A Grid System For Never Ending Location-Based Services

[1]Shaik Shakeera, [2]Shaik Asha
[1]Assistant Professor, [2]Assistant Professor
[1]Department of Computer Science and Engineering , [2]Department of Computer Science and Engineering
[1]KG Reddy College of Engineering and Technology, Moinabad, Hyderabad, India
[2]KG Reddy College of Engineering and Technology, Moinabad, Hyderabad, India

Abstract: The major aim of Location Based Services (LSB) is based on the location, it is used to provide services like food ordering system, Google map and some kind of e-shopping websites are completely based on locations. Based on the user location LSB provide continuous report to the un trusted server to obtain the services which can avoid privacy risks.LBS have existing privacy-preserving techniques as requiring a fully-trusted third party, it offers limited privacy guarantees and incurring high communication overhead. In this paper, we propose a user-defined privacy grid system which is called as dynamic grid system (DGS). Privacy-preserving snapshot and continuous LBS have four effective requirements. . (1) A semi-trusted third party is responsible for matching operations correctly and it does not have any information about a user's location. (2) Secure snapshot and continuous location privacy is guaranteed to defined adversary models. (3) The communication cost for the user does not depend on the user's desired privacy level; it depends on the number of relevant points of interest in the domain of the user. (4) The system can be easily enlarged to support other spatial queries without changing the algorithms run by the semi-trusted third party and the database server. Experimental results show that the DGS is more efficient than the state-of-the-art privacy-preserving technique for continuous LBS.

Keywords: **Grid system, Location Based services (LBS), Location privacy, Dynamic Grid system.**

I.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). 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. LBS can be very valuable. The consumer market for location-based services (LBS) is estimated to grow from 2.9 billion dollars in 2010 to 10.4 billion dollars in 2015. While navigation applications are currently generating the most significant revenues, location based advertising and local search will be driving the revenues going forward. The legal landscape, unfortunately, is unclear about what happens to a subscriber's location data[1]. The nonexistence of regulatory controls has led to a growing concern about potential privacy violations arising out of the usage of a location-based application. While new regulations to plug the loopholes are being sought, the privacy conscious user currently feels reluctant to adopt one of the most functional business models of the decade. Privacy and usability are two equally important requirements for successful.

A user-defined privacy grid system is 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. The idea of DGS is first querying the user determines a query area, where as the user is comfortable to reveal the fact that they are in somewhere within this query area. The query area is divided into fixed-sized grid cells based on the dynamic grid structure specified by the user. Then, the user encrypts a query that includes the data 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[2]. 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. Simply QS 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[3], 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 and density queries

II. SYSTEM ARCHITECTURE

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.

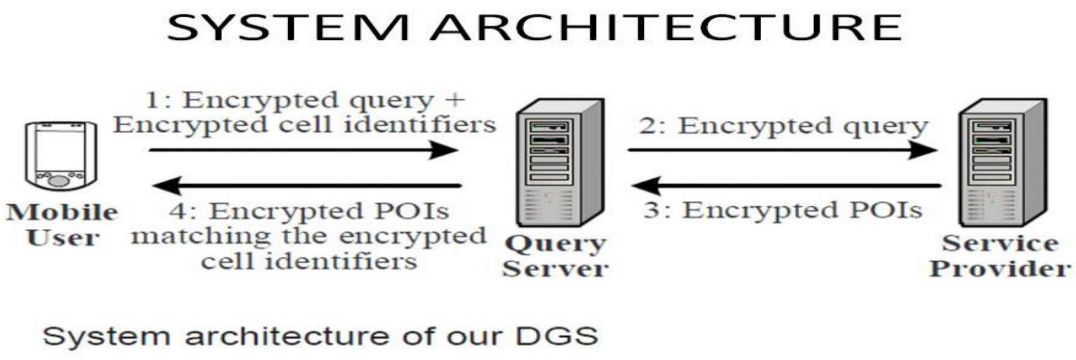

Fig. 1: System Architecture of DGS

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 (xu, yu). 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 distinguish 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[4]. (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 3 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[5] 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 and health data. (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

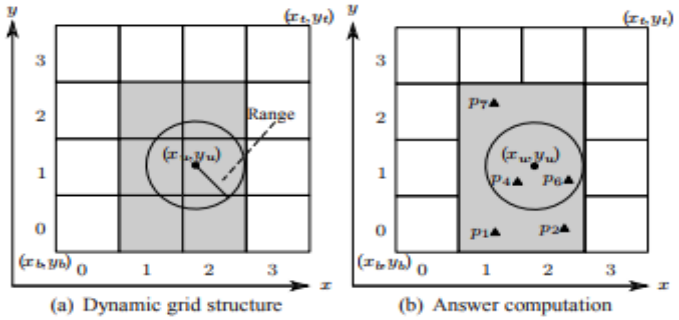

Fig. 2: Example of range query Processing in DGS

III.DYNAMIC GRID SYSTEM

DGS for processing continuous range queries and incrementally maintaining their answers, DGS to support k-NN queries.
In general, the privacy-preserving range query processing protocol has 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 (xb, yb) and top-right vertex (xt, yt). Notice that the user is not necessarily 5 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 × m equal-sized grid cells to construct a dynamic grid structure, where m is a user-specified parameter[6].

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, Se, for a QS. The user first selects a random key K and derives three distinct keys:

$$(HK,EK, MK) \leftarrow KDF(K) \quad (1)$$

where KDF(·) is a key derivation function ([24]). Then, the user sets query and Se as follows: (1) Query generation. An encrypted query for a specific SP is prepared as:

$$query \leftarrow IBE.EncSP (POI\text{-}type, K, m,(xb, yb),(xt, yt)) \quad (2)$$

where IBE.EncSP (·) is Identity-Based Encryption (IBE) under the identity of SP. 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, (xb, yb), and (xt, yt). (2) Encrypted identifier generation[12]. Given the query region of the range query, the user selects a set of grid cells Sc in the dynamic grid structure that intersect the query region, i.e., a circle centered at the user's current location (xu, yu) with a radius of Range. For each selected grid cell i in Sc, its identity (ci , ri) is encrypted to generate an encrypted identifier:

$$hi \leftarrow H(ci , ri) \quad (3)$$
$$Ci \leftarrow SE.EncHK(hi) \quad (4)$$

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

$$request \leftarrow hSP, query, Sei \quad (5)$$

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 Sc, and thus, the user has to encrypt each identity of these grid cells[7].

IV. SECURITY ANALYSIS

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 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.

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 is a *QS*, and the challenger C plays the roles of the client and all the service providers.

V. EXPERIMENTAL RESULTS

Evaluate the performance of our DGS for both continuous range and *k*-NN queries through simulations.
Baseline algorithm. We implemented a continuous spatial cloak- ing scheme using the *fully-trusted third party model* (TTP). TTP relies on a fully-trusted location anonymizer, which is placed[8],[9] 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 1 other users to satisfy the user specified - 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 1 users. A privacy-aware query processor at *SP* returns a set of candidate POIs to the querying user through the location anonymizer .

Simulated experiment.

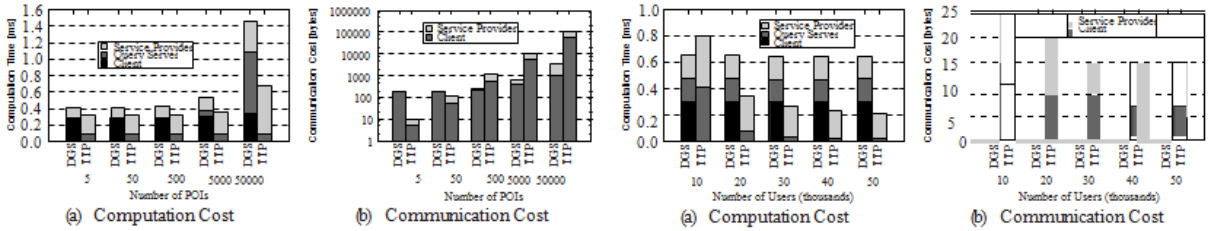

Fig. 3: Number of POIS in Queries

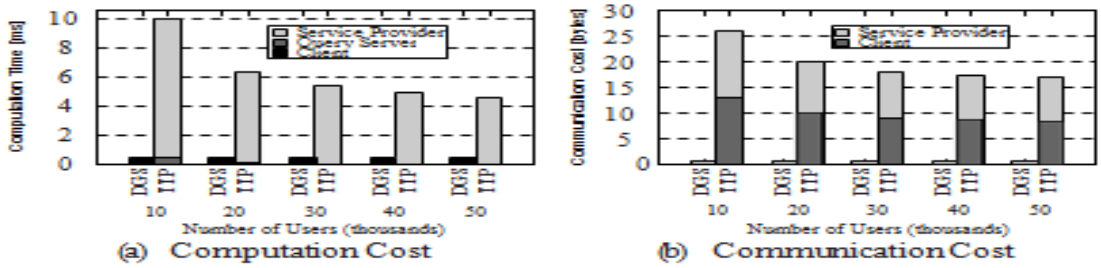

Fig. 4: Number of Mobile users

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 - anonymous, i.e., the user can be identified to be one of 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,[10] 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.

Number of POIs

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.

Number of Mobile Users

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. Figure shows the performance of our DGS and TTP for continuous NN queries. The computation time of DGS remains constant, well below 1 ms. For TTP, the computation time is between 10 to 20 times higher than that of DGS, decreasing as the number of user's increases. This is because[11] 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 are 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. The following figure shows that the communication cost of DGS is lower than TTP and fairly constant in terms of bandwidth consumption.

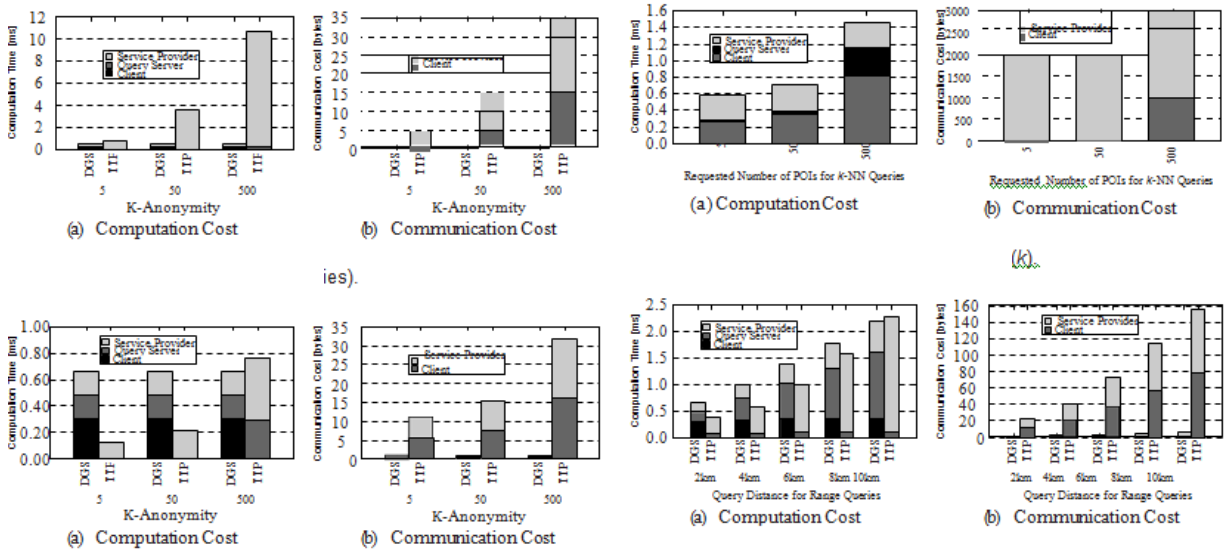

Fig. 5: Communication cost of DGS and TTP

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 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[13],[14]. In our protocol, the mobile device has to per- form 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).

VI. CONCLUSION

In this paper a dynamic grid system (DGS) for providing privacy-preserving continuous LBS. Our DGS includes the query server (*QS*) and the service provider (*SP)*, and crypto- graphic 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[15], 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.

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 pro- cessing 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, "Pri- vate 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.