Journal of Computer and Communications, 2015, 3, 7-20
Published Online May 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.35002

A User Proprietary Obfuscate System for Positions Sharing in Location-Aware Social Networks

Wei Cherng Cheng, Masayoshi Aritsugi

Computer Science and Electrical Engineering, Graduate School of Science and Technology, Kumamoto University, Kumamoto, Japan
Email: alan@dbms.cs.kumamoto-u.ac.jp, aritsugi@cs.kumamoto-u.ac.jp

Received February 2015

Abstract

A user's trajectory can be maliciously monitored by adversaries when they share the positions in location-aware social networking applications which require users to update their own locations continuously. An adversary infers user's locations from the trajectories, and gleans user's private information through them via location-aware social networking applications and public available geographic data. In this paper, we propose a user proprietary obfuscate system to suit situations for position sharing and location privacy preserving in location-aware social network. Users transform the public available geographic data into personal obfuscate region maps with pre-defined profile to prevent the location leaking in stationary status. Our obfuscation with size restricted regions method tunes user's transformed locations fitting into natural movement and prevents unreasonable snapshot locations been recorded in the trajectory.

Keywords

Location Privacy, Obfuscate Region Maps, Location-Aware, Positions Sharing

1. Introduction

The users' physical locations are widely adopted by applications installed in mobile devices as fundamental information to enhance the services. Social networking application is one of the popular mobile applications involved with location that Pew Research showed 28% of the mobile users utilize the application on a typical day basis [1]. Existing social network applications, such as Facebook and Twitter, only provide users to turn on or off of the location exposedness selection. Newly merged types of mobile social network applications are location-aware social networks application, such as Foursquare and Beetalk, and users have to share the updated locations to enable the application service. With the new type of social network applications, users are able to build up new social connections with strangers and friends, but then lose the permission control in location sharing. Malicious adversaries and stalkers in the same social network applications can track and record others' locations or movements with the mutual related distance provided by the application, and infer the physical locations or trajectory with public available geographic data. Today, people recognize visited locations as part of personal identification, since the locations reflect user's interests, habits, religious, political intention and other privacy. Thus, while using the newly merged type of social network applications which required users to share and update their own locations continuously, the location privacy preserving concerns are: "How to share a close enough location in the social network applications, and still preserve the location privacy from the others?"

The statement points out two keynotes for location privacy challenges. First, sharing and updating user's locations are needed in location-aware social network applications. Users have the needs to query and connect with nearby others. Existing location privacy preserving approaches such as cryptographic techniques [2] [3] encrypt the users' actual location before sharing, and only authorized friends can decrypt it. The approach suits in the check-in service but not location sharing in the location-aware social networking, since users attempt to connect nearby strangers. Second, all mobile device users are not always staying in stationary, and they could be in any locations. While they're moving with velocity, the movement shall sustain undisclosed all the time. Another location privacy preserving approach by Damiani *et al.* proposed a spatial obfuscating system [4], based on user's pre-defined sensitive semantic model, and user's concerned locations will not be disclosed when they move into the sensitive locations. The drawback of the approach is that when users move into undefined areas, snapshot locations are able to be inferred from recorded movement [5] [6].

The approaches above did not consider both keynotes of location privacy protection at the same time. They are locations sharing to all others and locations cloaking anywhere. Hence, we propose a system to transform wider space of actual geographic data into correlated regions map in advance, and use regions' coordinates to substitute for actual ones. It conquers the two keynotes privacy concerns when sharing positions in location-aware social network applications.

Our proposed system performs the obfuscate region map generation to provide the users' privacy preserved positions for sharing in location-aware social network applications. The generation progresses in two phases. First is based on user proprietary privacy profile to transform publically available 3rd party geographic data into undistinguished obfuscate region map in a separate server. The user proprietary privacy profile is the sensitivity preferences of semantic locations which can be adjusted by user flexibly. The output of obfuscated regions then satisfies the user's location privacy need, and arbitrary region coordinates suit into the situations for positions sharing anytime and anywhere in heterogeneous networks. Second, we take the scenario that users are in the movement into consideration, and constrain the output region scale to make the movement normal. Thus, an attacker in the same location-aware social network application cannot infer the user's location with the shared positions. The contribution of this work can be summarized as follows:

- The users defined proprietary privacy profiles obfuscate the publically available 3rd party geographic data into unextinguished region map to suit the position sharing anytime and anywhere in location-aware social network applications.
- Two phases of the obfuscate region map generation protect user's location privacy. First phase model preserves user concerned sensitive semantic locations and second phase model suits the moving users' positions sharing with smooth movement.
- The obfuscate algorithms with different scenarios are evaluated. The regions size presents the arbitrariness of the obfuscate result with the user's location privacy preserved. And the distance between the actual location and cloaked locations during movements demonstrates the approach is suitable for users to provide the positions sharing in location-aware social networks.

The rest of the paper is organized as follows. Section 2 overviews the related work. Section 3 proposes the system architecture and defines the attacking model. In Section 4 we represent different types of obfuscate approach against the position sharing. Section 5 experimentally evaluates our solution, and Section 6 concludes the paper and some future work.

2. Related Work

The lightweight design of mobile devices and speedy internet connectivity made the mobile social network applications outgrown the desktop ones [7]. Social networking applications executed by mobile devices can now be in anywhere, stationary, or in the move. Users store and transmit personal sensitive information, such as ID and the continuously updated location information [8] [9] to the applications, and the sensitive information is potentially disclosed. Thus, the location privacy and security risks in mobile devices with social network applications have become an issue. The location-aware social networking applications allow users to create their own social network by connecting with strangers and friends, but request users to provide locations to enable the application service, and it leads to users' location privacy preserving concerns.

Common approaches used in location privacy preserving are *k*-anonymity with spatial obfuscation and encryption [10]. As a *k*-anonymity approach, Mokbel *et al.* [11] notated a user as center and anonymized other *k*-1 users' locations in a region, or expanded the region size till the criteria of *k* users in the region fit, and thus all *k* users were undistinguished from others. The method did not apply to the specific area where the total users are not reaching *k* users or the regions become extra huge and fell out of searching range. Xue *et al.* [12] and Bamba *et al.* [13] enhanced the *k*-anonymity technique with taking *l* number of different semantic type locations into consideration when generating the cloak region. User's privacy protection is insufficient with the solutions when the adversaries are using the same geographic data. The spatial obfuscation approach devoted by Damiani *et al.* [4] [14] adopted personal concerns as profile to protect individuals' sensitive locations with Hilbert movement algorithm. The approach expanded the map-aware obfuscation area in certain semantic locations, but users' trajectories are still insecurely monitored while users move into the non-cloaked regions. In a cryptography-based technique, Wernke *et al.* [3] proposed to split-up user's obfuscated location into n shares and separately distributed them to different on-line processing servers before sharing them. The approach is costly with more involved on-line processing servers guarantee better security level and does not suit to cases when there are fewer location servers involved. Wei *et al.* [15] proposed to take telecommunication cellular towers as encryption servers to perform and store users' encrypted location data, and also generate fake users. Similarly, Wen *et al.* [2] improved the idea by adding an extra symmetric key between cellular tower and user in transmission. However, the assumption limits connections with cellular towers only but no other wireless protocols. Also, practically, a cellular tower did not perform as a functional server for specific mobile applications. Moreover, Lin *et al.* [16] used multiple transformations by categorizing users into different groups while moving, but it required all users to be authorized, which did not suit the situation when users initially intend to query strangers. The work in [17] employed the obfuscating approach with Hilbert curve movement to transform area map into region map before sending the coordinates for sharing and querying in social networking application. It reduces the risk of user being tracked by adversary all the time when utilizing the application under location snapshot. However, the random movement protection within large obfuscate regions in short timeframe is insufficient.

In this paper, we propose utilizing the obfuscate region map to act for user's physical location when users are in stationary status, and obfuscation based with restricted the region size solution to prevent the locations leaking during movement. It suits to different network protocols, and is flexible for users to adjust the sensitive locations while sharing. It also overcomes the location leaking problem that users and adversaries use the same geographic database for both stationary status and in the move.

3. Architecture and Attack Model

The proposed system generates region map with user's proprietary concerns about semantic locations. The region map's coordinates then act as user's physical positions to preserve user location privacy in location-aware social network applications. As the user queries others in the applications, obfuscated positions are disclosed to others, no matter users are moving or in the stationary status.

3.1. Service Model and Architecture

The application service model consists of four components, namely, mobile devices, obfuscate server, social networking service server (hereafter, we use SNS server as abbreviation), and 3rd party location-based service server (hereafter, we use LBS server as abbreviation). **Figure 1** illustrates the service model's architecture. While using location-aware social network applications with mobile device, users obtain the initial physical coordinates from satellite, Wi-Fi station or other cellular protocols infrastructure. Mobile devices are installed with location-aware social network applications and users may choose to connect to SNS server directly or through obfuscate server. SNS server is the application platform which provides the function for users to connect with each other. Without location obfuscating, adversaries and stalkers in the same application are able to track and glean target user's privacy through the visited locations. We assume the obfuscate server is the trusted location anonymity server selected by user between mobile devices and SNS server. According to user's initial location, and preference setting, it performs the region map generation with geographic data from public LBS

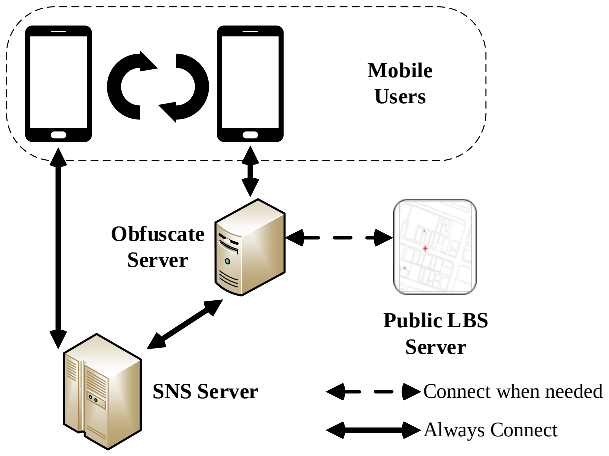


Figure 1. Service model architecture.

server. User's correlative transformed region coordinates will then be sent to SNS server for users' position sharing purpose. Location-aware social network applications allow user to query and connect to all surrounding nearby others with showing the Euclidean distance as pair wise distance.

The obfuscate server is the core component of the service in our proposal and it performs the following functions:

1) User profile maintain: Mobile users register the initial locations and specify semantic locations' sensitivity level preference to conduct the user profile.
2) Geographic data obtain and obfuscate region map process: User's initial location is needed for wider area's geographic data downloading. Location update from LBS server can be done periodically with longer interval or when user is almost falling out of the area. Based on the user profile and downloaded geographic data from LBS server, the obfuscate server then generates the user proprietary region map. The region map can be stored in the obfuscate server or in mobile devices.
3) Region map's coordinates substitution for position sharing: While running a location-aware social network application, mobile users share the region map's coordinates as their own positions with others in the same application.

We promote to generate the obfuscate region map in a separated obfuscate server instead of building the function in device because in-device software is restricted by disadvantages, such as extra CPU, memory, and storage space consumption, device battery drained off by frequent location update, and consequently forced to update due to mobile devices operation software upgrade. In addition, a separated server setup approach suits to heterogeneous network.

3.2. Location Privacy Attack Model

In our proposal, user's location privacy is preserved by obfuscate region. An obfuscate region's coordinates will act as user's physical location while using the location-aware social network applications. In **Figure 2(a)**, we assume that area geographic data is gridded into small cells, and obfuscate server transforms the area into two obfuscate regions OR_1 and OR_2. When the user is in stationary status or moving within the OR_1, the OR_1 coordinates act as the user's any positon within the region. In stalker's view, the user does not change the location, so the user is able to share the OR_1 coordinates to query and connect with nearby others without leaking the actual locations.

However, let us consider a situation that a user is moving continuously from position l_1 via l_2 to l_3 as shown in **Figure 2(b)**, and the stalker is close to user's position l_1. User's obfuscated regions movement report as l'_1 via l'_2 to l'_3 at time t_1, t_2, and t_3 to the stalker. Assume the user is moving with speed v and v is between average walking speed 5 km/h [18] and riding bicycle speed 10 km/h. If the reported distance from stalker point of view, Euclidean distance of $(l'_2 - l'_1)$ is much greater than the distance of $v \cdot (t_2 - t_1)$, then the stalker can infer either OR_1 or OR_2 is a fake location. Following, the Euclidean distance of $(l'_3 - l'_2)$ is greater than $v \cdot (t_3 - t_2)$, then the

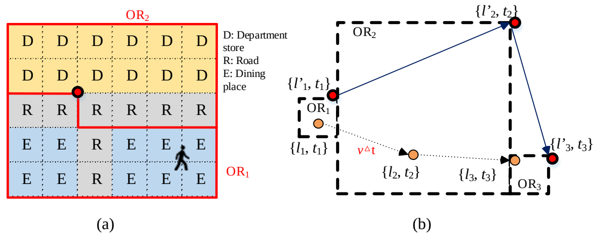


Figure 2. (a) Stationary user location in stalker's view; (b) User's movement in stalker's view.

stalker can confirm the obfuscate region OR_2 coordinate is not real. In this case, when the obfuscate region OR_1 size is small, user's actual location can be identified.

In this proposal, we assume the user is either in stationary status or moving with velocity, and the obfuscate region map will be applied to preserve user's location privacy from stalkers in the same application.

4. Obfuscate Region Generation

We adopt the strategy of transforming public available area geographic data into obfuscate region map before sharing the coordinates with SNS server. The approach allows us to process each individual's personal map based on user's profile setting which includes the sensitivity level in different semantic types' locations and corresponding threshold value. We first introduce the sensitivity model defined in the paper, and then detail three algorithms of the approach to aggregate the obfuscate regions from a reference space. A summary of notations used in this section is given in **Table 1**.

4.1. Sensitivity Model

The reference area is represented as a 2 dimensional bounded space, and we assume that it is fulfilled with different and disjoint spatial objects [19], such as road, hospital, restaurant, shop, parks, and varied constructions. We describe those spatial objects as different features types. Users provide the sensitivity concerns for major features types with values from 0 to 1 as their profiles. Sensitivity value 0 represents the feature type has no sensitive concerns at all, and value 1 represents as extremely sensitive to user. The threshold value is introduced as the comparison value. The feature type sensitivity value higher than threshold value implies that the corresponding locations need be cloaked. The reference space Ω is gridded and subdivided into regular and sufficiently small unit size cells, denoted as c and $\Omega = \{c_1, c_2, \ldots, c_n\}$ while two cells, c_i and $c_{i+1,}$ are adjacent cells sharing common border. In unit cell c, we define the feature density function D (ft, c) to quantitate the feature shares in the unit area. The feature density function is used to judge the extent of the unit cell. We define sensitivity level of feature types to be rated in value, and denote the sensitivity level as v (ft) where v (ft) has value between [0, 1], and the greater value indicates users demand higher sensitive concerns when they are in the feature. For example, unit cell c contains different features as a bank and a school in **Figure 3(a)**. The feature density of c with respect to bank and school is calculated, and denotes as D (*Bank*, c) and D (*School*, c). In **Figure 3(a)**, within unit cell c, bank density of cell c is greater than school density of cell c, *i.e.*, D (Bank, c) > D (*School*, c). We then take feature bank's sensitivity value v (*Bank*) as sensitivity of unit cell c extent, denoted as E (*Bank*, c), in **Figure 3(b)**. A region r is a combination of common border cells with different sensitivity value, where $c \in r$. We denote S (r) as the mean sensitivity value of a region to represent its sensitivity level.

Personal profile as our transformation variablesis proprietary to user and it makes the region map distinguish one from others. Personal profile defined as *profile*, which includes a list of major concerned feature types FTs = $\{ft_1, \ldots, ft_n\}$, for example restaurant, school, hospital and bank, with respective sensitivity value V= $\{v(ft_1), \ldots, v(ft_n)\}$. An example of user defined feature types and corresponding sensitivity values table is listed in **Table 2**.

A sensitivity threshold, T, needs to be decided by users as privatized basis to adjust the map. When feature type's sensitivity is greater than the threshold value, it implies user's actual location needs to be cloaked in the feature. The user profile takes the form of the tuples: $<FTs, V, T>$.

The approach of the obfuscation region map is implemented by aggregating the cells into regions from the comparison result with threshold value. The formed regions are the expansion of user concerned sensitive loca-

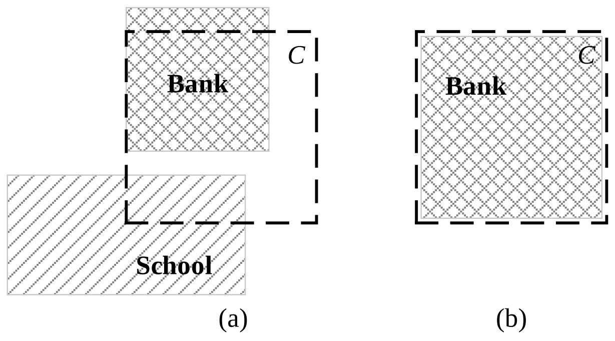


Figure 3. (a) Example of a bank and a school in the unit cell *c*; (b) Bank sensitivity represents the sensitivity of unit cell *c*.

Table 1. Summary of notations.

Notation	Definition
Ω	Reference space
Ω'	Reference space with Obfuscate Regions
ft	features
c	sufficient small unit cell
r	combination of border connected unit cells
$D(ft, c)$	feature density function in unit cell
$v(ft)$	Feature’s sensitivity level value
$E(ft, c)$	sensitivity of unit cell *c* extent with feature *ft*
$S(r)$	mean sensitivity of region *r*
T	Sensitivity threshold value defined by user
$\mathrm{x}_{\max}$	Maximum region size

Table 2. Feature types and sensitivity values.

Item	Feature	Sensitivity Value
1	Hospital, and care center	0.7
2	Bank, post office	0.6
3	School	0.55
4	Traffic station	0.5
5	Entertainment location, shopping stores	0.45
6	Accommodations	0.4
7	Department store and mall	0.35
8	Dining restaurant	0.3
9	Religious location	0.25
10	Public facility (parks, tourist spots)	0.2
11	Residence and unclassified building	0.1
12	Mountain or unclassified land	0.09
13	Others (such as road, sidewalk, bridge)	0.09
14	Unreachable location	1

tions and combination of non-sensitive regions. Theoretically, wider obfuscate region leads to less positions accuracy and better cloaked result. We experiment two aggregation methods in this paper. However, too wide region result is disadvantaged when users are in the moving among regions. Stalkers can track the unnatural movement to infer the user's actual location. Thus, in the third method, we restrict the each region's maximum size to x_{max} cells while generation with first method to prevent the long distance update in a short time period during movement. In the summary, the approach with three algorithms are listed as below:

1) Hilbert Movement: The algorithm expands cells to regions by comparing the sensitivity with threshold value following the Hilbert curve movement to create the arbitrary shape of regions. It suits to situations where users are in stationary status.
2) Pyramid Scan: The algorithm aggregates cells in the area as a region and divides the region into smaller sub regions by comparing the mean region sensitivity value with threshold to create the arbitrary shape of regions. It also suits in stationary status for positon sharing.
3) Hilbert Movement with Region Size Restriction: Based on the Hilbert movement algorithm, it constrains the region size no greater than x_{max} cells in generation. The output is suitable for users in the movement.

4.2. Hilbert Movement

In the first algorithm, we choose the Hilbert curve movement as the moving route to generate the region maps. In a 2 dimensional reference space gridded into unit cells, Hilbert curve movement will linearly run through all cells without repeating. In the algorithm, cells with sensitivity values are compared with the threshold value *T* following the Hilbert curve movement steps. The obfuscation region map is generated by Hilbert Movement processes as follows:

1) Based on user defined profile with features type and sensitivity values, load the reference space of $2^h \times 2^h$ gridded cells with sensitivity values into the data set.
2) Follow the Hilbert curve route steps, and compare the unit cell's sensitivity value with the threshold value. If the unit cell sensitivity is greater than the threshold, the merge status update to start, and form a sensitive region by marking the next cell with same region identity. Calculate the region's mean sensitivity value and compare it with threshold value again. The merge process continues until the region mean sensitivity value is less than the threshold value. Add the region to region map.
3) Cells with sensitivity values less than the threshold value and not merged to region will be marked with separate identity, and then pass to next cells.
4) Check and merge the cells in the reference space with separate mark and share the same border as regions. Add the regions to region map.

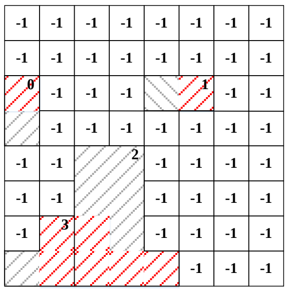
Figure 4(a) shows the pseudo code for the Hilbert movement to generate the obfuscate region map. **Figure 4(b)** shows an example that red mark cells are with sensitive value higher than threshold value, and initial result from processes 1 to 3 generated four sensitive regions with identity from 0 to 3, and non-sensitive cells marked identity −1. We remap those cells with valueless than threshold and shared border as a region in process 4. Final result in **Figure 4(c)** shows that there is no sensitive regions or non-sensitive regions difference in the region map.

4.3. Pyramid Scan

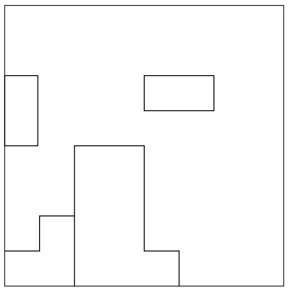
A level-based aggregation approach in Pyramid structure can also generate obfuscate region map for sharing purpose. Considering reference space is gridded into $2^h \times 2^h$ cells, where h indicates the level height of the space. In Pyramid structure, Level h represents bottom level with $2^h \times 2^h$ individual unit cells, and Level 0 indicates top level, where all $2^h \times 2^h$ cells are considered as one region. From the top level towards the bottom level, each region in a level of Pyramid structure is divided into 4 square sub-regions equally in the next level. In **Figure 5(a)**, we show an example of 8 × 8 grids space in Pyramid structure. Level height in the example is 3. Level 3 is the bottom level with $2^3 \times 2^3$ individual cells, and level 0 is the top level is one region with 4^3 cells in it. In level 1, four divided sub-regions are from level 0. Each of the region in level 1 contains 4^2 cells in it. Following the same division rule, we take each region in level 1 as parent, we process further sub-region division in level 2. In **Figure 5(b)**, we show partially the region's numbering after division. The approach of the Pyramid Scan processes is as follows:

```
Algorithm 1: Hilbert Movement
function Hilbert_Movement(grid, threshold)
        bStart = false;
        RegionMap = Ø
        for idx = 0 to maxHilbertIdx(grid) do
           i = cell[PosX, PosY];
           if (!bStart && (i >= threshold))              // Initial cell with sensitivity greater than
               bStart = true;                            // threshold, start a region
               iCount = 0;
               sum = 0;
               iRegionCount++;
           if (bStart)                                   // Calculate the mean sensitivity value of
               iCount++;                                 // the region
               sum += i;
               average = sum / iCount;
               if (average < threshold)                  // If the mean value is less than threshold,
                   bStart = false;                       // the region ends
                   average = 0;
           if (bStart)                                   // New region start with new region id
               cell[PosX, PosY].iRegion = (iRegionCount - 1);
               cell[PosX, PosY].fSum = sum;
               RegionMap = cell[PosX, PosY].iRegion  // Add the formed region into Region Map
           else
               cell[PosX, PosY].iRegion = -1;            // Mark cells with value less than threshold
               cell[PosX, PosY].fSum = 0;                // as region id " -1 "
        end for
        Remap(cell)                                      // Combine the region id = -1 and with
        return RegionMap                                 // shared border as one region
end function
```

(a)

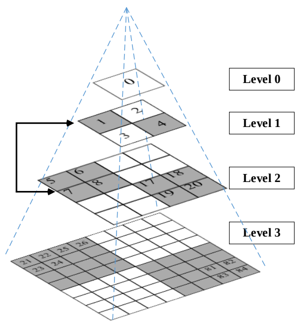


(b)



(c)

Figure 4. (a) Pseudo code for Hilbert Movement; (b) Result of sensitive regions generated; (c) Final obfuscate regions.



(a)

Level	Region ID	Parent Region ID	Contains Cell Amount
0	0		8*8
1	1	0	4*4
1	2	0	4*4
1	3	0	4*4
1	4	0	4*4
2	5	1	2*2
2	6	1	2*2
2	7	1	2*2
2	8	1	2*2
2	9	2	2*2
2	10	2	2*2
2	11	2	2*2
2	12	2	2*2

(b)

Figure 5. (a) Pyramid structure in 8 × 8 grid; (b) Sub region break and region's re-numbering.

1) Load the $2^h \times 2^h$ gridded cells with user sensitivity values into the data set. Follow the division rule from Level 0 towards Level ($h - 1$) that one region as parent in level i is divided into 4 square sub son_regionsas in level $i + 1$. Number all regions with identity and calculate the regions' mean sensitivity value.
2) Compare the region mean sensitivity with threshold value. In the same level, if the region mean sensitivity value is greater than the threshold, it is merged with neighbor regions to form a new region. Add the region to region map.
3) Regions with mean sensitivity value less than threshold and not merged with neighbors move to the next level for threshold comparison and follow merge process in process 2 until the bottom level.

4) Scan the cells in the bottom level, cells without merge and share the same border combined as regions. Add the regions to region map.

Figure 6(a) shows the pseudo code of Pyramid scan to generate obfuscate regions. In **Figure 6(b)** shows an initial result with Pyramid scan algorithm processes 1 to 3 from a grid space 8 × 8 cells with same sensitivity conditions in red mark as **Figure 4(b)**. Four sensitivity regions are generated. We then remap those cells with sensitivity valueless than threshold and shared border as a region in process 4. A region map without distinguished between non-sensitive and sensitive regions by pyramid scan algorithm is generated as shown in **Figure 6(c)**.

4.4. Hilbert Movement with Region Size Restriction

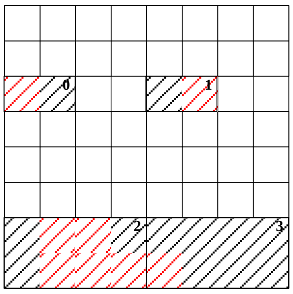
In stationary status, users' physical locations can be protected by region coordinates. However, in Section 3.2, we described a stalker might sense target user's cloaked locations to infer the physical ones when moving in or out extra-large regions. To prevent the problem, we constrain a region size to maximum x_{max} cells in the region during the generation.

Comparing the two algorithm results in the **Figure 4(c)** and **Figure 6(c)**, the Pyramid scan result is lack of variation and monotonous as a polygonal region map. In the third algorithm, we adopt the Hilbert movement as a based algorithm to constrain the region size during region generation. The summary of the algorithm processes describes as follows, and pseudo code is shown in **Figure 7**:

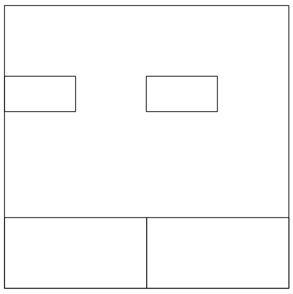
1) Load the $2^h \times 2^h$ gridded cells with sensitivity values into the data set. The bState judgment indicates to start a new region, and the iMax Cell Count is cell's amount included in the region. Follow Hilbert Curve route.
2) When the initial cell is with sensitivity value greater than the threshold, the State flag updates to indicate a

```
Algorithm 2: Pyramid  Scan
function Pyramid_Scan(level, threshold)
     iCount = (4^(Level + 1) - 1) / (4 - 1);
     for (i = 0; i < iCount; i++)
     new RegionAverage[iCount];
     Pyramid_Average(Level, 0, 0, 0);                          // Calculate region's mean sensitivity value in different level
     Pyramid_Region(Level, 0, threshold);                      // Compare the region mean sensitivity value with theshold
  function Pyramid_Average(int level, int iRegionIndex, int iX, int iY)
     if (level > 0)
        for (s = 1; s < 4; i++)
        Pyramid_Average(level - 1, ((iRegionIndex << 2) + s));   // Number and calculate four sub son_regions'
        RegionAverage[iRegionIndex].fAverage +=                  // mean sensitivity value in  level-1
              RegionAverage[(iRegionIndex << 2) + s].fAverage / 4; //Region's mean sensitivity value calculation in level
     else
        RegionAverage[iRegionIndex].fAverage =   Cell[iX, iY].fValue;
  function Pyramid_Region(int level, int iRegionIndex, threshold)
     if (level > 0)
        for (s = 1; s < 4; i++)                                // If sub son_region mean sensitivity value is greater than threshold
          if (RegionAverage[(iRegionIndex << 2) + s].fAverage >= threshold)
             f3 = Mean value of (RegionAverage[(iRegionIndex << 2) + s]) and 2 border connected neighbor regions;
             f2h = Mean value of (RegionAverage[(iRegionIndex << 2) + s]) and it's horizontal connectted neighbor region;
             f2v = Mean value of (RegionAverage[(iRegionIndex << 2) + s]) and it's vertical connectted neighbor region;
             MarkRegion(f3, f2h, f2v, iRegionIndex);            // Among f3, f2h, f2v value, select the greatest one, and
        RegionMap  =  iRegionIndex                              // mark included neighbor regions with same region identity
          if (RegionAverage[(iRegionIndex << 2) + s]  is not marked as a region)
             Pyramid_Region(level - 1, (iRegionIndex << 2) + s, theshold);
Remap(cell)                                                    // Combine cells which are not included in regions
return RegionMap                                               // and with shared border as one region
end function
```

(a)



(b)



(c)

Figure 6. (a) Pseudo code for Pyramid Scan; (b) Result of sensitive regions generated; (c) Final obfuscate regions.

```
Algorithm 3: Hilbert Movement with size restriction
function Hilbert_Movement4restrict(grid, threshold, $x_{max}$)
    RegionMap = Ø
    bState = true;
    for idx = 0 to maxHilbertIdx(grid) do
      i=cell[PosX, PosY];
      if (i >= threshold))            // Initial cell with sensitivity greater than thresold,
        bStart = true;                // start a region
        bState = true;
      if (bStart)                     // Calculate the mean sensitivity value of the region
        if (average < threshold)
          bStart = false;
          bState = true;
      if (bState)                     // when new region start, the flag of
        iMaxCellsCount = 0;           // bState update
        iRegion4Count++;
        bState = false;
      if (iMaxCellsCount >= $x_{max}$)      // Within the region, if cells amount reached $x_{max}$ ,
        iMaxCellsCount = 0;           // then new region conducts to region map
        iRegion4Count++;
      iMaxCellsCount++;
      cell[PosX, PosY].iRegion4 = (iRegion4Count - 1);
      cell[PosX, PosY].fAverage = average;
      RegionMap = cell[PosX, PosY].iRegion4
    end for
    Remap(cell)                       // Combine cells which are not included in regions
    return RegionMap                  // and with shared border as one region
end function
```

Figure 7. Hilbert movement with region size restriction.

sensitive region starts. Add next cells into the region by marking the cells with same identity until the region mean sensitivity value is less than the threshold. If the iMax Cell Count reached the x_{max} value, the region with x_{max} cells separates out as a region and add to region map. The iMax Cell Count counter is reset.

3) If the initial cell is with sensitivity value less than the threshold, the bState flag updates as well and a non-sensitive region starts. Add next cell into the region and mark the cells with same identity until next cell is with sensitivity value greater than the threshold.
4) Check and merge the cells in the reference space with value less than the threshold and share the same border as a region. Add the region to region map.

Thus, we now generate the obfuscate region map with concerning user's sensitivity preference, and restrict the region size up to x_{max}.

5. Experiment and Result

In this paper, we presented three algorithms to generate users' proprietary obfuscate region maps. In this section, we evaluate obfuscate results of the three algorithms. The experiment is divided into three parts. First, we evaluate the obfuscate results in terms of position sharing in stationary status with Hilbert movement (Algorithm 1) and Pyramid scan algorithm (Algorithm 2). Second, we simulate the situation when users are in moving status, and compare the obfuscated sharing positions in all three algorithms. Finally, we perform different settings for x_{max} with Algorithm 3 to evaluate protection performance of sharing positions in different environment circumstances. With proper suggestion in the x_{max} size setting, the obfuscate system is able to prevent malicious tracking of users in both stationary and moving modes.

5.1. Experiment Setup

The experiment reference space assumes to be surrounded by the Traffic Center of Kumamoto City in Japan for 10 km × 10 km square wide size, and we obtain the road and street data set from OpenStreetMap data [20], which is assisted by commercial LBS companies. The rectangular unit cell is defined as 10 m × 10 m square size, so the reference space city size is with 1024 × 1024 grids. Spatial grid map fulfills with features and replaces with sensitivity value from user profile setting accordingly. All features types from geographic data are categorized into 14 types, and the correlated sensitivity values are given previously in **Table 2** as example settings. We randomly pick upinitial locations within the reference space and move along with the actual road or street in 10 to 13 minutes, and sample the user's snapshot locations every minute in the movement. Assume there are 50000 users randomly distributed in the reference space using the location-aware social network applications. We run the experiment on a 64 bits Windows 7 desktop computer with Intel Core i7-3770K 3.5GHz CPU and 16G RAM.

5.2. Result and Discussion

We verify the obfuscate effects by the region's shape, amount, and size. Firstly, we demonstrate an example of 16 × 16 grids map from reference space in **Figure 8(a)**. Based on user profile provided in **Table 2**, sets the threshold value as 0.35, which means 7 types of feature are considered sensitive to user. **Figure 8(b)** and **Figure 8(c)** show the generated polygonal region map by Algorithms 1 and 2, respectively. The result shows that the generated regions are arbitrary in region shape, size, and amount by different algorithms.

The less generated region's amount means the average region size is wider, and the wider region size implies the less position accuracy. In location-aware social network applications, we consider the users' location privacy, but not expect the users' cloaked sharing positions are too far away from the actual locations, since it can cause application mis-querying nearby others. In the 1024 × 1024 city reference space with the feature sensitivity value remain as **Table 2**, the result in **Figure 9(a)** shows Algorithms 1 and 2 perform similarly with higher threshold values. But Algorithm 2 generates lower region amounts with low threshold value. The Pyramid scan algorithm's result is considered too fuzzy comparing to Hilbert movement in low threshold cases. In **Figure 9(b)**, region size over 50 cells (equal to 5000 m^2) is 8% and 7% in T equal to 0.3, and 4% and 4% in T equal to 0.4 by means of Algorithms 1 and 2, respectively. More than 90% of users' sharing positions are protected within 50 cells. Both algorithms provide sufficient location cloaked area for stationary users.

Secondly, we evaluate the situation when users are in moving status with all three algorithms. The trajectory is in downtown walking district, and we assume that a user has non-stop walk in 10 minutes. We set the threshold value as 0.3, sensitivity feature value as in **Table 2**, and with different x_{max} values for evaluation. A stalker refreshes user's location every minute and considers normal moving speed is less than 10 km/h. User's sharing positions are regions' coordinates according to the three algorithms. In **Figure 10**, Algorithm 1 and Algorithm 2 movement showed abnormal walking speed in time 1, 7, and 9. On the other hand, Algorithm 3 represents a more constant and smooth speed in the movement (shown as curve). In the moving status, sharing the obfuscate region position is possible to be detected as fake locations with Algorithms 1 and 2.

Finally, we simulate the region map with different x_{max} value from 10 to 50 in two different types of moving trajectories, downtown and suburban. In downtown area, **Figure 11(a)** shows that the velocity is slightly over 10 km/h in some snapshots when x_{max} value sets to 10, 30, 40, and 50. In suburban area, **Figure 11(b)** shows the

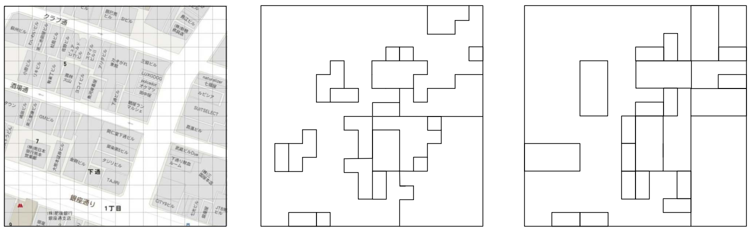


Figure 8. (a) 16 × 16 gird map; (b) Hilbert movement result; (c) Pyramid scan result.

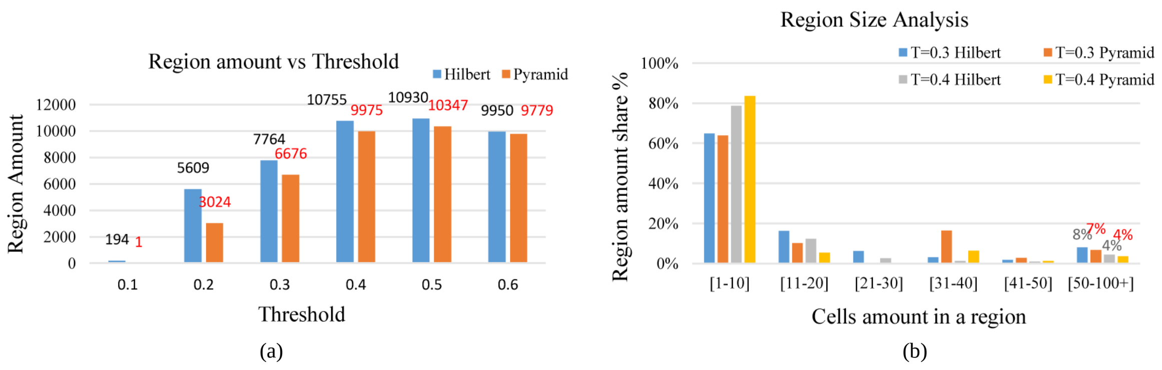


Figure 9. (a) Obfuscate region amount with threshold; (b) Regions amount and region size.

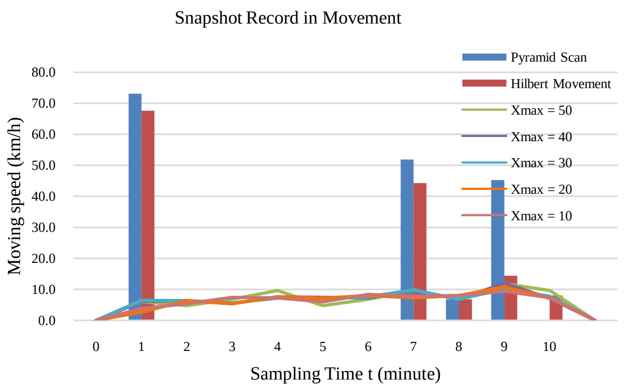
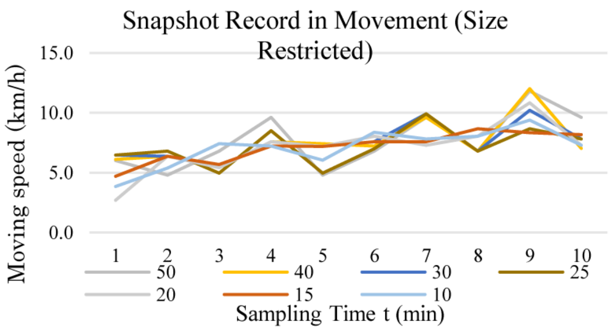
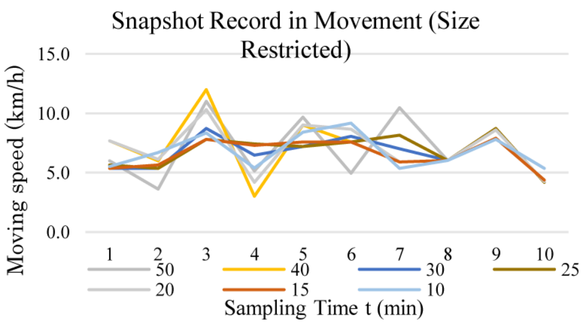


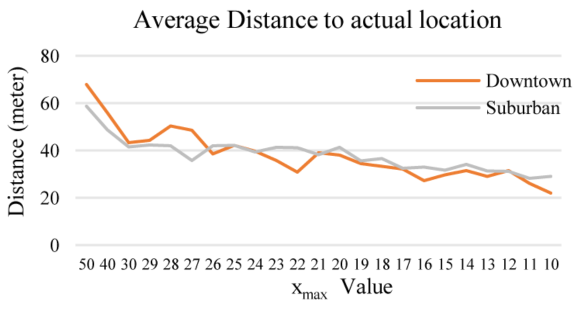
Figure 10. A trajectory in downtown Kumamoto city (walking district only).



(a)



(b)



(c)

Figure 11. (a) Downtown snapshot record; (b) Suburban snapshot record; (c) Distance between actual and transformed location.

abnormal speed happened at values 40 and 50. According to the experiment result, we propose the x_{max} value set in between 10 to 30 shall be suitable for both downtown and suburban areas. In **Figure 11(c)**, we show average distance between the transformed sharing positions and user's actual locations at every snapshot for downtown and suburban areas. For x_{max} value between 10 and 30, the figure indicates the region maps provide 20 to 50 meters distance shifting for protecting user's physical location. The curve slides down when the x_{max} value goes smaller. It indicates that the transformed distance shifting rate is getting less, and the location protection will tend to be insufficient when x_{max} value goes too small.

A general region size comparison in **Table 3** shows in stationary status, the Hilbert movement and Pyramid scan algorithms provide better obfuscate result with wider region size than Algorithm 3, and Pyramid scan algorithm has wider obfuscate result than Hilbert movement for both downtown area and suburban area. However, it might be a problem when users are in the downtown area and adopt the Pyramid scan algorithm to obfuscate the

Table 3. Region size in different algorithm and x_{max} value.

Algorithm		Downtown (cells)	Suburban (cells)
Pyramid		192.7529	273.0667
Hilbert movement		51.3605	244.5373
Region size restriction	50 cells	29.8978	41.5838
	40 cells	26.5133	34.7119
	30 cells	20.2005	27.0810
	20 cells	16.7184	18.9192
	10 cells	9.4760	9.8639

actual location, since high degree of obfuscating the actual positions is not practical for searching nearby others in location-aware social network applications. On the other hand, Hilbert movement algorithm and Hilbert Movement with region size restriction algorithms are suitable for position sharing in location-aware social network applications. The user proprietary profile creates the arbitrary shape of regions to distinguish one from others with same geographic data. The Hilbert movement region map protects user's sharing location in stationary, and the region size restricted approach protects the user's sharing positions in the move. The combination of Algorithms 1 and 3 is suitable for location privacy protection and position sharing in location-aware social network.

6. Conclusion and Future Work

In this paper, we have proposed a user proprietary obfuscate system with algorithms to safely guard user's location privacy for position sharing in location-aware social network applications. The experiment verified two algorithms for protecting position sharing in stationary status and simulated protection with the third algorithm in the movement with actual road and street data. We concluded that the obfuscate system is able to provide sufficient location privacy protection and not impacting the continuous position sharing model in location-aware social network. The approach created the arbitrary polygonal regions to obfuscate the user's actual location while sharing the positions, but the possibility existed that a region was partially included in the search range and the assigned coordinate of the obfuscate region was outside the range due to the region's arbitrariness. It impacted the searching accuracy. In the future, we attempt to improve the obfuscate system with algorithm enhancement for the query accuracy purpose, and include more environmental data for analysis.

References

[1] Pew Research Center (2014) Social Networking Fact Sheet. http://www.pewinternet.org/fact-sheets/social-networking-fact-sheet/

[2] Wen, M., Li, J., Lei, J.S. and Yang, J.J. (2012) A Lightweight Privacy-Aware Location Query Protocol in Mobile Social Networks. *Information and Computational Science*, **9**, 4429-4437.

[3] Wernke, M., Durr, F. and Rothermel, K. (2012) PShare: Position Sharing for Location Privacy Based on Multi-Secret Sharing. *IEEE International Conference on Pervasive Computing and Communications*, Lugano, 19-23 March 2012, 153-161. http://dx.doi.org/10.1109/PerCom.2012.6199862

[4] Damiani, M.L., Bertino, E. and Silvestri, C. (2010) The PROBE Frame Work for the Personalized Cloaking of Private Locations. *Transactions on Data Privacy*, **3**, 123-148.

[5] Cho, E., Myers, S.A. and Leskovec, J. (2011) Friendship and Mobility: User Movement in Location-Based Social Networks. *Proceedings of the* 17th *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, 21-24 August 2011, 1082-1090. http://dx.doi.org/10.1145/2020408.2020579

[6] Krumm, J. (2007) Inference Attacks on Location Tracks. *Proceedings of the* 5th *International Conference on Pervasive Computing*, Toronto, 13-16 May 2007, 127-143. http://dx.doi.org/10.1007/978-3-540-72037-9_8

[7] Jain, A.K. and Shanbhag, D. (2012) Addressing Security and Privacy Risks in Mobile Applications. *IT Professional*, **14**, 28-33. http://dx.doi.org/10.1109/MITP.2012.72

[8] Ball, J. (2014) Angry Birds and "Leaky" Phone Apps Targeted by NSA and GCHQ for User Data. http://www.theguardian.com/world/2014/jan/27/nsa-gchq-smartphone-app-angry-birds-personal-data

[9] Krishnamurthy, B. and Wills, C.E. (2010) Privacy Leakage in Mobile Online Social Networks. *Proceedings of the* 3rd *Workshop on Online Social Networks*, Boston, 22-25 June 2010, 4.

[10] Wernke, M., Skvortsov, P., Durr, F. and Rothermel, K. (2014) A Classification of Location Privacy Attacks and Approaches. *Personal and Ubiquitous Computing*, **18**, 163-175. http://dx.doi.org/10.1007/s00779-012-0633-z

[11] Mokbel, M.F., Chow, C.Y. and Aref, W.G. (2006) The New Casper: Query Processing for Location Services without Compromising Privacy. *Proceedings of the* 32*nd International Conference on Very Large Data Bases*, Seoul, 12-15 September 2006, 763-774.

[12] Xue, M., Kalnix, P. and Pung, H.K. (2009) Location Diversity: Enhanced Privacy Protection in Location Based Services. *Proceedings of the* 4*th International Symposium on Location and Context Awareness*, *LoCA*'09, 70-87. http://dx.doi.org/10.1007/978-3-642-01721-6_5

[13] Bamba, B., Liu, L., Pesti, P. and Wang, T. (2008) Supporting Anonymous Location Queries in Mobile Environments with Privacy Grid. *Proceedings of the* 17*th International Conference on World Wide Web*, Beijing, 21-25 April 2008, 237-246. http://dx.doi.org/10.1145/1367497.1367531

[14] Ghinita, G., Damiani, M.L. and Silvestri, C. (2009) Preventing Velocity-Based Linkage Attacks in Location-Aware Applications. *Proceedings of the* 17*th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Seattle, 4-6 November 2009, 246-255. http://dx.doi.org/10.1145/1653771.1653807

[15] Wei, W., Xu, F. and Li, Q. (2012) MobiShare: Flexible Privacy-Preserving Location Sharing in Mobile Online Social Networks. 2012 *Proceedings IEEE INFOCOM*, Orlando, 25-30 March 2012, 2616-2620. http://dx.doi.org/10.1145/1367497.1367531

[16] Lin, D., Bertino, E., Cheng, R. and Prabhakar, S. (2009) Location Privacy in Moving-Object Environments. *Transactions on Data Privacy*, **2**, 21-46.

[17] Cheng, W.C. and Aritsugi, M. (2014) A User Sensitive Privacy-Preserving Location Sharing System in Mobile Social Networks. *Procedia Computer Science*, **35**, 1692-1701. http://dx.doi.org/10.1016/j.procs.2014.08.262

[18] Carey, N. (2005) Establishing Pedestrian Walking Speeds. Project Report, Portland State University, ITE Student Chapter.

[19] OGC Technical Committee (1999) Open GIS Simple Features Specification for SQL. Revision 1.1. Open GIS Consortium.

[20] http://www.openstreetmap.org