the user specified dynamic grid structure covering j by $(c_j, r_j) = \left(\left\lfloor \frac{x_j - x_b}{(x_t - x_b)/m} \right\rfloor, \left\lfloor \frac{y_j - y_b}{(y_t - y_b)/m} \right\rfloor\right)$.

Then, SP generates $(\mathrm{HK}, \mathrm{EK}, \mathrm{MK}) \leftarrow \mathrm{KDF}(K)$ and computes the following:

$$h_j \leftarrow H(c_j, r_j) \tag{6}$$

$$C_j \leftarrow \mathsf{SE.Enc}_{\mathrm{HK}}(h_j) \tag{7}$$

$$l_j \leftarrow \mathsf{SE.Enc}_{\mathrm{EK}}(x_j, y_j) \tag{8}$$

$$\sigma_j = MAC_{\mathrm{MK}}(C_j, l_j) \tag{9}$$

where $MAC_{key}(\cdot)$ is a message authentication code under key key. The C_j is the corresponding encrypted identifier of a POI, the l_j contains the exact location of a POI in encrypted form, and the σ_j is included to prevent certain attacks by QS (such as tampering with the encrypted POIs).

Finally, SP sends the set of selected POIs back to QS in the following form:

$$\langle \mathrm{POI}_j = (C_j, l_j, \sigma_j) \rangle, \qquad \text{where } j = 1, \ldots, n_p. \tag{10}$$

Step 5. Encrypted identifier matching (by QS). Upon receiving n_p triples, QS determines the set of matching POIs by comparing the encrypted identifiers C_j $(1 \leq j \leq n_p)$ of the received POI with the set of encrypted identifiers S_e previously received from the user. A match between a C_j and some C_i in the set S_e indicates that the POI j is in one of the grid cells required by the user. Thus, QS forwards every matching POI $\langle l_j, \sigma_j \rangle$ to the user. If the query is a snapshot query, QS then deletes the received POIs and their encrypted identifiers. However, if the query is a continuous one, QS keeps the received POIs along with their encrypted identifiers until the user unregisters the query.

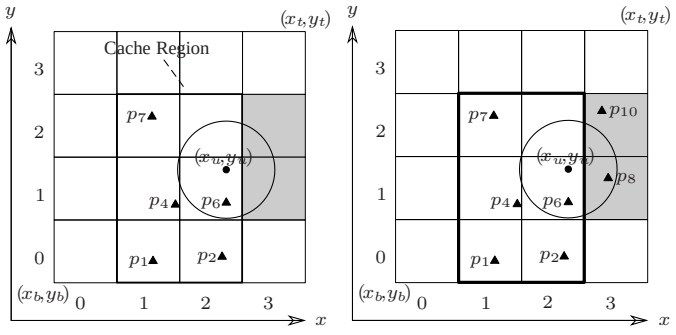
Step 6. Answer computation (by the user). Suppose that there are μ matched POIs received by the user. For each of these matched POIs, say $\langle l_j, \sigma_j \rangle$, the user decrypts l_j using EK and gets access to the exact location (x_j, y_j) of the POI. From (x_j, y_j) and l_j, the user verifies σ_j by re-calculating the MAC value and compares it against σ_j. If they match, the user finds the answer that includes the POI whose location is within a distance of $Range$ of the user's current position (x_u, y_u). In the running example (Fig. 2b), the user receives five POIs from QS, where the range query answer includes two POIs, i.e., p_4 and p_6.

3.1.2 Incremental Range Query Answer Maintenance

After the user gets the initial response for a range query from QS, she can find the initial (or snapshot) query answer locally. Then, incremental answer updates can be performed to maintain the answer when the user's location changes. This phase has four main steps.

Step 1. Cache region (by the user). The user keeps track of the grid cells previously requested and caches the POIs returned by QS for those cells. These cells define a *cache region*. The user is able to find a query answer from the cached POIs as long as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing



(a) Cache region (b) Answer refinement

Fig. 3. Example of incremental range query maintenance

the query region (i.e., a circular area centered at the user's current location (x_u, y_u) with a radius of range distance $Range$ specified by the user) is contained within the cache region. However, when the query region intersects some grid cells (within the query area) outside the cache region, the user executes the following steps (Steps 2 to 4) to enlist QS for help to find a query answer.

Fig. 3 depicts a running example for the incremental range query answer maintenance phase, where the user has requested the POIs within six grid cells, as illustrated in Fig. 2. The combined area of these six grid cells constitutes the cache region, represented by a bold rectangle. As shown in Fig. 3a, the query region of the range query intersects two grid cells outside the cache region, i.e., $(3, 1)$ and $(3, 2)$ (represented by shaded cells), so the user has to execute Steps 2 to 4 to get the POIs within these two grid cells from QS.

Step 2. Incremental request generation (by the user). This step is similar to the request generation step (Step 2) in the range query processing phase (Section 3.1.1), except that the user only generates encrypted identifiers for the set of grid cells S_c that intersect the query region but are outside of the cache region, i.e., that have not been requested by the user before. For each grid cell in S_c, its identity is encrypted to generate an encrypted identifier C_i using Equations 3 and 4. Then, an answer update request including the set of encrypted identifiers S_e is sent to QS. In the running example (Fig. 3a), the user has to get the POIs within the two grid cells $(3, 1)$ and $(3, 2)$ to evaluate the range query, so she encrypts their cell identities to generate two encrypted identifiers in S_e, and sends an update request along with S_e to QS.

There is the possibility that a small amount of information about the movement of a user is leaked if the user only requests the cells outside the cache region. For example, if a user requests six cells in an initial request, and then two cells in an answer update request a short while later, an adversary (e.g., QS) might be able to infer information about the movements of the user. If this is a concern, a user can pad the number of encrypted identifiers in an answer update request to obtain the same number of encrypted identifiers as in the initial request by generating encrypted identifiers for additional grid cells, which have not been requested by the user before and are around the grid cells in S_c. In this way, the adversary cannot distinguish between the "actual" encrypted identifiers and the "additional" encrypted identifiers.

Step 3. Request processing (by QS). Because QS already cached the POIs along with their encrypted identifiers within the user-specified query area from the initial request, it does not need to contact SP to deal with the answer update request. Instead, similar to the encrypted identifier matching step (Step 5) in the range query processing phase, QS simply returns the POIs whose encrypted identifier matches one of the encrypted identifiers in S_e in the answer update request sent by the user.

Step 4. Answer computation (by the user). This step is similar to the answer computation step (Step 6) in the range query processing phase, except that the user evaluates the query based on both the POIs newly returned by QS and the previously cached POIs. Fig. 3b depicts that the user gets two new POIs, p_8 and p_{10}, from QS, and the new query answer includes POIs p_6 and p_8.

3.2 K-Nearest-Neighbor Queries

Similar to continuous range queries, the privacy-preserving query processing for continuous k-NN queries has two main phases. The first phase finds an initial (or snapshot) answer (Section 3.2.1), while the second phase maintains the correct answer when the user moves by using incremental updates (Section 3.2.2). However, unlike range queries, the required search area of a k-NN query is unknown to a user until the user finds at least k POIs to compute a required search area, i.e., a circular area centered at the user's location with a radius from the user to the k-th nearest POI. Thus, the privacy-preserving query processing protocol of k-NN queries is slightly different.

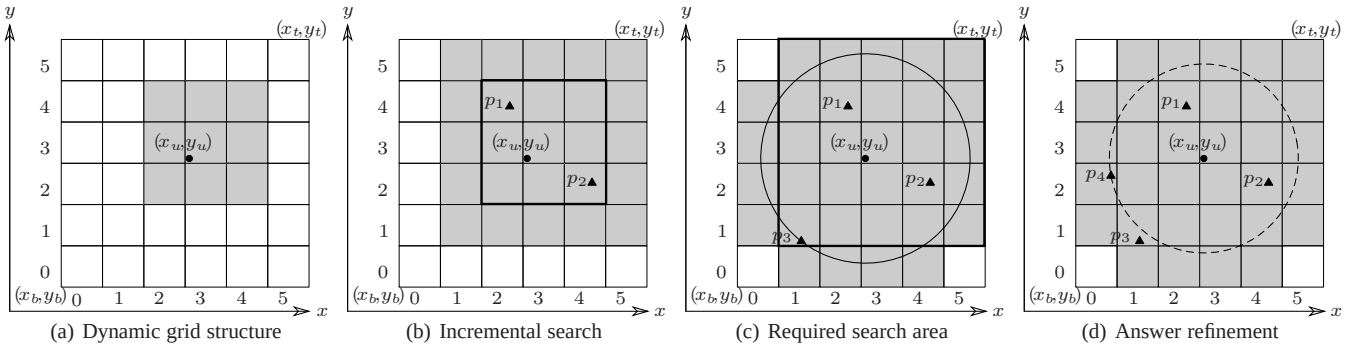
3.2.1 K-Nearest-Neighbor Query Processing

A continuous k-NN query is defined as keeping track of the k-nearest POIs to a user's current location (x_u, y_u) for a certain time period, as presented in Section 2. In general, the privacy-preserving k-NN query processing has six major steps to find an initial (or snapshot) query answer. Fig. 4 depicts a running example of the privacy-preserving query processing of a k-NN query, where $k = 3$.

Step 1. Dynamic grid structure (by the user). This step is the same as the dynamic grid structure step (Step 1) in the range query processing phase (Section 3.1.1). It takes a user-specified query area with a left-bottom vertex (x_b, y_b) and a right-top vertex (x_t, y_t) and divides the query area into $m \times m$ equal-sized cells, as illustrated in Fig. 4a ($m = 6$).

Step 2. Request generation (by the user). The required search area of the k-NN query is initially unknown to the user. The user first finds at least k POIs to compute the required search area as a circular area centered at the user's location with a radius of a distance from the user to the k-th nearest known POI. The user therefore first attempts to get the nearby POIs from a specific SP. In this step, the user requests the POIs in the cell containing the user and its neighboring cells from SP. Given the user's current location (x_u, y_u) and a query area specified by the user in Step 1, she wants to get the POIs within a set of grid cells S_c that includes the cell containing herself, i.e., $(c_u, r_u) = \left(\left\lfloor \frac{x_u - x_b}{(x_t - x_b)/m} \right\rfloor, \left\lfloor \frac{y_u - y_b}{(y_t - y_b)/m} \right\rfloor\right)$, and its at most eight neighboring cells $(c_u - 1, r_u - 1)$, $(c_u, r_u - 1)$, $(c_u + 1, r_u - 1)$, $(c_u - 1, r_u)$, $(c_u + 1, r_u)$, $(c_u - 1, r_u + 1)$, $(c_u, r_u + 1)$, and $(c_u + 1, r_u + 1)$. For each cell i in S_c, the user generates an encrypted identifier C_i using Equations 3 and 4, as in the request generation step (Step 2) in the range query processing phase. The user also creates a query to be sent to SP based on Equation 2. Finally, the user sends a request, which includes the identity of SP, the query, and the set of encrypted identifiers (in random order) S_e, as given in Equation 5, to QS.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing



(a) Dynamic grid structure (b) Incremental search (c) Required search area (d) Answer refinement

Fig. 4. Example of k-nearest-neighbor query processing in DGS

In the running example (Fig. 4a), the user located in the grid cell $(3,3)$ and therefore requests the POIs in the cells $(3,3)$ and its neighboring grid cells, i.e., $(2,2)$, $(3,2)$, $(4,2)$, $(2,3)$, $(4,3)$, $(2,4)$, $(3,4)$, and $(4,4)$, (represented by shaded cells) from SP through QS.

Step 3. Request processing (by QS). This step is identical to Step 3 for range queries in the query processing phase (Section 3.1.1).

Step 4. Query processing (by SP). This step is identical to Step 4 for range queries in the query processing phase (Section 3.1.1). Thanks to this query abstraction feature, our DGS can be easily extended to support other continuous spatial query types, e.g., reverse NN queries and density queries.

Step 5. Required search area (by the user and QS). This step is similar to the encrypted identifier matching step (Step 5) for range queries in the query processing phase (Section 3.1.1), with the difference that this step may involve several rounds of interaction between the user and QS. QS matches the encrypted identifiers of the encrypted POIs returned by SP with the encrypted identifiers in S_e sent by the user in Step 2, and sends the matching encrypted POIs to the user.

If at least k encrypted POIs are returned to the user, she can decrypt them to compute a required search area for the k-NN query in the form of a circle centered at the user's location with a radius of the distance between the user and the k-th nearest POI. On the other hand, if less than k POIs are returned, the user starts the next iteration by requesting the grid cells from QS one hop further away from the position of the user, i.e., the neighboring cells of the grid cells that have already been requested by the user. This incremental search process is repeated (i.e., requesting more cells moving steadily outward from the user's position) until the user has obtained at least k POIs from QS. After the user determines the required search area, there are two possibilities:

1) The user has already requested all the cells which intersect the required search area. In this case, the user proceeds to the next step.
2) The required search area intersects some cells which have not yet been requested from QS. The (at least) k POIs found so far may in that case not be an exact answer, and the user requests those cells from QS which intersect the required search area but have not been requested yet (in Fig. 4c these would be the shaded cells outside the bold rectangle). After receiving all encrypted POIs in the newly requested cells from QS, the user proceeds to the next step.

In the running example, Fig. 4b depicts that the grid cells initially requested by the user (within the bold rectangle) contains less than three POIs. The user therefore requests the neighboring grid cells (the grid cells adjacent to the bold rectangle) from QS. This will result in discovering three POIs in total, i.e., p_1, p_2, and p_3. The user then computes the required search area represented by a circle (Fig. 4c). As the required search area intersects eight cells which the user has not yet requested from QS (i.e., they are outside the bold rectangle), the user will launch another request for the grid cells $(0,1)$, $(0,2)$, $(0,3)$, $(0,4)$, $(1,0)$, $(2,0)$, $(3,0)$, and $(4,0)$.

Step 6. Answer refinement (by the user). Having received all the POIs within all the grid cells intersecting the required search area, the user can decrypt them to get their exact locations, as in the answer computation step (Step 6) for range queries in the query processing phase (Section 3.1.1), and determine the exact answer by selecting the k nearest POIs. The previous steps ensure that these k POIs are indeed the closest ones. In the running example (Fig. 4d), the user can find the exact answer for the 3-NN query, which includes three POIs p_1, p_2, and p_4.

3.2.2 *Incremental k-NN Query Answer Maintenance*

After the user computes an initial (or snapshot) k-NN query answer, the incremental answer update phase allows to maintain the answer as the user moves around. Similar to range queries, the incremental answer maintenance phase has four steps. The first two steps are the same as the cache region step and the incremental request generation step as in Section 3.1.2. In the third step (i.e., request processing) performed by the QS, since QS has already cached the encrypted POIs, together with their corresponding encrypted identifiers calculated by SP in Step 4 of the query processing phase, it does not need to contact SP. It can simply forward the encrypted POIs matching one of the encrypted identifiers in S_e to the user. In the last step (i.e., answer refinement) performed by the user, she decrypts the received POIs and sorts the ones located within the required search area according to their distance to the user in ascending order. The k-nearest POIs to the user constitute the new query answer.

4 Security Analysis

In this section, we define several security models which formalize the location privacy of our DGS, and show that the proposed schemes in Section 3 are secure. In our schemes, the query server (QS) sees a user's encrypted queries and POIs from a service provider (SP). Since the user query and returned POI locations are encrypted, QS could only learn the user's location from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

the number of returned POIs rather than the encrypted values. However, this is not the case in our scheme. The number of POIs returned by SP depends on the query area size, which is encrypted and unknown, and therefore, QS is not able to derive any useful information from the number of POIs, e.g., whether the user is in a dense or sparse region. See Lemma 2 for the detailed analysis.

After decrypting the query forwarded by QS, a SP obtains the query area which contains the user. Other than this, it learns nothing, since the user could be anywhere in the area. See Lemma 1 for the proof. Regarding the integrity, every encrypted POI is authenticated by SP using a MAC and the authentication key is only shared between the user and SP. Guaranteed by the security of the MAC, QS is unable to modify the information of any POI returned to the user, nor to add a "fake" POI. See Lemma 3 for the proof. As shown in Fig. 1 (Section 2) there are four message flows in a basic query in our DGS. We denote them by Msg_1 (from the user to QS), Msg_2 (from QS to SP), Msg_3 (from SP to QS) and Msg_4 (from QS to the user), respectively. In the following subsections we analyze the security of our scheme in detail.

4.1 Privacy Against Service Provider (SP)

We require that SP cannot learn the user's location any better than making a random guess. Formally, we consider the following game played between a challenger C and a (malicious) SP, denoted by $\mathcal{A}$.

The challenger prepares the system parameters, and gives them to $\mathcal{A}$. $\mathcal{A}$ specifies a POI-type, the grid structure, a query area and two locations (x_0, y_0) and (x_1, y_1) in this area, and gives them to C. C chooses at random $b \in \{0, 1\}$, uses (x_b, y_b), the specified grid structure and POI-type to generate Msg_2^b with respect to the identity of $\mathcal{A}$, i.e., the message that the malicious SP expects to receive. C then gives Msg_2^b to $\mathcal{A}$. $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Definition 1. *A DGS achieves **Privacy Against Service Provider (SP)** if for any SP, its success probability of winning the game above is at most $1/2 + \mathrm{negl}(\ell)$, where $\mathrm{negl}(\cdot)$ is a negligible function[1] in the security parameter.*

Lemma 1. *Our DGS achieves Privacy Against SP.*

Proof. The message that SP receives from QS is $\mathsf{Msg}_2 = \mathsf{query}$, which is an IBE of POI-type, K, the query area and grid structure under SP's identity (see Equation 2), and is independent of the user's location. Hence a (malicious) SP does not gain any advantage in guessing b chosen by the challenger. □

4.2 Privacy Against Query Server (QS)

This requires that QS cannot tell from the user's request or SP's transcript about where the user is, provided that it does not collude with the intended SP. Formally, we consider the following game played between an adversary $\mathcal{A}$ (which is the dishonest QS) and a challenger C which acts the roles of the user and service providers.

Given the system parameters, $\mathcal{A}$ begins to issue **Private Key Query** for polynomially many times: it submits the identity of a SP to C, and receives the corresponding private key. This models the case that QS colludes with a (non-intended) SP. $\mathcal{A}$ then specifies the POI-type, the identity of the intended SP (the private key of which has not been queried), the grid structure, a query area, and two user locations (x_0, y_0) and (x_1, y_1) in the query area, and gives them to C. C tosses a coin $b \in \{0, 1\}$, and uses (x_b, y_b) and the other information specified by $\mathcal{A}$ to generate Msg_1^b as the user's message to QS, and the corresponding SP message Msg_3^b. It sends both Msg_1^b and Msg_3^b to $\mathcal{A}$. $\mathcal{A}$ continues to issue queries as above except that it cannot ask for the private key of the intended SP. Finally, $\mathcal{A}$ outputs a bit b' as its guess of b, and wins the game if $b' = b$.

Definition 2. *A DGS achieves **Privacy Against Query Server (QS)** if for all probabilistic polynomial-time $\mathcal{A}$, its probability of winning the above game is negligibly close to $1/2$.*

Lemma 2. *Our DGS achieves Privacy Against QS.*

Proof. We prove the lemma by a series of games, $\mathbf{G}_0, \cdots, \mathbf{G}_i$, and denote by X_i the event that $\mathcal{A}$ wins game $\mathbf{G}_i$. $\mathbf{G}_0$: This is the original game defined in Definition 2. Given the system parameters, $\mathcal{A}$ begins to issue private key queries. At some point, it chooses a POI-type, the identity of the intended SP, the grid structure, a query area and two user locations (x_0, y_0), (x_1, y_1) in the query area, and sends them to the challenger, which selects (x_b, y_b) for some random bit b to generate the user message Msg_1^b and the corresponding SP message Msg_3^b, and returns them to $\mathcal{A}$. The adversary continues to issue private key queries as before except to ask for the private key of the intended SP. Finally, $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$. Notice that there exist two ranges, e.g., R_0 and R_1, such that the grids intersected with the circles with ranges R_0 and R_1, and centered at (x_0, y_0) and (x_1, y_1) respectively, contain the same number of POIs. Besides, the ranges are chosen by the users, and thus are unknown to the adversary.

$\mathbf{G}_1$: We change the generation of $(\mathrm{HK}, \mathrm{EK}, \mathrm{MK})$. Now we randomly select them from the space $\mathcal{K}_H \times \mathcal{K}_E \times \mathcal{K}_M$. Guaranteed by the security of KDF, we have that $|\Pr[X_1] - \Pr[X_0]|$ is negligible [25], [26].

$\mathbf{G}_2$: We change the generation of l_i in Msg_1^b. Instead of the encryption of the real location, now l_i is the encryption of a random location (other than (x_0, y_0) and (x_1, y_1)) under the key EK. Guaranteed by the semantic security of SE, we have that $|\Pr[X_2] - \Pr[X_1]|$ is negligible [25], [26].

$\mathbf{G}_3$: Similar to $\mathbf{G}_2$, we change all the C_i's in Msg_1^b and all the C_i's in Msg_3^b to be the encryptions of randomly selected h_i's, under the constraint that all POIs in the same grid cell share the same h_i with the grid cell itself. The semantic security of SE implies that $|\Pr[X_3] - \Pr[X_2]|$ is negligible.

$\mathbf{G}_4$: We change request in the client message to the encryption of a random tuple with the same length. The security of IBE implies that $|\Pr[X_4] - \Pr[X_3]|$ is negligible [17]. In $\mathbf{G}_4$, both Msg_1^b and Msg_3^b are independent of b chosen by the challenger. Therefore, $\mathcal{A}$ could succeed in outputting the correct bit with probability at most $1/2$, and we have $|\Pr[X_0] - \frac{1}{2}| \leq \sum_{i=0}^{3} |\Pr[X_{i+1}] - \Pr[X_i]| + |\Pr[X_4] - \frac{1}{2}|$ which is negligible. □

4.3 Integrity

This is to ensure that QS cannot modify any messages returned by SP or add any messages without being detected. Formally, we consider the following game, where the adversary $\mathcal{A}$ is a QS, and the challenger C plays the roles of the client and all the service providers.

1. A function $f : \mathbb{N} \to [0, 1]$ is negligible if for all positive polynomial $\mathrm{poly}(\cdot)$ there exists an N such that for all $n > N$, $f(n) < 1/\mathrm{poly}(n)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

Given the system parameters, $\mathcal{A}$ adaptively issues the following queries for polynomially number of times: **Private Key Query**: $\mathcal{A}$ submits the identity of a SP, and is returned the corresponding private key. **Service Query**: $\mathcal{A}$ generates a user message Msg_1 and sends it to SP indicated in Msg_1. It then receives the answer Msg_3. $\mathcal{A}$ then submits the POI-type, the identity of the intended SP (the private key of which has not been queried), the grid structure, a query area, and a user's location (x, y). C generates the user's message Msg_1^*, and the intended SP's message Msg_3^*, and sends them to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs Msg_4^*. Let Msg_4 be the correct answer that an honest QS would return to the user. $\mathcal{A}$ wins the game if $\mathsf{Msg}_4^* \not\subseteq \mathsf{Msg}_4$ but the user accepts Msg_4^*.

Definition 3. *A DGS achieves **integrity** if there is no probabilistic polynomial-time adversary $\mathcal{A}$ which wins the above game with non-negligible probability.*

Lemma 3. *Our DGS achieves integrity.*

Proof. Again, we use X_i to denote the event that $\mathcal{A}$ wins the game $\mathbf{G}_i$. $\mathbf{G}_0$ is the original game described above.

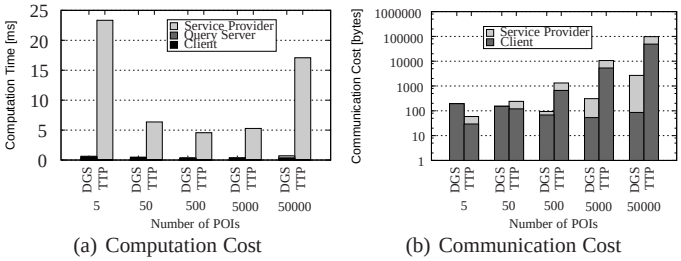
$\mathbf{G}_1$: We change the generation of Msg_1^* and Msg_3^*. Instead of being output by KDF on input a random K, MK is now selected at random from $\mathcal{K}_M$. It is also used in the verification of the adversary's final output, i.e., checking the validity of the MACs in Msg_4^*. The security of KDF implies that $|\Pr[X_1] - \Pr[X_0]|$ is negligible. Below we show that the adversary wins this game only with negligible probability.

We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ which breaks the strong unforgeability of MAC [25], [26]. Given oracle access to O_M, $\mathcal{B}$ generates system parameters which include the IBE key. It then invokes $\mathcal{A}$ on input the system parameters, and begins to answer $\mathcal{A}$'s queries as following. **Private Key Query**: Given the identity of a SP, $\mathcal{B}$ uses the IBE key to generate the private key of SP, and returns it to $\mathcal{A}$. **Service Query**: Given a client message $\mathsf{Msg}_1 = \langle SP, request, S_e \rangle$, $\mathcal{B}$ uses the private key of SP (which can be computed from the IBE key) to decrypt request and obtains POI-type, K, m, (x_b, y_b) and (x_t, y_t). It then follows the prescribed strategy of a SP to generate Msg_3, and returns it to $\mathcal{A}$.

When Msg_3 is ready, $\mathcal{A}$ outputs the identity of an intended SP, the private key of which has not been queried. $\mathcal{A}$ also outputs POI-type, the grid structure, a query region and a user's location (x, y). $\mathcal{B}$ then generates Msg_1^* using the information given by $\mathcal{A}$ and also Msg_3^* as follows:

For all the POIs, $\mathcal{B}$ computes C_i and l_i using keys derived from K which is embedded in UMsg^*. It then sends (C_i, l_i) to the oracle O_M and is returned $\sigma_i = MAC_{\mathrm{MK}}(C_i, l_i)$.

$\mathcal{B}$ returns both Msg_1^* and Msg_3^* to $\mathcal{A}$, which finally outputs $\mathsf{Msg}_4^* = \{(\overline{C}_i, \overline{l}_i, \overline{\sigma}_i)\}$. Assume that $\mathcal{A}$ wins its game. We have that the user accepts Msg_4^*, and thus all the $\overline{\sigma}_i$'s are valid MAC tags on $(\overline{C}_i, \overline{l}_i)$'s, and that Msg_4^* is not a subset of the message Msg_4 that an honest QS would return to the user. Then there must exist some tuple $(\overline{C}_j, \overline{l}_j, \overline{\sigma}_j) \in \mathsf{Msg}_4^*$ which is different from all the tuples contained in Msg_3^*. $\mathcal{B}$ outputs $((\overline{C}_j, \overline{l}_j), \overline{\sigma}_j)$ as its forgery of the MAC scheme. It is readily seen that $\mathcal{B}$'s output is valid, since $\mathcal{B}$ did not ask its oracle O_M to produce a MAC tag $\overline{\sigma}_j$ on the message $(\overline{C}_j, \overline{l}_j)$. Therefore, the security of the MAC scheme is broken; hence, we have $\Pr[X_0] \leq |\Pr[X_1] - \Pr[X_0]| + \Pr[X_1]$ which is negligible. □



(a) Computation Cost (b) Communication Cost

Fig. 5. Number of POIs (NN queries).

5 Experimental Results

In this section, we evaluate the performance of our DGS for both continuous range and k-NN queries through simulations.

Baseline algorithm. We implemented a continuous spatial cloaking scheme using the *fully-trusted third party model* (TTP) [2]. TTP relies on a fully-trusted location anonymizer, which is placed between the user and the service provider (SP), to blur a querying user's location into a cloaked area that contains the querying user and a set of $\mathcal{K} - 1$ other users to satisfy the user specified $\mathcal{K}$-anonymity privacy requirement. To preserve the user's continuous location privacy, the location anonymizer keeps adjusting the cloaked area to contain the querying user and the $\mathcal{K} - 1$ users. A privacy-aware query processor at SP returns a set of candidate POIs to the querying user through the location anonymizer [6], [27]. Then, the querying user computes an exact query answer from the candidate POIs. We compare our DGS with the TTP scheme for both continuous range and k-NN queries.

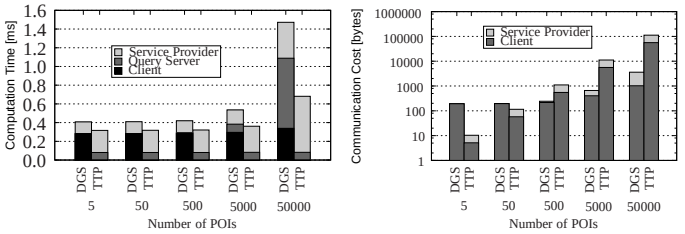
We chose TTP as the baseline algorithm to compare against, as it is architecturally most similar to our DGS approach in that both systems require third-party servers to perform the main computation of the respective algorithm (although DGS only requires a semi-trusted third party). Other approaches such as private information retrieval (PIR) or oblivious transfer (OT) are fundamentally different and put a much higher burden in terms of complexity of the computation on the user's side. They typically also compare unfavorably against TTP and our DGS in terms of communication bandwidth required (an important attribute in mobile environments), making the comparison between TTP and DGS the most equivalent one.

Simulated experiment. Our DGS and the TTP scheme are implemented in C++. For the cryptographic functions we use the OpenSSL library (http://www.openssl.org), GMP (http://www.gmplib.org) and PBC (http://crypto.stanford.edu/pbc/) for the IBE. In all experiments, we generate a set of moving objects on the real road map of Hennepin County, Minnesota, USA [28] with a total area of 1,571 km^2. Mobile users are initially distributed among the road segments in the road map proportional to the length of the roads, and then move along the roads at speeds of $30 - 70$ miles per hour. The experiments were run on a 64-bit machine with a 2.4GHz Intel CPU and 4GB RAM.

Performance metrics. We measure the performance of our DGS and the TTP scheme in terms of the average computation time per query on the client side, at the query server (QS) (or the location anonymizer for the TTP scheme), and at SP. We also measure the average communication cost per query between the user and QS, as well as the communication cost between QS and SP.

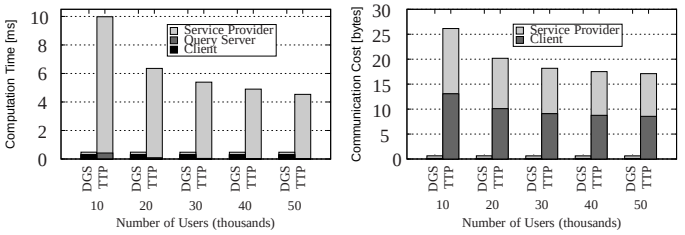
Parameter settings. Unless mentioned otherwise, the experiment considers 20,000 mobile users and 10,000 POIs. The default query

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing



(a) Computation Cost (b) Communication Cost

Fig. 6. Number of POIs (range queries).



(a) Computation Cost (b) Communication Cost

Fig. 7. Number of mobile users (NN queries).

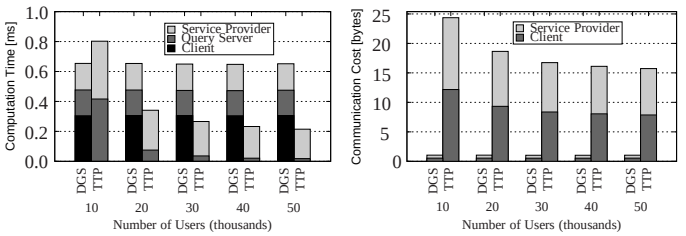
distance for range queries is 2km, and the default requested number of POIs for k-NN queries is $k = 10$. Since our DGS provides more secure continuous location privacy than the TTP scheme, we only consider a moderately high $\mathcal{K}$-anonymity level for the TTP scheme, where $\mathcal{K} = 200$ [2], to give a fair comparison.

Grid system. The grid system is created by laying a grid over a query area that is defined by its 4 corners. The grid divides the query area into non-overlapping, equal-sized cells (in terms of latitude/longitude). While the grid system does not require a specific coordinate system, for convenience we use WGS84 in our prototype. WGS84 is the coordinate system used for the Global Positioning System (GPS), and hence most easily applicable to mobile devices that contain a GPS receiver.

Points of interest. The POIs are generated randomly, by placing them onto vertices of the road map used in the experiment. For example, to generate 10,000 POIs, the prototype randomly picks a vertex of the road map as the location of a POI, repeating this 10,000 times.

5.1 Comparison of DGS with TTP

Although DGS and TTP have architectural similarities, DGS provides better location privacy and privacy guarantees than TTP for the two reasons: (1) In a TTP system, the user is only $\mathcal{K}$-anonymous, i.e., the user can be identified to be one of $\mathcal{K}$ users, but without being able to determine the exact user. In DGS, however, QS has no information at all to narrow down the anonymity set, while SP can only narrow the anonymity set down to the query area, but the query area can be chosen arbitrarily large by the user without negative performance impacts. Furthermore, TTP requires the cloaking area to expand as the user moves around, while in DGS the query area can stay fixed without an impact on the anonymity of the user. (2) The trusted third party in TTP needs to be fully trusted because it has access to all locations of all users in the system. In DGS, however, neither SP nor QS need to be fully trusted, as neither of them ever has access to the exact location of a user.



(a) Computation Cost (b) Communication Cost

Fig. 8. Number of mobile users (range queries).

In DGS, no entity has access to the exact location of users and the user's anonymity is not defined by a $\mathcal{K}$ value but by the query area, which can be chosen suitably large, so DGS provides better privacy guarantees than TTP. To achieve the same level of privacy with TTP compared to DGS, the $\mathcal{K}$ parameter in TTP would have to be chosen to correspond to the number of users present in the whole query area of DGS. However, the query area of DGS is typically chosen on a city-level, resulting in a large $\mathcal{K}$ value, e.g., tens of thousands of users. In TTP, however, $\mathcal{K}$ values are typically chosen in the lower hundreds, e.g., 200 [2]. This shows that DGS will by default provide much better privacy than TTP.

5.2 Number of POIs

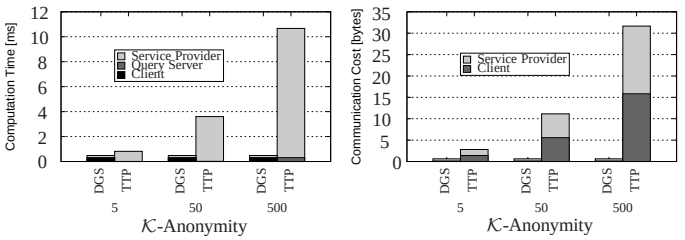
Fig. 5 shows the performance of our DGS and TTP for NN queries when varying the number of POIs several orders of magnitude from 5 to 50,000. The results show that DGS outperforms TTP (for a system with 50 or more POIs in total), as shown in Fig. 5a and 5b. The computation time for DGS is well below 1 ms for all cases, compared to TTP which is significantly more expensive (4 to 24 ms). The computation time of TTP also increases more quickly than DGS as the number of POIs increases above 500. This is mainly because TTP has to keep expanding cloaked areas to preserve the user's continuous location privacy. A larger cloaked area generally leads to a larger search area for a NN query which increases computation cost. For the communication cost, in DGS, most of the data is transferred between SP and QS, while the data transferred from QS to the user is small (one POI). However, in a system with more than 50 POIs, TTP requires a much larger amount of data to be transferred to the user, as the size of the user's cloaked area increases.

Fig. 6 shows the scalability of our DGS and TTP for continuous range queries when varying the number of POIs from 5 to 50,000. The computation cost of DGS is slightly larger than TTP (Fig. 6a) for POI databases with up to 5,000 POIs, and about twice as expensive for a database with 50,000 POIs. This is mainly due to the fact that the privacy-aware query processing cost for range queries using TTP is much lower than that for NN queries. Although our DGS incurs a higher computation cost than TTP due to the use of cryptographic primitives in DGS, DGS provides better location privacy and privacy guarantees for the user than TTP. Fig. 6b shows that for a system of more than 50 POIs, the communication cost of our DGS is smaller than in TTP, and significantly so for larger POI databases (e.g., 3.5KB for DGS versus 110KB for TTP for a database of 50,000 POIs), making our DGS more suitable for mobile environments.

5.3 Number of Mobile Users

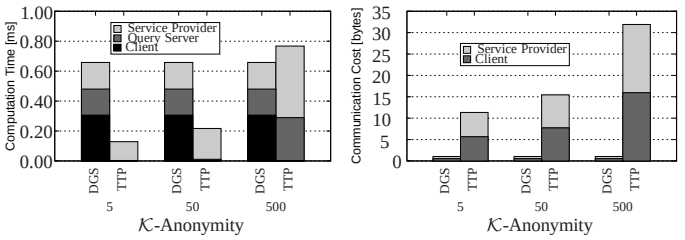
Fig. 7 and 8 depict the scalability of our DGS and TTP with respect to varying the number of mobile users from 10,000 to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing



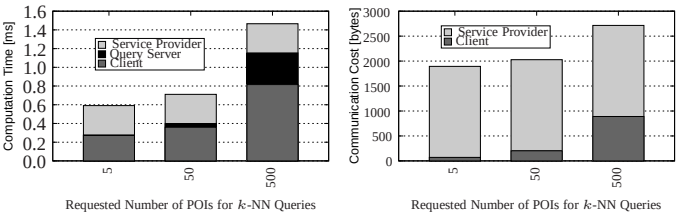
(a) Computation Cost (b) Communication Cost

Fig. 9. $\mathcal{K}$-anonymity levels (NN queries).



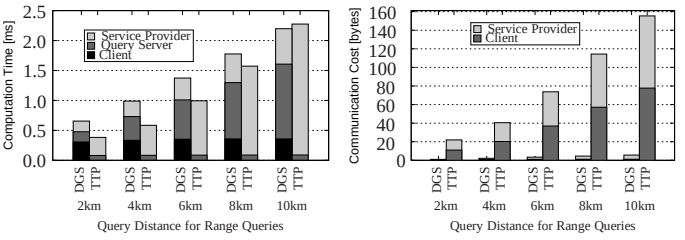
(a) Computation Cost (b) Communication Cost

Fig. 10. $\mathcal{K}$-anonymity levels (range queries).



(a) Computation Cost (b) Communication Cost

Fig. 11. The requested number of NN POIs (k).



(a) Computation Cost (b) Communication Cost

Fig. 12. Query distance for range queries.

50,000. The results show that DGS is independent of the number of users, while TTP depends heavily on the user density and/or the user distribution. Thus, DGS has the desirable privacy feature for privacy-preserving location-based services that it is free from privacy attacks based on the user distribution or density. Fig. 7 shows the performance of our DGS and TTP for continuous NN queries. The computation time of DGS remains constant, well below 1 ms (Fig. 7a). For TTP, the computation time is between 10 to 20 times higher than that of DGS, decreasing as the number of users increases. This is because the cloaked area computed by TTP becomes smaller with more users. Fig. 7b shows similar results for communication cost, which is constant for DGS, while significantly higher for TTP. Fig. 8 shows the results for continuous range queries. For $10,000$ users, TTP is slightly more expensive than DGS, in terms of computation cost (Fig. 8a), while DGS is two to three times more expensive than TTP for the number of users from $20,000$ to $50,000$. This can again be explained by the use of cryptographic functions that provide a much more secure scheme than TTP. However, the computation cost of DGS is constant, showing that it does not depend on the number of users. Fig. 8b also shows that the communication cost of DGS is lower than TTP and fairly constant in terms of bandwidth consumption.

5.4 $\mathcal{K}$-Anonymity Levels for the TTP Scheme

Fig. 9 and 10 show the results of increasing the $\mathcal{K}$-anonymity levels from 5 to 500 for continuous NN and range queries, respectively, while keeping the other parameters constant. Since our DGS uses a two-tier architecture (i.e., QS and SP) and cryptographic functions to preserve the user location privacy, its performance is not affected by the increase of $\mathcal{K}$. For DGS the query area was divided into 50×50 cells, and spanned $4 \times 4\text{km}^2$, and these parameters stayed fixed throughout the experiment. For NN queries, DGS performs much better than TTP, as shown in Fig. 9a and 9b, when the anonymity requirement becomes stricter (i.e., larger $\mathcal{K}$). This is because TTP has to generate larger cloaked areas to satisfy the stricter anonymity requirements. These larger cloaked areas incur higher query processing overhead at SP and lead to a larger set of candidate POIs returned to the user. Since the overhead for processing range queries in TTP is much lower than that for NN queries, the computation cost of TTP is better than DGS for low anonymity levels, but it performs worse than DGS for stricter anonymity levels, i.e., when $\mathcal{K} \geq 500$ (Fig. 10a). In terms of communication cost (Fig. 10b), DGS is much less expensive than TTP. The communication cost of TTP increases with $\mathcal{K}$, because more POIs have to be returned from SP to the user through the fully-trusted third party, as the cloaked area size gets larger.

5.5 Query Parameters

Fig. 11 shows the performance of our DGS when increasing the requested number of POIs (k) for k-NN queries from 5 to 500. As depicted in Fig. 11a, the computation cost of DGS increases as k gets larger. The increase in the computation cost only affects the user and QS, the workload for SP remains the same in the experiments even when k increases. This is because an increase of k does not increase the size of the query area, which is the only parameter relevant to SP. The communication cost also increases steadily (Fig. 11b). However, the increase of k only results in a higher communication cost for the user, as the user requests more POIs in the vicinity. Also, for this experiment we increased the query area from the default of 2km to 10km to ensure that enough POIs were returned to the client, especially for the case of $k = 500$ POIs. Fig. 12 shows the effect on the performance of DGS and TTP when increasing the query distance of range queries from 2km to 10km. The computation cost of DGS and TTP is comparable (Fig. 12a), with DGS slightly more expensive than TTP for smaller query distances, while it performs better than TTP as the query distance gets larger (i.e., the query distance is equal to or larger than 10km). For communication cost (Fig. 12b), DGS is much better than TTP as the query distance increases.

5.6 Mobile Device Performance

In a real-world scenario, however, some of the operations will be performed on a mobile device. While mobile devices have become

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

TABLE 1
Benchmark of cryptographic operations performed on a mobile device.

	Java	Native
IBE Encryption (per request)	1074ms	52.2ms
Hash / Encryption (per request)	0.147ms	0.0046ms (4.6μs)
AES Decryption (per POI)	0.24ms	0.0042ms (4.2μs)

quite powerful, we decided to run additional benchmarks of our scheme on actual mobile devices to verify that the cryptographic operations necessary in our protocol can be run efficiently on mobile devices. In our protocol, the mobile device has to perform three operations that necessitate cryptographic calculations: **(1) Request Generation**. The initial request generation by the user requires encrypting the request using IBE over elliptic curves, which is expensive in terms of computational power required. **(2) Hashing / Encryption**. To retrieve the matching POIs from QS, the client needs to hash and then encrypt the grid indices for each grid cell that it is interested in. This involves one hashing (SHA256) and one symmetric encryption operation (AES). **(3) POI Decryption**. Once the client receives the response from QS, it needs to decrypt the POIs to display them locally. This requires the client to perform one decryption operation per POI using a symmetric cipher (AES).

Benchmark setup. The performance benchmarks for mobile devices were done on a Samsung Galaxy SIII Android smartphone that has a quad-core Cortex-A9 1.4GHz CPU and 1GB RAM. Android uses Java to write applications, but it also allows to run code natively by cross-compiling for the ARM architecture and then calling native code from within an application written in Java. For the benchmark we first evaluated the performance of doing the cryptographic operations in Java, using built-in symmetric cryptography and JPBC (http://gas.dia.unisa.it/projects/jpbc/index.html), a Java port of PBC, for performing IBE. We then also cross-compiled the libraries used in the prototype (OpenSSL, GMP and PBC) for ARM to evaluate the performance when running the cryptographic operations natively on the device.

Results. Table 1 shows the benchmark results for IBE encryption and AES decryption that refer to encrypting one request for SP and decrypting one POI received from QS, respectively. These results show that smartphones are easily capable of performing the cryptographic operations necessary in our protocol. While IBE is still a somewhat expensive operation requiring on the order of 50ms, this is an infrequent request and hence not a bottleneck. Hashing / encrypting and AES decryption on the other hand are much more frequent operations in our protocol, but the operations are very efficient when done natively, so that a mobile device can easily decrypt up to several hundred thousand POIs per second, or even more if multiple cores are used (at which point bandwidth is more likely to become the bottleneck).

6 Related Work

Spatial cloaking techniques have been widely used to preserve user location privacy in LBS. Most of the existing spatial cloaking techniques rely on a fully-trusted third party (TTP), usually termed *location anonymizer*, that is required between the user and the service provider (e.g., [1]–[8]). When a user subscribes to LBS, the location anonymizer will blur the user's exact location into a cloaked area such that the cloaked area includes at least $k-1$ other users to satisfy k-anonymity. The TTP model has four major drawbacks. (a) It is difficult to find a third party that can be fully trusted. (b) All users need to continuously update their locations with the location anonymizer, even when they are not subscribed to any LBS, so that the location anonymizer has enough information to compute cloaked areas. (c) Because the location anonymizer stores the exact location information of all users, compromising the location anonymizer exposes their locations. (d) k-anonymity typically reveals the approximate location of a user and the location privacy depends on the user distribution. In a system with such *regional location privacy* it is difficult for the user to specify personalized privacy requirements. The feeling-based approach [29] alleviates this issue by finding a cloaked area based on the number of its visitors that is at least as popular as the user's specified public region.

Although some spatial clocking techniques can be applied to peer-to-peer environments [30]–[32], these techniques still rely on the k-anonymity privacy requirement and can only achieve regional location privacy. Furthermore, these techniques require users to trust each other, as they have to reveal their locations to other peers and rely on other peers' locations to blur their locations. In [33], another distributed method was proposed that does not require users to trust each other, but it still uses multiple TTPs. Another family of algorithms uses incremental nearest neighbor queries, where a query starts at an "anchor" location which is different from the real location of a user and iteratively retrieves more points of interest until the query is satisfied [34]. While it does not require a trusted third party, the approximate location of a user can still be learned; hence only regional location privacy is achieved.

Cryptographic tools were used to protect outsourcing data. An order-preserving encryption scheme [35] uses a bucket-based encryption E such that $E(x) < E(y)$ for every pair of values for which $x < y$. However, there does not seem to be a straightforward way to extend it to protect spatial data. Another approach described in [36] for outsourcing data uses homomorphic encryption to enable aggregate SQL queries over encrypted databases. The scope focuses only on simple numerical domains and aggregate queries in SQL. This approach has also been shown to be insecure in [37].

For spatial data, another family of privacy-preserving techniques uses cryptographic tools such as private information retrieval (PIR) or oblivious transfer (OT). PIR allows a user to retrieve a POI from a database without the server knowing which POI was retrieved. OT has the additional property that the user only learns the requested POI and does not learn anything about any other POI. Ghinita et al. proposed a PIR-based scheme which eliminates the trusted location anonymizer [9]. Their work uses a PIR matrix with n POIs in total and size $t \times t$ with $t = \lceil \sqrt{n} \rceil$. Using PIR a user can retrieve POIs only column-wise, corresponding to $O(\sqrt{n})$ POIs for each request. This is significantly more expensive than just retrieving the $O(1)$ relevant POIs. Their experimental results show that the communication overhead of their scheme is much higher than that of using the TTP model.

Vishwanathan et al. proposed to use a two-level combination of PIR and OT [11]. First, a user selects the appropriate column in a grid using PIR and then uses OT to retrieve the exact grid cell. Their approach focuses on protecting the data of the database system by allowing the user to only learn the POIs in the current grid cell of the user. Because of the nature of PIR, however, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2388488, IEEE Transactions on Mobile Computing

user still needs to receive the whole column (and thus $O(\sqrt{n})$ points of interest). A scheme proposed in [10] uses OT to hide users' locations from a service provider while enabling a payment infrastructure, but the scheme still requires a proxy as a TTP.

Also studied for privacy in LBS are methods which work on encrypted or transformed data. For example, Khoshgozaran and Shahabi proposed a system which uses Hilbert curves to map locations into a different space and then solves NN queries in the transformed space [38]. A similar approach but using encryption was proposed by Wong et al. in [39]. Their work focuses on outsourcing a database in encrypted format to a service provider and allows users to perform k-NN queries on the encrypted database. Their focus, however, is more on protecting the database instead of the privacy of the users. Similar work was done in [40].

A few privacy-preserving techniques have attempted to use the TTP model for continuous LBS [2], [7], [41]. The idea of [2] is to keep expanding an initial cloaked area to include at least the same k users, [7] is to predict a user's footprints and blur each footprint into a k-anonymized area, and [41] is to use a mix-zone to make the users located in there at the same time indistinguishable. The TTP model has been extended to protect the privacy of an anonymized group of users by generalizing their spatial query regions to make their queries indistinguishable [42] and guarantee that the number of their requested service values is at least m to achieve m-invariance [43]. Temporal cloaking and encryption techniques are used for aggregate traffic data collection [44], but they cannot provide privacy-preserving continuous LBS.

Another technique proposed to protect continuous LBS is using dummy queries together with a real query [45]. However, this technique issues more queries than the user really needs. In our system a user never transmits actual location information (apart from the query area, which is only seen by the service provider and which can be chosen arbitrarily large), and the type of POI in a query can only be read by the service provider. The query server has even less information available, it does not know the query area nor the type of POI searched for.

7 Conclusion

In this paper, we proposed a dynamic grid system (DGS) for providing privacy-preserving continuous LBS. Our DGS includes the query server (QS) and the service provider (SP), and cryptographic functions to divide the whole query processing task into two parts that are performed separately by QS and SP. DGS does not require any fully-trusted third party (TTP); instead, we require only the much weaker assumption of no collusion between QS and SP. This separation also moves the data transfer load away from the user to the inexpensive and high-bandwidth link between QS and SP. We also designed efficient protocols for our DGS to support both continuous k-nearest-neighbor (NN) and range queries. To evaluate the performance of DGS, we compare it to the state-of-the-art technique requiring a TTP. DGS provides better privacy guarantees than the TTP scheme, and the experimental results show that DGS is an order of magnitude more efficient than the TTP scheme, in terms of communication cost. In terms of computation cost, DGS also always outperforms the TTP scheme for NN queries; it is comparable or slightly more expensive than the TTP scheme for range queries.

Acknowledgments

Chi-Yin Chow was partially supported by the Guangdong Natural Science Foundation (No. S2013010012363) and a research grant (CityU Project No. 9231131). Qiong Huang was supported by the National Natural Science Foundation of China (No. 61472146) and the Guangdong Natural Science Foundation (No. S2013010011859). Duncan S. Wong was supported by a research grant from the Research Grants Council, HKSAR (Project No. CityU 121512).

References

[1] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with PrivacyGrid," in *WWW*, 2008.
[2] C.-Y. Chow and M. F. Mokbel, "Enabling private continuous queries for revealed user locations," in *SSTD*, 2007.
[3] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE TMC*, vol. 7, no. 1, pp. 1–18, 2008.
[4] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *ACM MobiSys*, 2003.
[5] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," *IEEE TKDE*, vol. 19, no. 12, pp. 1719–1733, 2007.
[6] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: Query processing for location services without compromising privacy," in *VLDB*, 2006.
[7] T. Xu and Y. Cai, "Location anonymity in continuous location-based services," in *ACM GIS*, 2007.
[8] ——, "Exploring historical location data for anonymity preservation in location-based services," in *IEEE INFOCOM*, 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 *ACM SIGMOD*, 2008.
[10] M. Kohlweiss, S. Faust, L. Fritsch, B. Gedrojc, and B. Preneel, "Efficient oblivious augmented maps: Location-based services with a payment broker," in *PET*, 2007.
[11] R. Vishwanathan and Y. Huang, "A two-level protocol to answer private location-based queries," in *ISI*, 2009.
[12] J. M. Kang, M. F. Mokbel, S. Shekhar, T. Xia, and D. Zhang, "Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors," in *IEEE ICDE*, 2007.
[13] C. S. Jensen, D. Lin, B. C. Ooi, and R. Zhang, "Effective density queries of continuously moving objects," in *IEEE ICDE*, 2006.
[14] S. Wang and X. S. Wang, "AnonTwist: Nearest neighbor querying with both location privacy and k-anonymity for mobile users," in *MDM*, 2009.
[15] W. B. Allshouse, W. B. Allshousea, M. K. Fitchb, K. H. Hamptonb, D. C. Gesinkc, I. A. Dohertyd, P. A. Leonebd, M. L. Serrea, and W. C. Millerb, "Geomasking sensitive health data and privacy protection: an evaluation using an E911 database," *Geocarto International*, vol. 25, pp. 443–452, October 2010.
[16] A. Gkoulalas-Divanis, P. Kalnis, and V. S. Verykios, "Providing k-anonymity in location based services," *SIGKDD Explor. Newsl.*, vol. 12, pp. 3–10, November 2010.
[17] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, 2001.
[18] A. Menezes, M. Qu, and S. Vanstone, "Some new key agreement protocols providing mutual implicit authentication," in *SAC*, 1995.
[19] S. Yau and H. An, "Anonymous service usage and payment in service-based systems," in *IEEE HPCC*, 2011, pp. 714–720.
[20] M. Balakrishnan, I. Mohomed, and V. Ramasubramanian, "Where's that phone?: Geolocating ip addresses on 3G networks," in *ACM SIGCOMM IMC*, 2009.
[21] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *USENIX Security*, 2004.
[22] G. Bissias, M. Liberatore, D. Jensen, and B. Levine, "Privacy vulnerabilities in encrypted HTTP streams," in *PET*, 2006.
[23] P. Golle and K. Partridge, "On the anonymity of home/work location pairs," in *Pervasive Computing*, 2009.
[24] IEEE, *P1363-2000: Standard Specifications for Public-Key Cryptography*, 2000.
[25] A. B. Lewko and B. Waters, "Efficient pseudorandom functions from the decisional linear assumption and weaker variants," in *ACM CCS*, 2009.

[26] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, 1986.
[27] C.-Y. Chow, M. F. Mokbel, and W. G. Aref, "Casper*: Query processing for location services without compromising privacy," *ACM TODS*, vol. 34, no. 4, 2009.
[28] U.S. Census Bureau, "TIGER. http://www.census.gov/geo/www/tiger/."
[29] T. Xu and Y. Cai, "Feeling-based location privacy protection for location-based services," in *ACM CCS*, 2009.
[30] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *ACM GIS*, 2006.
[31] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "MOBIHIDE: A mobile peer-to-peer system for anonymous location-based queries," in *SSTD*, 2007.
[32] ——, "PRIVE: Anonymous location-based queries in distributed mobile systems," in *WWW*, 2007.
[33] G. Zhong and U. Hengartner, "A distributed k-anonymity protocol for location privacy," in *IEEE PerCom*, 2009.
[34] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu, "SpaceTwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services," in *IEEE ICDE*, 2008.
[35] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order-preserving encryption for numeric data," in *ACM SIGMOD*, 2004.
[36] H. Hacigümüş, B. Iyer, and S. Mehrotra, "Efficient execution of aggregation queries over encrypted relational databases," in *DASFAA*, 2004.
[37] E. Mykletun and G. Tsudik, "Aggregation queries in the database-as-a-service model," in *DBSec*, 2006.
[38] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *SSTD*, 2007.
[39] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *ACM SIGMOD*, 2009.
[40] M. L. Yiu, G. Ghinita, C. S. Jensen, and P. Kalnis, "Enabling search services on outsourced private spatial data," *VLDB Journal*, vol. 19, no. 3, pp. 363–384, 2010.
[41] B. Palanisamy and L. Liu, "Mobimix: Protecting location privacy with mix zones over road networks," in *IEEE ICDE*, 2011.
[42] S. Mascetti, C. Bettini, X. S. Wang, D. Freni, and S. Jajodia, "ProvidentHider: An algorithm to preserve historical k-anonymity in LBS," in *MDM*, 2009.
[43] R. Dewri, I. Ray, I. Ray, and D. Whitley, "Query m-Invariance: Preventing query disclosures in continuous location-based services," in *MDM*, 2010.
[44] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. M. Bayen, R. Herring, J. C. Herrera, M. Gruteser, M. Annavaram, and J. Ban, "Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines," *IEEE TMC*, vol. 11, no. 5, pp. 849–864, 2012.
[45] A. Pingley, N. Zhang, X. Fu, H.-A. Choi, S. Subramaniam, and W. Zhao, "Protection of query privacy for continuous location based services," in *IEEE INFOCOM*, 2011.

Roman Schlegel has an MSc from EPFL in Switzerland in Communication Systems and a PhD in Computer Science from City University in Hong Kong. During his doctoral studies he also spent a year as a research assistant at Indiana University Bloomington in the US. After finishing his PhD he joined ABB Corporate Research as a research scientist for security in industrial control systems. His research interests include privacy, network security and applied cryptography.

Chi-Yin Chow received the B.A. and M.Phil. degrees from The Hong Kong Polytechnic University in 2002 and 2005, respectively. He received the M.S. and Ph.D. degrees from the University of Minnesota-Twin Cities in 2008 and 2010, respectively. He is currently an assistant professor in Department of Computer Science, City University of Hong Kong. His research interests include spatio-temporal data management and analytics, GIS, mobile computing, location-based services, and data privacy. He was the co-organizer of ACM SIGSPATIAL MobiGIS 2012, 2013, and 2014.

Qiong Huang got his B.S. and M.S. degrees from Fudan University in 2003 and 2006 respectively, and got his PhD degree from City University of Hong Kong in 2010. Now he is a professor at South China Agricultural University. His research interests include cryptography and information security, in particular, cryptographic protocols design and analysis.

Duncan S. Wong received the B.Eng. degree from the University of Hong Kong in 1994, the M.Phil. degree from the Chinese University of Hong Kong in 1998, and the Ph.D. degree from Northeastern University, Boston, MA, in 2002. He is currently an associate professor in the Department of Computer Science at the City University of Hong Kong. His primary research interest is cryptography; in particular, cryptographic protocols, encryption and signature schemes, and anonymous systems. He is also interested in other topics in information security, such as network security, wireless security database security, and security in cloud computing.