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

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.

EXISTING SYSTEM:

1. 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.
2. 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.
3. In a system with such *regional location privacy* it is difficult for the user to specify personalized privacy requirements. The feeling based approach 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, these techniques still rely on the k-anonymity privacy requirement and can only achieve regional location privacy.
4. 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, another distributed method was proposed that does not require users to trust each other, but it still uses multiple TTPs.
5. 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. 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.

DISADVANTAGES OF EXISTING SYSTEM:

- The TTP model has four major drawbacks.
- It is difficult to find a third party that can be fully trusted.
- 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.
- Because the location anonymizer stores the exact location information of all users, compromising the location anonymizer exposes their locations.
- k-anonymity typically reveals the approximate location of a user and the location privacy depends on the user distribution.

PROPOSED SYSTEM:

1. In this paper, we propose a user-defined privacy grid system called *dynamic grid system* (DGS) to provide privacy-preserving *snapshot* and *continuous* LBS.
2. 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.
3. *Semi-trusted* in this context means that while 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.
4. 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.
5. 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 he encrypted query to SP specified by the user. SP decrypts the query and selects the POIs within the query area from its database.

ADVANTAGES OF PROPOSED SYSTEM:

1. 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.
2. 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.
3. After the user receives the encrypted POIs, she decrypts them to get their exact locations and computes a query answer.

MODULES:

- Service providers
- Mobile users
- Query servers
- Supported spatial queries

MODULES DESCSRIPTION:

Service providers

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 our system architecture, 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. 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 the query. 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.

Query servers

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. The control/data flows of our DGS are as follows: 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.

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 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 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 and density queries because recent research efforts have shown that the answer of these queries can be maintained by monitoring a region. 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, and the second phase incrementally maintains the query answer based on the user's location updates.

SYSTEM REQUIREMENTS:
HARDWARE REQUIREMENTS:

System	:	Pentium IV 2.4 GHz.
Hard Disk	:	40 GB.
Floppy Drive	:	1.44 Mb.
Monitor	:	15 VGA Colour.
Mouse	:	Logitech.
Ram	:	512 Mb.

SOFTWARE REQUIREMENTS:

Operating system	:	Windows XP/7.
Coding Language	:	JAVA/J2EE
IDE	:	Netbeans 7.4
Database	:	MYSQL