Unchained Cellular Obfuscation Areas for Location Privacy in Continuous Location-Based Service Queries

Jia-Ning Luo¹ and Ming-Hour Yang²

¹Department of Information and Telecommunications Engineering, Ming Chuan University, Taoyuan, Taiwan
²Department of Information and Computer Engineering, Chung Yuan Christian University, Taoyuan, Taiwan

Correspondence should be addressed to Ming-Hour Yang; mhyang@cycu.edu.tw

Received 9 February 2017; Revised 6 July 2017; Accepted 10 August 2017; Published 28 September 2017

Academic Editor: Christos Goumopoulos

To access location-based service (LBS) and query surrounding points of interest (POIs), smartphone users typically use built-in positioning functions of their phones when traveling at unfamiliar places. However, when a query is submitted, personal information may be leaked when they provide their real location. Current LBS privacy protection schemes fail to simultaneously consider real map conditions and continuous querying, and they cannot guarantee privacy protection when the obfuscation algorithm is known. To provide users with secure and effective LBSs, we developed an unchained regional privacy protection method that combines query logs and chained cellular obfuscation areas. It adopts a multiuser anonymizer architecture to prevent attackers from predicting user travel routes by using background information derived from maps (e.g., traffic speed limits). The proposed scheme is completely transparent to users when performing continuous location-based queries, and it combines the method with actual road maps to generate unchained obfuscation areas that conceal the actual locations of users. In addition to using a caching approach to enhance performance, the proposed scheme also considers popular tourist POIs to enhance the cache data hit ratio and query performance.

1. Introduction

Currently, most mobile devices feature built-in positioning functions, and smartphone users frequently use location-based services (LBS) to query points of interest (POIs) within their vicinity (e.g., when searching for Chinese restaurants within a 10 km radius). Although using LBSs to rapidly locate places and routes is highly convenient, LBS providers may exploit the opportunity to collect the query contents and travel routes of specific users and then analyze these datasets to determine the users’ dietary habits, shopping preferences, and even personal medical histories. These behaviours are a severe breach of LBS user’ right to privacy.

Numerous previous scholars [1, 2] developed peer-to-peer (P2P) cloaking algorithms to mask the identity and location of users to guarantee location privacy. These P2P algorithms satisfy k-anonymity by sharing the user location with other users. However, the approaches proposed in those studies search for k − 1 conspirators surrounding the user, which may enable attackers to triangulate a user within an obfuscation area (OA) and deploy a variance-based attack (VBA) [3]. In [3], an approach was proposed that searches for other conspirators surrounding a user. Subsequently, a random conspirator in the group is selected to search for other conspirators. This process is repeated until the k-anonymity requirement is satisfied; that is, the user cannot be triangulated within an obfuscation area. Subsequently, P2P necessitates the exchange of location information between users. Therefore, users are required to trust other users in the obfuscation area. A malicious user could select different k to obtain the location of the other users by using the k-anonymity algorithm in [3]. They may even partner with LBS providers to steal personal data from regular users, increasing the risk of privacy leaks.

Recent studies have proposed methods for masking the identity [4, 5], location [6–8], and query information [9] of users by using secure third-party anonymizers to encode the location of a user or POI. Anonymizers not only protect
user privacy but also reduce communication time and costs. One study [4] proposed using an anonymizer to mask the identities of a group of query users by using the identity of one random user in the group. These queries, which contain the same metadata, are transmitted to the LBS server. Another study [7] used a Hilbert curve to create an obfuscation area to mask the location of users. Anonymizers mask users by randomly selecting a representative user in proximity to a group of users. The metadata of the representative are copied to all queries before they are transmitted to the LBS server, thereby satisfying $k$-anonymity and obfuscation requirements. Anonymizers typically create obfuscation areas in grid [7, 10–12] or pyramid structure to mask user locations. In [13], a method was proposed to resolve the incompatibility between the original obfuscation area and query criteria by creating an additional obfuscated query area to keep privacy.

Even when a user is masked within an obfuscation area to satisfy $k$-anonymity, LBS servers can collect user queries for area information when they submit continuous LBS queries in a short period. Users are more likely to use LBSs in unfamiliar rural tourist locations (rather than in urban areas) where roads are more dispersed. The simpler road network structures of rural areas enable LBS servers to determine the locations of users by analyzing maps and road conditions. To prevent LBS servers from cross-referencing continuous queries to obtain user location information, previous scholars have added reachable query routes to confuse LBS servers [14–18]. However, LBS servers can extrapolate known data, such as user habits, interests, and actual maps, to determine the most probable route of travel through an elimination process. In [19], a method was proposed to determine reasonable POIs within a user’s query area by analyzing his or her past query records. Subsequently, the user’s actual location is combined with a corresponding reasonable POI to generate a dummy query, preventing LBS servers from filtering out unreasonable dummy queries. A subsequent study [20] proposed a method that selects a nearby insensitive location from a user’s past travel routes to substitute sensitive query locations. However, this method was prone to leak the query location because it failed to account for map data and user mobility. In response, another study [17] combined an anonymizer with map data (all intersection branches within the road network) and user mobility. To confuse LBS servers, the anonymizer used in that study generated obfuscation areas that include the section of road extending from the user’s current intersection, but they do not include blind alleys or overlapping routes according to the user’s privacy requirements.

In this study, we proposed a method combining the anonymizer provided by trusted third-party servers with actual road maps and users’ movement patterns to create multiple virtual paths. When user content cannot be detected in the cache, the mechanism is applied to guarantee the privacy of user queries. The proposed method provides users with high query performance when the query volume is high while guaranteeing location privacy. The method uses the popular query characteristics of tourist locations to enhance the cache hit ratio, query performance, and protection of users’ POI and query locations. The proposed method also considers similarities between pseudoqueries and users’ actual queries, as well as cached POIs, to prevent the generated pseudoqueries from being filtered out by the LBS server, thereby increasing the cache life and hit ratio. The proposed method in the present study is suitable for continuous queries. It has the following contributions:

1. The privacy of users’ POIs is maintained, even during continuous querying.
2. POIs that are difficult for LBS servers to filter out are generated by incorporating area characteristics, logs, and user queries.
3. User privacy requirements are satisfied, even when the location obfuscation algorithm is known to the attacker.
4. Obfuscation areas are generated from real-time maps, thereby avoiding exposing user locations.
5. Cache data are used to reduce the communication costs and time of the anonymizer and LBS server.

The remainder of this paper is organized as follows. Section 2 describes the system architecture and initialization phase, and Section 3 discusses the development of the proposed method. The security analysis and the performance analysis are discussed in Section 4, and Section 5 presents the conclusion.

2. System Architecture

The system architecture is illustrated in Figure 1. When multiple users access LBSs to submit queries, the queries are transmitted to a trusted obfuscated server to protect user privacy. An anonymizer cross-references the query content with the cache database. If POI data matching the query content are detected, the query results are encrypted and returned to the users. In Figure 1, the queries “night market” and “super market” are returned to users from the anonymizer (indicated by the dotted line). If relevant data are not cached, the anonymizer obfuscates the user's query and location and transmits the obfuscated query to the LBS server. In Figure 1, the user’s “fast food” query and location are obfuscated and transmitted to the LBS server (indicated by the solid line). Once the anonymizer receives the POIs within the obfuscation area from the LBS server, it updates the cache database, filters out the pseudodata, encrypts the query results, and returns the results to the user.

To reduce the computation load of the LBS server for processing user queries, the proposed method uses cell numbers instead of coordinates to represent the query range sent to the LBS server. However, this process necessitates additional computations and transmission costs to synchronize the maps, cell sizes, and cell numbers on the anonymizer and LBS server. Accordingly, we adopted a numbering system for the cellular structure to reduce the overhead costs. This method synchronizes only the center point of the map and the sides of the cells to maintain consistent map segregation and numbering between the anonymizer and LBS server.

The proposed method adopts a trusted anonymizer to protect users’ queries from being collected by the LBS server
or other attackers. However, five criteria must be met to successfully implement the proposed method. First, the map on the LBS server must be divided into a cellular structure with a cell side length \( R \) (Figure 2(a)), and the user’s query range must be an inscribed circle of the cellular structure (Figure 2(b)), where the query radius is the POI within the range of \(( \sqrt{3}/2 \) \( R \). Second, the LBS server cannot frequently revise the cellular structure of the map. Third, the anonymizer must be reliable for masking user locations. Fourth, the maps on the anonymizer must contain intersections, length of road sections, and speed-limit information. Fifth, the algorithm must be available to the public.

Threat models for the LBS server and general attackers are defined in this section. The effectiveness of the proposed method for guarding against these threat models is discussed in Section 4. First, the LBS servers and general attackers can continuously tap, collect, and leak user information. However, they do not alter inbound or outbound query information (e.g., query number). Second, attackers use the open obfuscation algorithm and their background knowledge on known intersections, road sections, and traffic speed limits to deduce users’ travel routes and determine their locations. Third, query results are returned from the LBS server to the anonymizer. This creates the opportunity for the LBS server or general attackers to analyze the cache data of the anonymizer by using known cache algorithms. Fourth, the LBS server and general attackers can cross-reference the obfuscation areas queried by different user IDs.
in different locations and at different times to identify the associations between different queries and determine the query information submitted by the same user.

Without changing the center and side lengths of the cells, the initialization of the anonymizer and LBS server needs to be performed only once (procedures are presented in Section 2.1). We developed a three-phase unchained location privacy protection method for processing user queries (procedures are presented in Section 3). The following section provides the initialization model. Notations lists and explains the notations used in this paper.

2.1. System Initiation. Once the anonymizer obtains the center coordinates \((H_x^{(0,0)}, H_y^{(0,0)})\) from the LBS server, it uses these coordinates to number each cell and determine their center points. The term \((H_x^{(i,j)}, H_y^{(i,j)})\) denotes the center coordinates of each cell, where \((i, j)\) represents the \(x\)- and \(y\)-axes of the cell. The cellular-structure map illustrated in Figure 2(a) is used to generate a cellular structure comprising cells with \(1 + 3n(n + 1)\) side lengths = \(R\), where \(n\) represents the number of layers in the structure. The cells are numbered according to the \((i, j)\) order, where the center of the map is \((0, 0)\). Assuming that the hexagonal cell has six directions, \(i\) increases in increments of 1 to the right and decreases in increments of 1 to the left; \(j\) increases in increments of 1 to the upper right and decreases in increments of 1 to the bottom left; \(i\) increases and \(j\) decreases in increments of 1 to the bottom right; and \(i\) decreases and \(j\) increases in increments of 1 to the upper left. Results are illustrated in Figure 2(a).

Once all the cells in the anonymizer are numbered, set \(V\) (all intersection in \(G = (V, E)\), which is a figure containing section length weights) and set \(E\) with all sections linking two intersections in \(G = (V, E)\) are matched to each cell.

\[
\begin{align*}
\{V_{(i,j)} \subseteq V & | V_{(i,j)} = V^1_{(i,j)} \cup V^2_{(i,j)} \cup \cdots \cup V^6_{(i,j)} \cup V^\theta_{(i,j)} \} \neq \emptyset, \\
\{E_{(i,j)} \subseteq E & | E_{(i,j)} = E^1_{(i,j)} \cup E^2_{(i,j)} \cup \cdots \cup E^6_{(i,j)} \cup E^\theta_{(i,j)} \} \neq \emptyset, \\
\end{align*}
\]

(1)

where \(V^\text{Tir}_{(i,j)}\) represents the intersections contained in triangle \(\text{Tir}\) in cell \((i, j)\) and \(E^\text{Tir}_{(i,j)}\) represents the sections with length weights contained in the triangle \(\text{Tir}\) in cell \((i, j)\).

The numbering method for triangle \(\text{Tir}\) is illustrated in Figure 3. \(\text{Tir} = 1, 2, 3, 4, 5, \text{and } 6\) refer to \(\Delta p^1 p^2 p^2, \Delta p^2 p^3 p^7, \Delta p^3 p^4 p^7, \Delta p^4 p^5 p^7, \Delta p^5 p^6 p^7, \text{and } \Delta p^6 p^7 p^7\), respectively. This method enables the fewest cells in the obfuscation area to be used to cover the query range. Details concerning the generation procedures and verification of the obfuscation areas are presented in Section 3.

3. Unchained Location Protection Scheme

When a user submits a query, he transmits his ID, \(\text{POI}^u(L_x, L_y)\), POI for the query, and the \(k\)-anonymity requirements to the anonymizer. The anonymizer applies the three-phase obfuscation algorithm (Figure 4) to obfuscate his location prior to sending the query to the LBS server. The server then returns the queried information to the anonymizer, which filters out nonuser information before returning the POI to the user. In Phase 1, the user’s real coordinates are used to calculate the cell number of the user location and the triangle \(\text{Tir}\) within the cell. If the cell number and POI information are already cached in the anonymizer, the algorithm skips to Phase 3. Otherwise, it continues to the next phase. In Phase 2, multiple obfuscation areas are generated according to the user’s privacy requirements. The obfuscation area that contains the query range is substituted with a pseudo-ID and a pseudoquery order before it is transmitted to the LBS server. The anonymizer then caches the information returned by the LBS server (including the user’s original query and generated pseudoquery). This information can then be used for similar queries in the future. Finally, the anonymizer uses the substituted user ID to retrieve the POI results. In Phase 3, the filtered query results are returned to the user.

Calculation of the cell numbers is explained in Section 3.1. generation of users’ obfuscation areas is described in Section 3.2. generation of multiple pseudodobfuscation areas to protect the privacy of multiple users simultaneously submitting queries and using the cache to achieve unchained location protection are presented in Section 3.3, and query submission is outlined in Section 3.4.

3.1. Calculating User Cell Number. Once the anonymizer receives the current location coordinates of the user \((\text{POI}^u(L_x, L_y))\), it applies (2) to calculate the vertical displacement of the user relative to the center coordinates of the map \((H_x^{(0,0)}, H_y^{(0,0)})\). The anonymizer then incorporates the vertical displacement into (3) to determine the cell number \((i, j)\) of the user. The calculation process is discussed as follows:

\[
\begin{align*}
\{L_y - H_y^{(0,0)} & \geq \frac{d}{3R} \Rightarrow j = 2 \left[ \frac{L_y - H_y^{(0,0)}}{3R} + d \right] + e, \text{ where } L_y \geq H_y^{(0,0)}, \\
\{L_y - H_y^{(0,0)} & < \frac{d}{3R} \Rightarrow j = -2 \left[ \frac{L_y - H_y^{(0,0)}}{3R} + d \right] - e, \text{ where } L_y < H_y^{(0,0)} \}
\end{align*}
\]

(2)
First, the vertical distance between the user \((L_y)\) and origin \((H_y^{(0,0)})\) is used to calculate \(j\) of the cell number for the user location. The distance between the center points of two vertically adjacent cells (e.g., \((0,0)\) and \((-1,2)\) in Figure 5(a)) is \(3R\), and the cell numbers of these two cells differ by 2. When \(L_y\) is located above the center coordinates \((L_y \geq H_y^{(0,0)})\), \((L_x, H_y^{(0,0)})\) on \(y = H_y^{(0,0)}\) is added to \(L_y - H_y^{(0,0)}\). This point extends downward vertically to the cell boundary for distance \(d\). The number of cells within \(d\) is calculated and incorporated into (3) to determine whether the final section smaller than \(3R\) crosses over to another cell. In Example 1 (Figure 5(b)), the user is located on a random point \((P^o)\) of \((L_x, L_y)\) above the center coordinates. \(|L_y - H_y^{(0,0)}| + d\) can be expressed as 

\[
\text{PPFC} = P^o P^c + P^c P^s = 3R + P^c P^d\]  

and \(j = 2 + e\) (see (2)). Moreover, \(e = 0\) because \(P^c P^s < 2d\) (see (3)). This way, the user in Example 1 is located in \(j = 2\) on the \(y\)-axis. In Example 3 (Figure 5(c)), \(j = 0 + e\) because \(P^c P^s < 3R\) (see (2)). However, \(e = 1\) because \(P^c P^s > 2d\) (see (3)). Thus, the user in Example 2 is located in \(j = 1\). The cell number \(j\) of users can be determined using (2) and (3).

The preceding discussion indicates that \(d\) must be determined before \(j\) is calculated. The length of \(d\) is associated with a user’s \(L_x\) coordinate. The value of \(d\) gradually increases as \(L_x\) shifts from the cell boundary \((d = R/2)\) to the center of the cell \((d = R)\). Figure 6 shows that the center distance between two cells is \(2R \ast \cos 30^\circ = \sqrt{3}R\). In other words, one length cycle of \(d\) is \(\sqrt{3}R\).

\[
d = f(L_x) = \frac{1}{\sqrt{3}}\left|\alpha(L_x) - \frac{\sqrt{3}}{2}R\right| + \frac{R}{2} \tag{4}
\]

\[
\alpha(L_x) = |L_x - H_x^{(0,0)}| \mod \sqrt{3}R \tag{5}
\]

The term \(d\) can be expressed as a triangular wave by using \(R \leq d \leq R/2\) and a length cycle of \(\sqrt{3}R\). The wave equation is expressed in (4), where \(1/\sqrt{3}\) is the positive slope of \(0 \leq x \leq \sqrt{3}R\) and \(R/2\) is the vertical displacement distance. Because each \(\sqrt{3}R\) represents one cycle (see (5)), the relationship diagram between \(L_x\) and \(d\) can be illustrated by calculating the location of \(L_x\) in a single cycle, as shown in Figure 7.

Coordinates on the \(y\)-axis increase by \((\sqrt{3}/2)R\) when 1 is added to the \(y\)-axis. Therefore, the offset \((j \ast (\sqrt{3}/2)R)\) caused by the change in the \(Y\)-coordinates must be subtracted when calculating the \(X\)-coordinates. Then, the current distance between \(L_x\) and the origin on the \(x\)-axis is multiplied by a unit length to obtain \(i\) on the \(x\)-axis, as illustrated in Figure 5(b). We know that \((L_x, L_y)\) is located in \((i, 2)\), \(j = 2\), and the distance between \(H_y^{(0,0)}\) and \(L_x\) is less than \((\sqrt{3}/2)R\). Hence, a user cell number of \(i = -1\) can be calculated using (3).

Therefore, when the location of the user is known \((L_x, L_y), (H_x^{(0,0)}, H_y^{(0,0)})\) and (2) and (3) can be used to determine the current location of the user, that is, \((-1, 2)\) in Figure 5(b).

### 3.2. Determining the Obfuscation Area.

Once the cell number containing the user location and center coordinates is confirmed and if the user query information has not been cached, the anonymizer produces an obfuscation area for the user query and transmits the obfuscation area to the LBS server. The triangle \(\text{Tir}\) of the cell in which the user is located must be determined to obtain the number of cells required to encompass the query range and produce the minimum obfuscation areas. First, three straight lines passing through a random cell in the cellular-structure map are conceptualized (the three red lines in the cell illustrated in Figure 3). The three lines divide the cell into six equilateral triangles. Without loss of generalizability, the linear equations of the three straight lines intersecting the cell containing the current location of the user \((i, j)\) can be used to determine the equilateral triangle with the user (Figure 8):

- Linear equation of \(P_1P_s\), \(f_1(x, y): x - H_x^{(i,j)} = 0\)
- Linear equation of \(P_2P_s\), \(f_2(x, y): (x-H_x^{(i,j)}) - \sqrt{3}(y-H_y^{(i,j)}) = 0\)
Linear equation of $\overline{P_3P_6}$, $f_2(x, y)$: $(x - H_x^{(i,j)}) + \sqrt{3}(y - H_y^{(i,j)}) = 0$

For example, when $f_0(L_x, L_y) \geq 0$, $f_1(L_x, L_y) < 0$, and $f_2(L_x, L_y) > 0$, $f_0(L_x, L_y)$ represent that the user is either on or to the right of $\overline{P_0P_3}$; $f_1(L_x, L_y)$ indicates that the user is on $\overline{P_2P_5}$; and $f_2(L_x, L_y)$ means that the user is on $\overline{P_3P_6}$. A combined analysis of the three lines shows that the user is in $\Delta P_0P_3P_6$ of $\text{Tir} = 1$ (Figure 8(a)). Thus, four cells are selected as the obfuscation area for the user's query. These cells are numbered $(i, j)$, $(i, j+1)$, $(i+1, j)$, and $(i-1, j+1)$. If the user's location is at the center of the cell $(L_x = H_x^{(i,j)})$ and $L_y = H_y^{(i,j)}$, a random equilateral triangle can be represented the user’s query location. In Algorithm 1, $QH_p^t$ represents the obfuscation area of query $t$ in $p$, and $p = 1$ is the obfuscation area of the location provided by the user.
Subsequently, whether the four cells encompass the user’s query range must be determined. In Algorithm 1, three cells neighboring a random triangular section of \((i, j)\) in Figure 8 are selected to form four cells. Because of the similarities among the three triangles, a random location in the upper-right triangular section of \((i, j)\) is selected, without loss of generalizability, to verify that the combined area of the four shaded cells is the minimum obfuscation area to encompass the user’s query range (Figure 9).

**Supporting Theorem 1.** If the query range center is \(P^q\), the radius of the query range is \((\sqrt{3}/2)R\). The user is in a random location in a triangular section with points \(P^1 = (H_x^{(i,j)}, H_y^{(i,j)} + R)\), \(P^2 = (H_x^{(i,j)} + (\sqrt{3}/2)R, H_y^{(i,j)} + (1/2)R)\), and \(P^3 = (H_x^{(i,j)} + (1/2)R, H_y^{(i,j)} + R)\) and a side length \(R\) (Figure 9). Subsequently, the query range must include the cell with the triangular section and three neighboring cells to create an OA comprising four cells, specifically \((H_x^{(i,j)}, H_y^{(i,j)} + R)\), \((H_x^{(i+1,j)}, H_y^{(i+1,j)} + R)\), \((H_x^{(i-1,j+1)}, H_y^{(i-1,j+1)} + R)\), and \((H_x^{(i,j+1)}, H_y^{(i,j+1)} + R)\).

Proof. Assume that \(P^q\) is in a random location in \(\Delta P^3P^4P^5\). Subsequently, only three cells are required to encompass the user’s query range. From Figure 9, the center point of the \(P^q\) on \(P^3P^4\) can be identified. With only three cells, the diameter of the query range must be less than or equivalent to \(R\) of \(P^3P^4\) (query diameter \(\leq R\)). In actuality, the query diameter is greater than \(R\) (\(\sqrt{3}R > R\)). Thus, this hypothesis is contrary to fact, verifying that at least four cells are required for the obfuscation area to encompass the users query range. 

**Theorem 1.** If the user appears in a random location on the map, his or her query range is a circle with a radius of \((\sqrt{3}/2)R\). The proposed algorithm can use the lowest number of cells to encompass the user’s query range. The algorithm can maintain one-third of the size of the obfuscation area when the obfuscation algorithm is known to the attacker.

Proof. Supporting Theorem 1 indicates that an obfuscation area comprising four cells could sufficiently encompass the user’s query range when the user is located in a random location in \(\Delta P^1P^2P^3\). Supporting Theorem 2 indicates that at least four obfuscated cells are required in order to sufficiently encompass the user’s query range when the user is located in a random location in \(\Delta P^1P^2P^3\). Naturally, the user must be in \(\Delta P^1P^2P^3\) or \(\Delta P^1P^2P^4\) for the LBS server to produce the shaded obfuscation areas with the algorithm (Figure 9). The combined area of the two triangles is guaranteed to be one-third of the obfuscation areas.

The user is located within a cell comprising six equilateral triangles. Therefore, the location of the user in \(\Delta P^1P^2P^3\) is verified regardless of which triangle the user is located in.
This suggests that if the user is located anywhere on the map, the proposed obfuscation algorithm produces an obfuscation area of at least four cells, which is the lowest number of cells required, and guarantees that the area of the cells is at least one-third of the obfuscation areas.

3.3. Producing the Obfuscation Areas of Multiple Pseudo-queries. Section 3.2 describes how an obfuscated query area is produced to prevent attackers from obtaining the locations of users in sensitive areas such as special clinics or gyms. Users largely assume that attackers have a 1 in k chance of intercepting submitted queries (k-anonymity). Thus, we developed an algorithm that can produce multiple pseudo-queries to satisfy users’ k-anonymity settings. To enhance the relevance of the pseudoqueries and reduce the number of obfuscation areas, we developed an algorithm that produces multiple pseudoqueries in batches so that individual obfuscation areas and queries can serve as pseudoqueries for other users. Finally, the algorithm replenishes inadequate queries while satisfying individual privacy requirements.

When the anonymizer receives t privacy requests \((k_1', k_2', \ldots, k_u')\) in T from u users in different locations \((CN_1', CN_2', \ldots, CN_u')\) and no matches are cached, these queries must be transmitted to the LBS server. The anonymizer uses the proposed algorithm (Algorithm 1) to generate different obfuscation areas for the users \((QH_1', QH_2', \ldots, QH_u')\). If the users collectively form an OA, then \(QS' = \{QH_1', QH_2', \ldots, QH_u'\}\) can satisfy the maximum privacy requirement of the users.

\[
k_{\text{MAX}} = \max (k_1', k_2', \ldots, k_u').
\]
must contain four times as many cells than the number of cells required for the maximum privacy requirement.

\[ |QH_1^i \cup QH_2^i \cup \ldots \cup QH_n^i| \geq 4 \ast k_{\text{MAX}}^i. \quad (7) \]

The privacy requirements of \( u \) users can be satisfied by combining the obfuscation areas of their query locations. However, the number of obfuscation areas must be generated when too few users are available or when users are close together (blue area in Figure 10). For example, Users A, B, and C in Figure 10 request a privacy strength of only 2. Therefore, \( k_1^i = k_2^i = k_3^i = 2 \). However, the three users area within the same obfuscation area generated by the anonymizer, causing \( |QH_1^i \cup QH_2^i \cup QH_3^i| = 4 \). In this instance, a pseudoobfuscation area consisting of four cells must be generated (Figure 10) to meet the obfuscated cell requirement of \( 4 \ast k_{\text{MAX}}^i = 8 \).

To generate a pseudoobfuscation area that meets the privacy requirements, we developed a method for producing multiuser pseudoobfuscation areas (Algorithm 2). The method follows three criteria to repeatedly produce obfuscation areas until the obfuscation requirement of \( k_{\text{MAX}}^i \) is met:

1. Avoid VBAs [3] in the center location of the pseudoobfuscation area generated for the user’s location.
2. Avoid generating pseudoobfuscation areas already cached in the anonymizer. Based on the open obfuscation area generation algorithm, attackers know that the queries transmitted to the LBS server are not cached in the anonymizer. Subsequently, the LBS server deduces the cache data of the anonymizer by using the open cache algorithm. Therefore, the pseudoqueries that are detected as cached queries by the LBS server are filtered out.
3. Create new pseudocell numbers (\( CN_{\text{dummy}}^i \)) three layers from the user cell (\( CN_{\text{sel}}^i \)) to avoid generating an obfuscation area that overlaps \( CN_{\text{dummy}}^i \) and \( CN_{\text{sel}}^i \) and reinforce obfuscation strength more rapidly. Therefore, the anonymizer randomly selects one out of six cells three layers away from \( CN_{\text{sel}}^i \), namely, \( (i_{\text{sel}} + 2, j_{\text{sel}} + 1), (i_{\text{sel}} + 3, j_{\text{sel}} - 2), (i_{\text{sel}} + 1, j_{\text{sel}} - 3), (i_{\text{sel}} - 2, j_{\text{sel}} - 1), (i_{\text{sel}} - 3, j_{\text{sel}} + 2), \) and \( (i_{\text{sel}} - 1, j_{\text{sel}} + 3) \), as \( CN_{\text{dummy}}^i \) to generate the obfuscation area.

A cell is randomly selected from \( CN_i^i \) to serve as the center point (Row (2)) to meet Criterion 1. Then, a pseudocell number \( CN_{\text{dummy}}^i \) (Row (3)) is randomly selected from the cells surrounding \( CN_{\text{sel}}^i \) to meet Criterion 2. Subsequently, \( CN_{\text{dummy}}^i \) must contain an intersection. The pseudocell number \( CN_{\text{dummy}}^i \) is added to \( CN_i^i \) so that \( CN_i^i = CN_i^i + CN_{\text{dummy}}^i \) (Row (5)), and the pseudoobfuscation area generated using \( CN_{\text{dummy}}^i \) is added to \( QS_i^i \) (Rows (6) to (9)). This process is repeated until \( QH_{\text{MAX}}^i \) is generated to meet the obfuscation requirement of \( k_{\text{MAX}}^i \). Finally, an obfuscation area set is produced.

\[
QS_i^i = \{ QH_{1}^i, QH_{2}^i, \ldots, QH_{q}^i \mid (i, j) \in QH_{a}^i, |V_{i,j}| \neq 0, 1 \leq a \leq q \}. \quad (8)
\]

In Figure 11, Users A, B, and C move along the red line and transmit queries at the red points at different times. The blue, yellow, and red areas represent the three obfuscation areas generated by the anonymizer for the users’ queries transmitted to the LBS server. The anonymizer receives the privacy requirements of Users A and B, which are \( k_A^1 = 1 \) and \( k_B^1 = 4 \). However, \( QH_A^1 \) and \( QH_B^1 \) generated for the locations of Users A and B overlap, creating an OA with only seven cells. This fails to meet User B’s privacy requirement of 2, which requires eight cells (\( 4 \ast k_B^1 = 8 \)). The anonymizer selects a random cell (\( CN_{B}^1 = (1, 1) \)) three layers away from \( CN_{A}^1 \) and...
The preceding obfuscation method has two problems. First, the anonymizer can immediately respond to the user without accessing the LBS server when a similar query is cached. Existing methods aimed at enhancing the cache hit ratio [25–27] effectively reduce the likelihood of exposing queries to the LBS server while conserving the communication cost and computation load of the anonymizer. For example, the proposed method uses a hierarchical clustering method [28–31] to group the cached queries according to popularity. These groups are then used to generate corresponding pseudoqueries to prevent attacks that exploit an uneven query distribution [32].

Because the query of User B is already cached in the anonymizer, it generates an obfuscation area only for User A in order to satisfy the two obfuscation requirements (red area in Figure 11).

The preceding obfuscation method has two problems. First, the anonymizer can immediately respond to the user without accessing the LBS server when a similar query is cached. Existing methods aimed at enhancing the cache hit ratio [25–27] effectively reduce the likelihood of exposing queries to the LBS server while conserving the communication cost and computation load of the anonymizer. For example, the proposed method uses a hierarchical clustering method [28–31] to group the cached queries according to popularity. These groups are then used to generate corresponding pseudoqueries to prevent attacks that exploit an uneven query distribution [32].

Because the query of User B is already cached in the anonymizer, it generates an obfuscation area only for User A in order to satisfy the two obfuscation requirements (red area in Figure 11).

The preceding obfuscation method has two problems. First, the anonymizer can immediately respond to the user without accessing the LBS server when a similar query is cached. Existing methods aimed at enhancing the cache hit ratio [25–27] effectively reduce the likelihood of exposing queries to the LBS server while conserving the communication cost and computation load of the anonymizer. For example, the proposed method uses a hierarchical clustering method [28–31] to group the cached queries according to popularity. These groups are then used to generate corresponding pseudoqueries to prevent attacks that exploit an uneven query distribution [32].

Because the query of User B is already cached in the anonymizer, it generates an obfuscation area only for User A in order to satisfy the two obfuscation requirements (red area in Figure 11).

The preceding obfuscation method has two problems. First, the anonymizer can immediately respond to the user without accessing the LBS server when a similar query is cached. Existing methods aimed at enhancing the cache hit ratio [25–27] effectively reduce the likelihood of exposing queries to the LBS server while conserving the communication cost and computation load of the anonymizer. For example, the proposed method uses a hierarchical clustering method [28–31] to group the cached queries according to popularity. These groups are then used to generate corresponding pseudoqueries to prevent attacks that exploit an uneven query distribution [32].

The preceding obfuscation method has two problems. First, the anonymizer can immediately respond to the user without accessing the LBS server when a similar query is cached. Existing methods aimed at enhancing the cache hit ratio [25–27] effectively reduce the likelihood of exposing queries to the LBS server while conserving the communication cost and computation load of the anonymizer. For example, the proposed method uses a hierarchical clustering method [28–31] to group the cached queries according to popularity. These groups are then used to generate corresponding pseudoqueries to prevent attacks that exploit an uneven query distribution [32].
Second, when the anonymizer transmits user IDs to the LBS server, attackers can determine users’ travel routes by analyzing the queries of similar IDs, even when the location of the user is obfuscated. The following section proposes a method for generating unrepeated random pseudouser IDs for each \( QH_i \).

### 3.4. Generating Obfuscated Query Information

We developed a method to prevent LBS servers from combining obfuscation areas and user IDs to deduce users’ travel routes. Even when a simple algorithm is applied to substitute different user IDs with the same ID, LBS servers can still combine intersection and traffic speed-limit data to deduce users’ travel range and travel routes [33–36]. To prevent this problem, when the anonymizer generates \( q \) obfuscation areas for \( u \) user queries \( \{QH_{i1}, QH_{i2}, \ldots, QH_{i3}\} \) to satisfy their user privacy requirements, it randomly produces the \( q \) pseudo-IDs:

\[
\text{VID}^q = \{\text{VID}^q_i, \text{VID}^q_2, \ldots, \text{VID}^q_{q} | q \in \mathbb{N}^+\},
\]

\[
\forall \text{VID}^q_i, \text{VID}^q_j \in \text{VID}^q, \text{VID}^q_i \neq \text{VID}^q_j.
\]

Then, the anonymizer combines all obfuscation areas and the corresponding POIs to generate

\[
\text{LocPoi}^q = \{QH^q_{i1} \parallel \text{POI}^q_i, QH^q_{i2} \parallel \text{POI}^q_i, \ldots, QH^q_{im} \parallel \text{POI}^q_j\\}.
\]

Changes are logged with the anonymizer and used to filter user query results once they are returned by the LBS server. Finally, \( \text{VID}^q \) and \( \text{LocPoi}^q \) are combined to transmit the protected query to the LBS server:

\[
\text{Query}^q = \{\text{VID}^q, \text{LocPoi}^q\}
\]

### 4. Analysis

This section analyzes the security and performance of the proposed method and compares the results with those of previous studies. In Section 4.1, we present the security analysis items and compare past security problems. In Section 4.2, the method is applied to a map to examine the method’s real-time performance.

#### 4.1. Security Analysis

The unchained location privacy protection method developed in the present study was based on a trusted anonymizer and existing user/anonymizer security architectures to protect information confidentiality. Therefore, this section discusses four threat models derived from attacks that occur during the communication between the trusted LBS server and anonymizer. The results verify that the proposed method can effectively guard against most LBS attacks when the algorithm is known to the attacker.

When attackers possess the background knowledge of the maps and the capacity to continuously monitor user query content, they can issue the following attacks on user privacy:

- **Location Homogeneity Attack (LHA)**. Attackers collect queries from a particularly sensitive area to collect user information, such as a hospital specializing in cardiology and heart surgery, to gain information on heart patients.

- **Map Matching (MM)**. Attackers use background knowledge to filter out unlikely query source locations (e.g., lakes) to enhance the likelihood of identifying the actual locations of users.

- **Known Algorithm Attack (KAA)**. Attackers who are aware of the obfuscation algorithm can use the algorithm to calculate the obfuscation areas generated in different locations and filter out the less likely results to reduce the obfuscation strength of user locations.

- **Distance VBA**. Attackers calculate the center points of obfuscation areas to estimate the actual locations of users [3].

When LBS servers and general attackers use known location obfuscation algorithms to analyze the queries submitted by multiple users in obfuscated locations, they can perform the following attacks on user privacy:

- **Known Algorithm Attack (KAA)**. Attackers who are aware of the obfuscation algorithm can use the algorithm to calculate the obfuscation areas generated in different locations and filter out the less likely results to reduce the obfuscation strength of user locations.

- **Distance VBA**. Attackers calculate the center points of obfuscation areas to estimate the actual locations of users [3].
that the user cannot reach to reduce the obfuscation areas of continuous queries.

**Multiple Query Attack (MQA).** Attackers cross-reference the members and movement of users in different obfuscation areas to filter out pseudusers and identify real users.

The results in Table 1 show that the proposed method effectively guards against all known attacks. The symbol “O” denotes that the method can defend against this type of attack, and the symbol “X” denotes that the method fails to defend against this type of attack. In [3, 21], methods were proposed to obfuscate the locations of numerous querying users. However, these methods failed to consider user locations that approximate sensitive areas, which enables attackers to exploit these areas by using LHAs to obtain user locations. In [3, 22], algorithms were developed to obfuscate multiquery submissions. However, these methods could not continuously obfuscate locations when the user is moving, which enables attackers to observe the route of the users by performing MQAs. In [20], a method was proposed to substitute sensitive query locations with nearby insensitive locations cached in the anonymizer. However, this method failed to consider user movement speeds, enabling attackers to filter user locations by performing MMBs. Moreover, [20] used the center location of users to generate obfuscation areas, enabling attackers to estimate the actual location of users by performing VBAs [14, 21, 23–25]. Attackers could also confirm the center location of users in an obfuscation area once the algorithm is known to the attacker. In [22], a method was proposed for generating road network obfuscation areas by searching neighboring intersections to avoid placing users on the same road. However, systematically searching neighboring intersections enables attackers to perform KAs to map the obfuscation method and identify user locations.

### Table 1: Security comparison chart for multi-LBS queries.

<table>
<thead>
<tr>
<th>Protocols</th>
<th>LHA</th>
<th>MM</th>
<th>MQA</th>
<th>MMB</th>
<th>VBA</th>
<th>KA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xu and Cai [21]</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Xu and Cai [14]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Shankar et al. [15]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Wang and Liu [22]</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Ardagna et al. [20]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Lee et al. [23]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Song et al. [24]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Niu et al. [25]</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Our scheme</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

4.2. **Performance Analysis.** We implemented simulations in Java 8 on a computer equipped with an Intel i5-4570 CPU to create a test environment with a road map of Oldenburg, Germany [37]. Figure 12 shows that the anonymizer expanded the side length of the map from 10 to 40 km while generating cells to satisfy $R$. Notably, reducing the $R$-value increased the number of cells generated on maps with similar side lengths, reducing the content of each cell. The proposed method uses the same number of cells to obfuscate user query range. Therefore, lower $R$-values reduce the user query range and decrease the amount of data required to return query results from the LBS server.

We observed intersection conditions by dividing the Oldenburg map into $R$-sized cells (Figure 13). Results showed that smaller cells contained fewer intersections. Although Figure 12 shows that cells with shorter sides reduce the transmission load, the results in Figure 13 indicate that smaller cells reduce the number of intersections per cell. Fewer intersections increase the likelihood of attackers estimating the actual location of users. Therefore, a balance between transmission efficiency and the privacy strength must be achieved.

In Figure 14, the privacy requirement of each user is assumed to be $k = 5$ and $R = 1200$ to compare the required average number of queries transmitted to the LBS server. Compared with the result of [25] regarding the number of queries submitted by a single user to generate an obfuscation
area, our cache hit ratio was 0, indicating that, without using the cache, four users or more are required to simultaneously transmit a query to meet the privacy requirements with a reduced number of pseudoqueries sent by the anonymizer to the LBS server. The proposed method can combine the user queries of similar obfuscation areas to meet various privacy requirements. In [25], a cache was used to reduce computation and transmission loads. In the present study, we adopted a cache hit ratio of 70%, similar to that used in [25]. Regardless of the number of users, we maintained the privacy protection strength equivalent to that reported in [25], and the performance of the proposed method improved as the number of users was increased. In our proposed method, the number of obfuscation areas must be generated when too few users are available or when users are close together. When the number of users is 2, only 3 queries are submitted to the LBS in [25], and our method requires 6.185 queries with hit rate = 0% or 3.32 queries with hit rate = 70%. But in our method, the obfuscation areas of the query locations can be combined when the number of users increases, which reduce the number of queries that needs to be sent to the LBS. In Figure 14, when the number of users = 8, 12 queries is submitted to the LBS in [25], our method needs 8.019 queries with hit rate = 0% and only 6.427 queries with hit rate = 70%. In this situation, our performance is better than [25].

Figure 15 shows that the average road lengths in the Oldenburg map that satisfy the $k$-anonymity value when the radius $R = 1,200$. In [22], the roads were simply extended to obfuscate the location of users. Niu et al.’s method [3] uses a random walk-based cloaking algorithm, and the method proposed in the present study divides the map into cells. Therefore, the average road lengths of the overall obfuscated areas using the proposed method and [3] were markedly longer than that determined using the method proposed in [22]. Moreover, we generate extra queries to simulate a multiuser environment which requires generating additional obfuscation areas when the cells overlap. The proposed method generated 4.88% longer road length than [3] when $k = 10$.

5. Conclusion

We developed a privacy protection scheme to protect the real location suitable for moving users. The scheme produces multiuser pseudoqueries and uses obfuscation areas to prevent LBS servers from directly deducing users’ real queries and precise locations. We verified that the method produces obfuscation areas with the least number of cells and guarantees one-third the original obfuscation area size when the algorithm is disclosed. We also considered the distinct characteristic of user queries in different areas and adopted a grouping approach coupled with actual maps to reduce the likelihood of the pseudodata being filtered out by the LBS server, thereby satisfying users’ privacy requirements. Furthermore, we incorporated a caching system to store users’ continuous queries. The cache system coupled with multiuser queries prevents the LBS server from completing deducing users’ routes. Instead, the LBS server can generate only scattered and obfuscated user locations. Therefore, the proposed method effectively protects location privacy during continuous querying. The cache approach also reduces the likelihood of user locations being transmitted to the LBS server, decreases the computation and transmission loads of the anonymizer, and enhances system performance. The proposed method is fully compatible with various user devices. They can use their original mobile devices and Internet.
service providers to access the trusted anonymizer to protect their location details when submitting a query. Finally, we verified that the proposed method effectively protects users’ identities, locations, and interests and guards against most currently known attacks on location privacy. We also used a real-time road map to test the proposed method. Figure 14 shows that the proposed method uses a cache approach to greatly reduce the amount of query information exposed to the LBS server. A summary of the results illustrated in Figures 14 and 15 shows that the proposed method outperformed other existing methods.

**Notations**

- \( R \): Cell side length
- \( u \): Number of real users
- \( q \): Index value, \( q \in \mathbb{N}^* \)
- \( k^t_q \): \( k \)-anonymity requirement of the user \( q \) in query \( t \)
- \( k^t_{\text{MAX}} \): \( k \)-anonymity requirement for the multiuser query \( t \)
- \( ID \): User ID
- \( VIDS_q \): Pseudo-ID \( q \) randomly generated by the anonymizer
- \( ID(u, j) \): Cell number
- \( (i, j) \): Cell number \( q \)
- \( (H_x^{(i,j)}, H_y^{(i,j)}) \): \( X \)- and \( Y \)-coordinates of cell \( (i, j) \)
- \( P^t(L_x, L_y) \): User’s current location
- \( (L_x^u, L_y^u) \): The real location of the user \( u \) in a multiuser query
- \( CN^t_q \): Cell number of user \( q \) in a query \( t \)
- \( CN^t \): Cell number set during a query \( t \)
- \( CN^t = \{ CN^t_1, CN^t_2, ..., CN^t_n \} \)
- \( Tir \): Number of the equilateral triangles formed by the six vertices of the cell \( \{ \text{Tir} \in \mathbb{N}^* \mid 1 \leq \text{Tir} \leq 6 \} \)
- \( V^t_{(i,j)} \): Intersection set contained in triangle Tir in cell \( (i, j) \)
- \( V \): The set of all intersections \( \{ V_{(i,j)} \subseteq V \mid V_{(i,j)} = V_{(i,1)} \cup V_{(i,2)} \cup ... \cup V_{(i,6)}, |V_{(i,j)}| \neq 0 \} \)
- \( E^t_{(i,j)} \): Road section set of triangle Tir in cell \( (i, j) \)
- \( E \): All road section sets \( \{ E_{(i,j)} \subseteq E \mid E_{(i,j)} = E_{(i,1)} \cup E_{(i,2)} \cup ... \cup E_{(i,6)}, |E_{(i,j)}| \neq 0 \} \)
- \( d \): Vertical distance between \( L_y^u \) and the lower or upper boundary of the cell
- \( t \): Current query in a continuous query, \( t \in \mathbb{N}^* \)
- \( T \): Wait time of the anonymizer before receiving a query from a user
- \( e \): Indicator of additional space between cells, \( e = \{0, 1\} \)
- \( QH^t_q \): One obfuscation area comprising four cells, all cell numbers within the obfuscation area in query \( t \)
- \( QS^t \): OA set of query \( t \)

**Conflicts of Interest**

The authors declare that there are no conflicts of interest regarding the publication of this paper.

**Acknowledgments**

This research was supported by the National Science Council of Taiwan under Grants nos. MOST 106-2221-E-130-001, MOST 106-3114-E-011-003, and MOST 106-2221-E-033-002.

**References**


