Location Traceability of Users in Location-based Services

Yutaka Yanagisawa†
Hidetoshi Kido†† Tetsuji Satoh†,††
†*NTT Communication Science Laboratories, NTT Corporation*
††*Graduate School of Information Science and Technology, Osaka University*
yutaka@cslab.kecl.ntt.co.jp h-kido@ist.osaka-u.ac.jp satoh.tetsuji@lab.ntt.co.jp

Abstract

In this paper, we introduce Location Traceability as an indicator to evaluate time-series location privacy for users in location-based services (LBS), because recently guaranteeing location privacy has become one of the most significant issues in LBSes.

Although evaluating the location privacy of moving users is important, we previously proposed a technique that focused on the location privacy of users who do not move but stay in one place. Therefore, we introduce Location Traceability as an indicator to evaluate the location privacy of dynamically moving users. The location traceability of a user is calculated from a formalized tree structure, which represents all possible paths of the moving user. As a result of simulation experiments, we validated the effectiveness of this method.

1. Introduction

In recent years, based on developments in sensing technology, we can obtain highly accurate trajectories of moving objects using positioning devices such as GPS receivers [2] Moreover, these obtained position data are used in various types of location-based services (LBS). For example, a navigation system is a typical LBS system using position data. A traffic information system, such as VICS [6], provides drivers with traffic information on city streets for avoiding traffic jams. LBSes offer beneficial location information based on user positions.

On the other hand, protecting user location privacy is one significant problem in LBSes. Protecting location privacy means preventing observers from specifying a user position at a current or past time [5]. To obtain location information around a users, the user must transfer data including position $p = (x, y, t)$ of the user to the system. In this manner, once the system receives user positions, a system manager can store data and freely extract patterns from them. When wishing to delete stored position data, the user can ask the manager to delete them, but generally the user cannot directly do so. In other words, users must trust managers to protect location privacy; however, sometimes dishonest system managers disclose personal user data. To avoid this problem, it is necessary to introduce a mechanism that independently protects location privacy from dishonest system managers.

Therefore, we previously proposed a dummy-based anonymous communication mechanism for LBSes [3, 4]. In our proposed mechanism, when users transfer *true* position p_t to a system, they mix it with many *false* positions $p_{d1}, p_{d2} \ldots, p_{dn}$ called dummies without any marks on the true data. Even if an observer tries to track a user, the numerous dummies confuse the observer by camouflaging the true position of the user. In this study, we also discussed indicators to estimate the performance of our proposed mechanism[3] using dummies. Our indicator is an extension of *k-anonymity* [5], one of the most popular indicators for estimating user privacy in databases.

To estimate user location privacy on non-tracking LBSes in which users transfer their positions only once to a server, we defined two extended *location* indicators of *k-anonymity* on the user's position data: 'Ubiquity' and 'Congestion.' These indicators are suitable for estimating the performance of the communication mechanisms of the location privacy of users in non-tracking LBSes; however, a tracking LBS raises another invasion of privacy issue problem. In tracking LBSes, in contrast to non-tracking LBSes, users continuously and frequently transfer their positions to a server. Frequent position transferring allows observers (for example, dishonest system managers) to minutely track the past positions of a user. Since past positions include much more private user information than a position transferred only once, tracking LBSes are a stronger threat to invade the location privacy of users than non-tracking LBSes.

Therefore, in this paper, we discuss an indicator to estimate location privacy in tracking LBSes and propose an extended dummy-based mechanism to protect

location privacy.

We call this indicator "Location Traceability" (LT). LT focuses on two factors: the "path length" of the period or distance that users are moving and the possibility of distinguishing user positions. The LT value is calculated by using a "location traceable tree" (LT-tree), which we construct using the position data of users. The length of the edges of an LT-tree denotes the length of a path of time or distance in which users are moving. The number of edges at a node of an LT-tree offers a possible way to specify user positions. Intuitively, while the value of LT is high, the safety of a user is strongly protected.

On the other hands, we also present an extended dummy-based mechanism adapted to tracking LBSes, and the mechanism let dummies move such that the trajectories of dummies frequently cross each other and the user's trajectories. In this paper, we explain the mechanism to move dummies naturally on the road map from user's trajectory.

Moreover, we conducted experiments to evaluate our proposed mechanism using our implemented simulation system. In the experiments, we used several types of real road networks and several *typical* user trajectories. The simulation system calculated the Location Traceability of our proposed algorithms of dummies moving using road networks and the given trajectories. As a result of the experiments, we concluded that our proposed mechanism can protect the location privacy of users in tracking LBSes. This paper describes the detailed results of experiments with the implementation of the simulation system.

The rest of our paper is organized as follows. In Section 2, we discuss how to estimate the location privacy of tracking LBS. Section 3 describes a method to construct LT-trees from position data. Furthermore, we explain a method to calculate LT from the LT-tree in Section 4. In Section 5, we explain dummy-based mechanisms to protect location privacy, and Section 6 shows simulation experiments.

2. Location privacy for an LBS

LBS is a service that provides users with geographic information by using position data obtained from GPS. Although an LBS is a useful service, position data stored in an LBS may invade user privacy. This problem is called invasion of location privacy. Beresford and Stajano argue that a system storing position data often invades location privacy [1]. For preventing the invasion of privacy, it is necessary to consider mechanisms to protect location privacy. To solve this invasion problem, we present two methods: protection and evaluation of location privacy. In this section, we explain the mechanism of a LBS and our anonymous communication technique and discuss methods to evaluate location privacy.

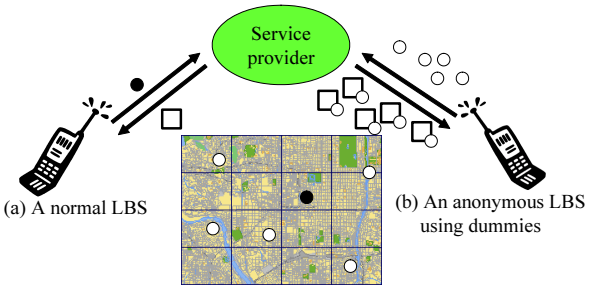

Figure 1. Location-based services

2.1. LBS and protection of location privacy

Figure 1 shows an example of the LBS service process. In this example, after user (a) sends both a user ID and position data to a service provider, the user receives service data corresponding to the position. Although the user ID includes information about who uses the service, anonymizing the user ID is difficult because it requires a toll to the identified user. On the other hand, the service provider can find a position and a time for the user from the received position data. It's highly unlikely that observers can invade location privacy from the position data only. However, if observers can obtain both position data and personal information, position data may allow an invasion of location privacy.

To avoid this problem, we proposed a technique in which users send several false position data (dummies) along with true position data [3], as explained using Figure 1 where another user (b) sends several dummies with its own position data to a service provider. The service provider replies with all service data in response to receiving all position data. The user receives the service data and selects only the necessary data by using its own position data, which it previously memorized. In this technique, the service provider cannot distinguish true position data from the dummies. Therefore, the user can obtain the service without revealing its true position. Moreover, in [3], we proposed algorithms that allow dummies to move naturally.

2.2. Evaluation of location privacy

For evaluating location privacy for LBS, in [3] we also proposed "Ubiquity," defined as a quantifier represented as the degree of scatter of a user and dummies at one time. If the user and dummies are spread widely over the entire area, it is extremely difficult for observers to specify the user's location. In this case, the Ubiquity value is high.

However, Ubiquity is not powerful enough to adequately evaluate location privacy. Ubiquity can only evaluate the location privacy of non-moving users, not

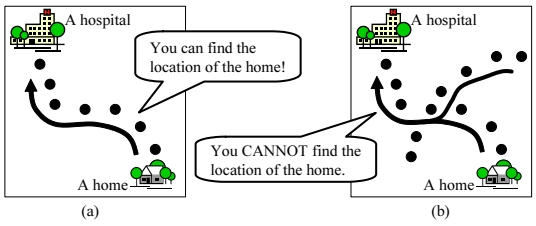

Figure 2. Example of crossing position data in road navigation services

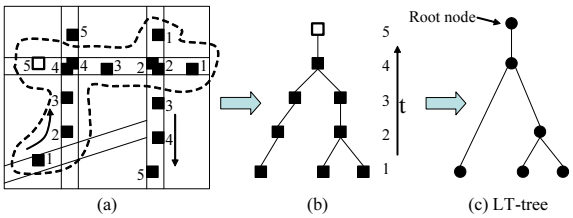

Figure 3. Construction of an LT-tree

of moving users. Most users of practical LBS systems frequently send position data to obtain the latest information. For example, a user of a navigation service must continuously send position data to receive the latest direction information about the user's destination. Thus, evaluating the location privacy of a moving user is more important than evaluating a non-moving user.

Since a moving user risks being traced by observers who monitor its position data, Ubiquity cannot evaluate the user's location privacy. If a user frequently sends position data, the sent position data construct a trajectory, as shown in Figure 2(a). Such a trajectory enables observers to easily find past position data, a risk defined as location traceability. As explained above, Ubiquity cannot evaluate location traceability.

Therefore, we must consider a new location privacy evaluation method for moving users. We call our method that accomplishes this "Location Traceability" (LT).

3. Location Traceable Tree (LT-tree)

In this section, we describe how to construct an LT-tree from position data. For legibility, we denote elements (x,y) of the position data as a position. We also denote a sequence of positions as a trajectory.

As a method to prevent observers from tracing a location, we consider the crossing between a user's trajectory and dummy trajectories, since the crossing of trajectories causes observers to confuse past user positions, as shown in Figure 2(b). Figure 2(b) illustrates two trajectories leading to a hospital. Observers cannot find the route of a user who goes to the hospital due to the crossing of trajectories. Therefore, we introduce a tree structure in which crossing is defined as a node. We define the tree structure as a Location Traceable Tree (LT-tree).

Before explaining LT-tree, we give the following moving user assumptions in LBS.

1. Users move on a road network that consists of roads and intersections.

2. A service provider cannot distinguish a user from a dummy. For example, it assumes that a user and a dummy enter the same intersection. Next, the user and the dummy move in different directions. In this case, the service provider cannot specify the true position of the user.

An LT-tree is a weighted and directed tree that has a root node, which is the most current position. The root node is the top ancestor, and each node has at least one edge. We explain LT-tree construction using Figure 3.

Figure 3(a) shows a road map and the positions of both a user and dummies. In (a), the number written near a position shows time. When focusing on one of the most current positions, as shown by an unfilled square, the positions surrounded by dotted lines are trajectory candidates of the focused data. The candidate positions sorted in order of time are shown in Figure 3(b). An LT-tree shown in Figure 3(c) can be created using (b). Because an LT-tree clearly shows the crossing of trajectories, it can be effectively used to discuss LT value.

Generally, an LT-tree has the following features:

1. For any position, we can construct a unique LT-tree whose position is the root node.

2. When an LT-tree has few nodes and long edges, observers can easily trace users.

3. If an LT-tree has numerous edges, observers do not trace users at all. In this case, we define the LT value of the LT-tree as zero. (Figure 4(a))

4. If an LT-tree has just one edge whose length is infinity, observers can completely grasp all user tracks. In this case, we define the LT of the LT-tree as infinity. (Figure 4(b))

5. A node might have multiple parents (discussed below).

Next, we explain how to determine the weights of edges in LT-trees and focus on two elements: moving period and moving distance. Two types of LT-trees using these elements are defined as follows:

Time-based LT-tree We define the weights of edges as the period during the movement between two intersections. Users who stay at one place for a long time have larger weights of edges than users who visit many places.

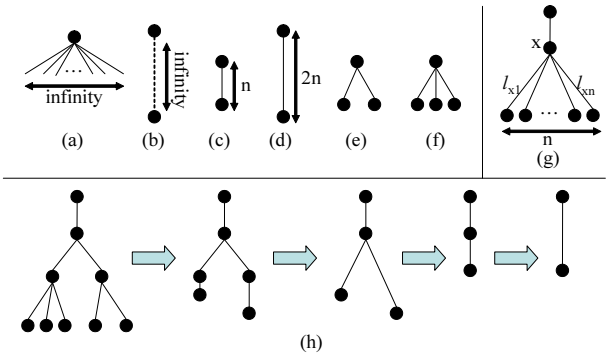

Figure 4. Examples of LT-trees

Distance-based LT-tree We define weights of edges as the distance that a user moves between two intersections. In an LT-tree using this distance, a user who visits many places has larger weights of edges than a user who seldom moves.

4. Location Traceability (LT)

In this section, we explain how to calculate LT value by using an LT-tree.

4.1. Relationship between LT and LT-tree

First, we describe the relationship between the shape of an LT-tree and the LT of a user.

Figure 4 shows examples of LT-trees. Figure 4(a) shows the LT-tree of the safest user. In this LT-tree, an observer cannot specify the past positions of the user. On the other hand, Figure 4(b) shows the LT-tree of the most at-risk user. In this LT-tree, observers can trace positions during past infinite periods. The LT-tree shown in Figure 4(c) (notated as (c)) and the LT-tree shown in Figure 4(d) (notated as (d)) have just one edge whose weights are n and $2n$. The difference in weights shows that the value of the LT of (d) is twice as large as (c). (e) has two edges whose weights are n. In this case, we define the value of the LT of (e) as half as large as (c). Similarly, the LT value of (f) is one third as large as (c). Based on the above discussion, the LT value of any complicated LT-tree can also be calculated because complicated LT-trees also consist of edges and nodes.

4.2. Calculation processes of LT value

We describe the LT definition in detail by using an LT-tree. We denote the LT value of node x as τ_x. When node x is a root node, we denote its value as τ_{all}.

First, we define the τ_{all} of the LT-tree shown in Figure 4(a) as zero and the τ_{all} of an LT-tree shown in Figure 4(b) as infinity. Then, the τ_{all} of all LT-trees

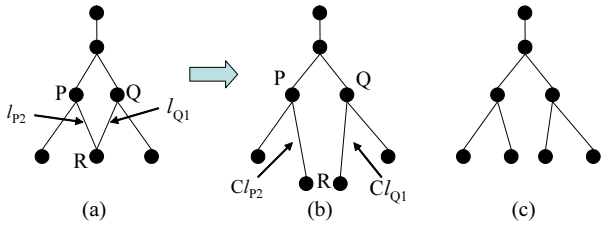

Figure 5. Division of nodes

is between zero and infinity ($0 \leq \tau \leq \infty$). When an LT-tree has just one edge, we define τ_{all} as the weight of edge l ($\tau_{all} = l$).

Next, when an LT-tree has a node with multiple downward edges, we recursively integrate the edges from the leaves of the LT-tree. Figure 4(g) shows an example of the integration of edges. In this figure, n means the number of edges, and $l_{x1}, l_{x2}, \cdots, l_{xn}$ means the weights of edges. When each edge has the same weight, we define τ_x as follows:

$$l_{x1} = l_{x2} = \cdots = l_{xn} = k \Longrightarrow \tau_x = \frac{k}{n}. \quad (1)$$

On the other hand, when the weights of the edges are not the same, we calculate τ_x using one of the following two expressions:

$$\tau_x = \Big(\sum_{k=1}^{n} l_{xk}\Big)/n^2, \quad (2)$$

$$\frac{1}{\tau_x} = \sum_{k=1}^{n} \frac{1}{l_{xk}}. \quad (3)$$

In expression (2), τ_x is the value that divides the summation of the weights of all edges by the second power of the number of edges. Expression (3) is the same calculation of combined parallel electric resistances. Both expressions satisfy expression (1). When the difference in weights of the edges is large, the value of expression (3) is smaller than the value of expression (2). We denote the integration of edges using expression (2) as *simple integration* and the integration of edges using expression (3) as *resistance integration*. Because of the recursive integration of edges, we can calculate τ_{all}, as shown in Figure 4(h).

Because multiple trajectories often cross each other, a node of an LT-tree might have multiple parent nodes. In this case, we have to divide the node to integrate edges, as shown in Figure 5. In Figure 4(a), node R has two parents: P and Q. However, we cannot integrate edges belonging to node P, Q. For the integration of such edges, we propose a method to divide R into two nodes, as shown in Figure 5(b). Because the τ_{all} of (a) is larger than (c), we multiply all edges connected to R by constant $C(> 1)$. After the division of a node is complete, we can integrate the edges.

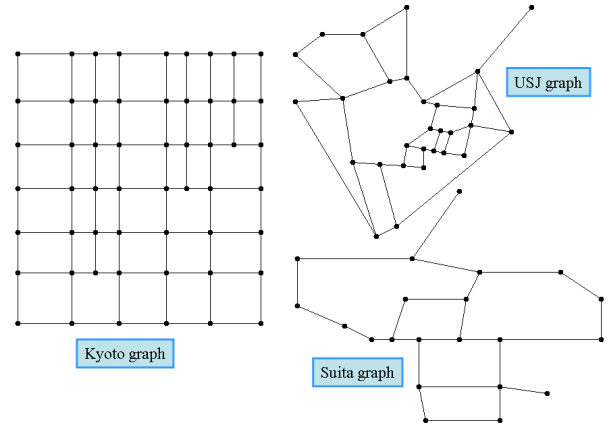

Figure 6. Examples of road graphs

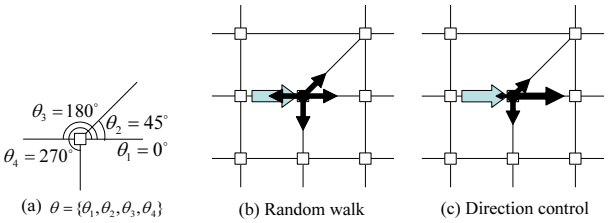

Figure 7. Random walk and direction control

5. Dummy control methods for tracking LBS

In tracking LBSes, the set of position data of a user illustrates a sequence line called a trajectory, as shown in Figure 2. In our proposal, anonymous communication technique, when a user generates dummies at random positions, they do not configure the trajectory. This allows an observer to determine whether position data are true or false. To avoid this, we must control the positions where dummies are generated. In this section, we propose three dummy control methods to reduce the location traceability of users: *random walk*, *direction control*, and *collaborative direction control*. For simplification, we assumed that an LBS user moves on a graph that consists of edges and nodes, as shown in Figure 6. An edge represents a road, and a node represents an intersection. The user generates dummies on the graph.

5.1. Random walk

A user must generate dummies so that they cannot be distinguished from true position data. Generally speaking, the distance each user can move in a fixed time is limited. Therefore, we estimate a dummy generation position by using the previous dummy generation position. First, we propose the following symbols, $P_n(t)$ and $V_n(t)$.

$P_n(t) = (x, y)$: position (x, y) of dummy n in time t $(t = 0, 1, 2, \ldots, m)$

$V_n(t) = (V_{x_n}, V_{y_n})$: displacement value of dummy n among $[t-1, t]$

In the case of $t = 0$, a user generates several dummies on nodes chosen at random. The velocity at which a dummy moves is v, identical to the velocity in which a user moves. Therefore,

$$|V_n(t)| = \sqrt{(V_{x_n})^2+, (V_{y_n})^2} = v. \tag{4}$$

When $P_n(t)$ is on an edge, the position of the next dummy $P_n(t+1)$ is decided by the following formula, unless the distance between $P_n(t)$ and $P_n(t+1)$ is shorter than the distance between $P_n(t)$ and the node for which dummy n heads.

$$P_n(t+1) = P_n(t) + V_n(t) \quad (t \geq 1). \tag{5}$$

If dummy n arrives at a node, $P_n(t+1)$ is the position of the node. We do not consider cases where $t = 0$ because we assume that dummies are generated on one of the nodes in $t = 0$.

Next, when $P_n(t)$ is on a node, the position of the next dummy $P_n(t+1)$ is decided by the following formula:

$$P_n(t+1) = \begin{cases} P_n(t) + |V_n(t)|(\cos\theta, \sin\theta) \\ \text{(move to another node)} \\ P_n(t) \text{ (stop at the node)} \end{cases} \tag{6}$$

Symbol θ represents one of the angles of the edges connecting a node, as shown in Figure 7(a). In the case of Figure 7(a), θ takes four values: $(\theta_1, \theta_2, \theta_3, \theta_4)$. Again, values θ are chosen randomly. Therefore, when arriving at a node, a user chooses an edge without reference to the direction from which the dummy approaches, as shown in Figure 7(b). We denote this method as *random walk*.

5.2. Direction control

In the *random walk* algorithm, the movement of a dummy is not natural because it moves too randomly. Because a user needs to generate natural movement for dummies, we propose another algorithm based on the behavior of a real user. Generally speaking, when a user arrives at an intersection, the user often goes straight and rarely retreats. Therefore, we consider the behavior of dummies on a node, that is, how to determine value θ of the *random walk* algorithm. We

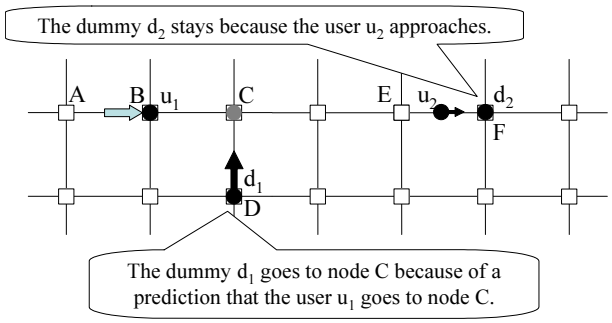

Figure 8. Examples of collaborative direction control method

propose the following algorithm to determine value θ. We denote the angle that a dummy arrives at a node as θ_{in} and the angle that a dummy leaves as θ_{out}.

An algorithm to determine value θ

if ($150° \leq \exists|\theta - \theta_{in}| \leq 210°$)
{ θ_{out} = {one of θ} with a probability of 60 %}
else if ($\exists|\theta - \theta_{in}| \neq 0°$)
{θ_{out} = {one of θ} with a probability of 90%}
else {$\theta_{out} = \theta_{in}$ }

Figure 7(c) shows the behavior of a dummy around a node by using the algorithm. Figure 7 shows that the dummy comes from the left side, often goes straight, sometimes turns left or right, and rarely retreats. This behavior resembles the behavior of a real user. We denote the method as *direction control*.

5.3. Collaborative direction control

In the *direction control* algorithm, each dummy moves independently. If dummies and the true user cross each other, the user's location traceability decreases. Therefore, we propose a method in which dummies move so that they and the true user frequently cross each other. The method is based on the *direction control* method. We add the following two conditions to the *direction control* method.

1. Predicting destination

   The key idea of this condition is that a dummy predicts the destination of another dummy. The condition is defined as follows.

   Predicting destination

   A dummy is on node X. A user is on node Y, which is two hops away from node X. The user comes from node Z.
   In this case, if node U is one hop away from both nodes X and Y, and node Y is between node U and node Z, the dummy goes to node U.

   We explain this condition using Figure 8. In Figure 8, symbols $A, B, C, D, E, andF$ show nodes at the white squares, u_1, u_2 show the true user or a dummy at the black circles, and d_1, d_2 show dummies illustrated at the black circles. User u_1 on node B comes from node A. Dummy d_1 is on node D. In this case, dummy d_1 goes to node C because of the prediction that user u_1 goes to node C.

2. Waiting for approaching user
   The condition's key idea is that a dummy waits for the approaching user. This condition is defined as follows.

   Waiting for approaching user

   A dummy is on node I. A user is on an edge that connects to node I. Symbol r shows the accuracy of position data.
   In this case, if the user approaches node I and the distance between the user and the dummy is between r and $2r$, the dummy stays at node I.

   We also explain the condition using Figure 8, in which dummy d_2 stops at node F because another user u_2 approaches node F.

The above two conditions create a chance for a crossing between a dummy and the true user or another dummy. Moreover, because the movement of each dummy only slightly changes, the dummy rarely moves along with the true user. As a result, the user's location traceability decreases. We denote this method as *collaborative direction control*.

6. Experiments

To evaluate LT validity, we implemented a simulator. In this section, we describe the experiments conducted using the implemented simulator.

6.1. Simulation environment

To validate the effectiveness of our proposed technique for tracking LBS, we implemented another simulator. Figure 9 illustrates a user interface of the simulator, which can load a graph that has edges and nodes,

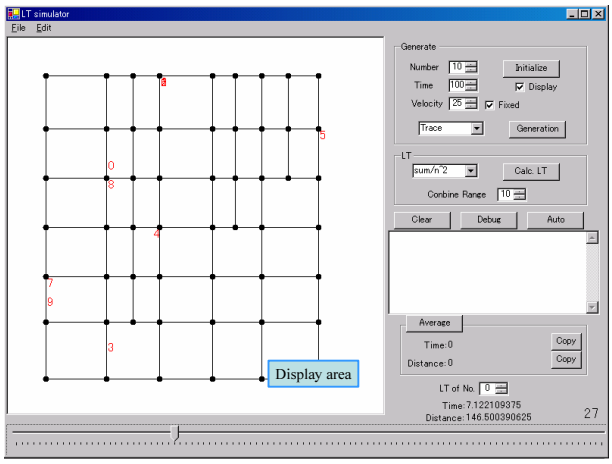

Figure 9. User interface of our implemented simulator for tracking LBS

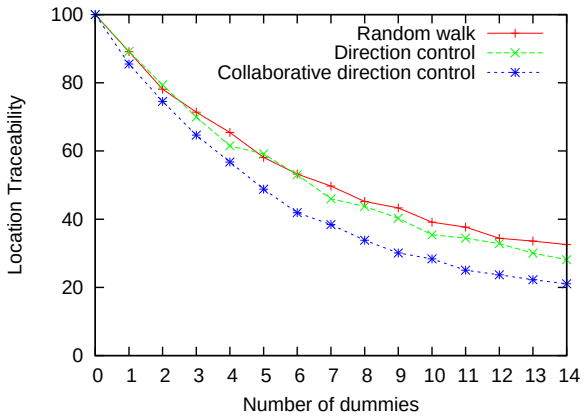

Figure 10. Location traceability values by dummy control methods

as shown in Figure 6. The simulator can also load the position data of a user moving on the graph. Moreover, the simulator can generate dummies that also move on the graph. Our proposed dummy control methods determine dummy behavior. When users or dummies A and B come close within a fixed distance, the simulator recognizes that they are crossing. The simulator generates an LT-tree from the crossing data and then calculates location traceability by using the LT-tree.

In this experiment, we set up the following parameters and data:

Graphs	:	Kyoto, USJ, and Suita (shown in Figure 6)
Dummy control methods	:	Random walk, Direction control, Collaborative direction control
Distance users are crossing	:	5,10,15,20 (default: 10)
LT-tree type	:	Time-based LT-tree
Integration type	:	*Simple integration*
Velocity of users	:	3 or 4 *km/h*
Number of dummies	:	$1 \sim 14$,
Data gathering time	:	100,

6.2. Results

Figure 10 shows the values of location traceability in each dummy control method. The X axis in Figure 10 shows the number of dummies, and the Y axis shows the values of location traceability (τ_{all}). We use the Kyoto graph. As shown in Figure 10, to halve location traceability, a user needs eight dummies in the *random walk* and *direction control* algorithms. In the case of *collaborative direction control*, however, the user only needs six dummies. Therefore, the tracing algorithm improves the effect of reducing location traceability.

The reducing rate of location traceability is shown as Figure 11. The X axis shows the number of dummies, and the Y axis shows the reducing rates (%). The following formula calculates the reducing rates:

$$(\textit{reducing rates}) = \left(100 - \frac{(\textit{random walk}) \text{ or } (\textit{direction control})}{(\textit{collaborative direction control})}\right) \times 100. \quad (7)$$

This figure shows that when the number of dummies is large, the reducing rate is also high. In particular, when there are more than six dummies, the reducing rate is more than 20Therefore, the *collaborative direction control* method improves location traceability more than the other two algorithms.

Figure 12 shows the values of location traceability in each graph. The dummy control method is *collaborative direction control*. In the USJ (Universal Studio Japan) and Suita campus of Osaka Univ. graphs, we investigate the location traceability of two types of user movements. User 'USJ-1' moves throughout the entire network, while user 'USJ-2' moves only in a complicated section. User 'Suita-1' stops frequently, but user 'Suita-2' stops infrequently. This figure shows that the location traceability of users 'Kyoto' and 'USJ-2' is lower than other users because the Kyoto graph has more nodes than the other graphs. Dummies are distributed throughout the entire graph, unlike user 'USJ-2.' Therefore, we found that the complexity of graphs or the behavior of users directly effects the values of location traceability.

Figure 13 shows the values of location traceability for each distance at which users cross each other. Here we use the Kyoto graph. The dummy control method

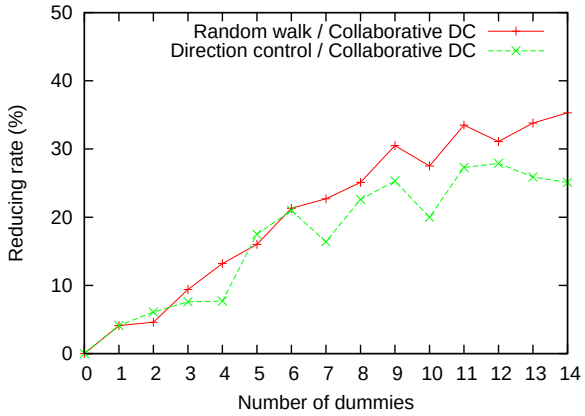

Figure 11. Reducing location traceability rate in each dummy control method

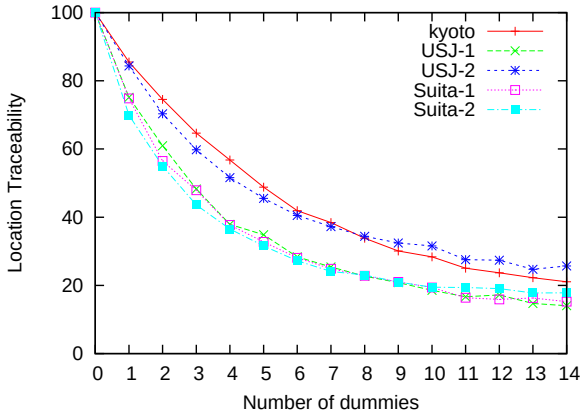

Figure 12. Location traceability values in each graph

is *collaborative direction control*. This figure shows that when the distance is longer, values of location traceability are larger, but the difference between the values is not so large.

As a result, we conclude that our proposed technique strongly protects the location privacy of users in an LBS.

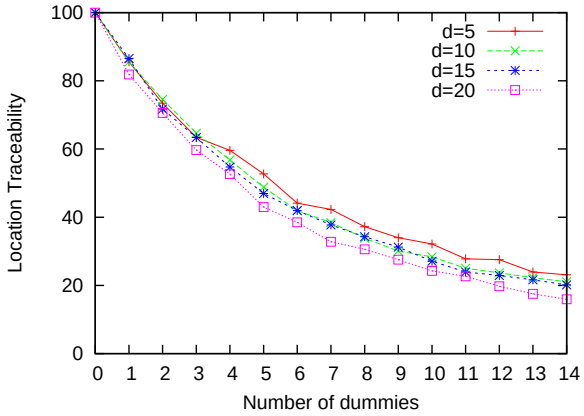

Figure 13. Location traceability values for each distance at which users cross

7. Conclusion

In this paper, we proposed a method to protect the location privacy of continuously moving users in tracking LBSes. We described how to create location traceable trees needed for the proposed method. As a result of simulation experiments, we validated the effectiveness of this method, moreover, we proposed a generation algorithm for dummies. In future, we will implement our mechnisms to the real LBSes to evaluate the performance of our proposed mechanisms.

References

[1] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *Proceedings of Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 127–131, Mar. 2004.

[2] I. Getting. The global positioning system. In *IEEE Spectrum*, volume 30, pages 36–47, December 1993.

[3] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *Proceedings of IEEE International Conference on Pervasive Services 2005 (ICPS2005)*, pages 88–97, July 2005.

[4] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of location privacy using dummies for location-based services. In *Proceedings of the International Special Workshop on Databases For Next Generation Researchers (SWOD2005)*, pages 118–122, Apr. 2005.

[5] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *the IEEE Symposium on Research in Security and Privacy*, May 1998.

[6] VICS. http://www.vics.or.jp/english/.