Memorizing Algorithm:
Protecting User Privacy using Historical Information of Location–Based Services

Quynh Chi Truong, National University of Ho Chi Minh City, Vietnam

Anh Tuan Truong, National University of Ho Chi Minh City, Vietnam

Tran Khanh Dang, National University of Ho Chi Minh City, Vietnam

ABSTRACT

The rapid development of location-based services, which make use of the location information of the user, presents both opportunities and challenges. Users can benefit from these services; however, they must often disclose their location information, which may lead to privacy problems. In this regard, the authors propose a solution with a memorizing algorithm, using trusted middleware that organizes space in an adaptive grid where it cloaks the user's location information in an anonymization area before sending it to the service providers. This newly introduced memorizing algorithm calculates on the spatial grid to decrease the overlapped areas as much as possible, which helps conceal users' locations. This solution protects the user's privacy while using the service, but also against data mining techniques with respect to their history location data. Experimental results with a user activities map establishes this theoretical analyses as well as the practical value of the proposed solution.

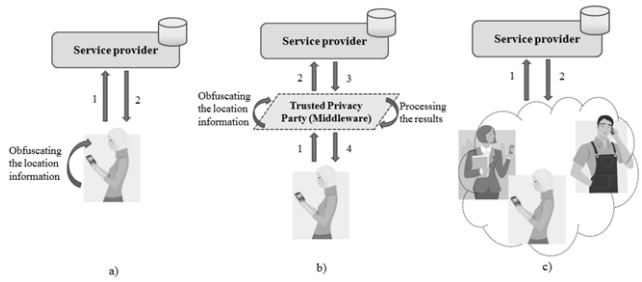
Keywords: *Data Mining, Location-Based Services, Location Privacy, Memorizing Algorithm, Privacy Preserving*

INTRODUCTION

Advances in location technologies and wireless communication technologies enable the widespread development of location-based services (LBS) that make use of the location information of users (Kupper, 2005; Schiller & Voisard, 2004). As location information is a part of users' private information, it requires a number of solutions to protect the location privacy of users while not affecting much on the quality of the location-based services. Location privacy can be defined as the right of individuals, groups, and institutions to determine themselves how, when, to whom, and for which purposes their location information is used (Ardagna, Cremonini, Vimercati, & Samarati, 2008; Kupper, 2005; Mohamed, 2007). When the user location information is not well protected, the user can face various kinds of location attacks. Some attacks just make the user annoying, for instance unconsenting advertisements, while the others can endanger the user such as stalking or physical harassment

DOI: 10.4018/jmcmc.2010100104

Figure 1. The privacy architectures. a) The non-cooperative architecture. b) The centralized trusted party architecture. c) The peer-to-peer cooperative architecture.



(Ardagna, Cremonini, Vimercati, & Samarati, 2008; Atluri & Shin, 2008; Bettini, Mascetti, & Wang, 2008). The problem of privacy preserving in LBS attracts numerous attentions from both research communities and industry sectors (Bettini, Mascetti, & Wang, 2008). The user's location privacy should be safeguarded in two stages. In the first stage, the location privacy should be protected at the time of using services. One popular method is to obfuscate the location with the service providers in order to hide the user's true location information (Mohamed, 2007). The solution focuses on preventing the user's locations from an illegal observation at the time of service calls. However, when a user uses the service several times in a specific area, it will cause an overlapping problem which can be exploited to identify the highest possible area where the user is (Gidófalvi, Huang, & Pedersen, 2007). Then, it leads to the second stage which ensures the user's privacy when the user's location information is stored in the database for data mining purposes (Gidófalvi, Huang, & Pedersen, 2007).

Although there are many researches on this field, they only concentrate on privacy preserving in either the first stage or the second stage. This paper proposes a novel approach for privacy preserving in both stages. Our solution bases on a LBS framework consisting of a trusted middleware (see Figure 1b). We also introduce an algorithm that applied in the middleware.

The algorithm receives the user's location and privacy requirement as inputs; then it cloaks the user's location in a grid-based map. The anonymiztion area yielded by the algorithm will satisfy the user's privacy requirement and also solve the overlapped-area problem. More importantly, the grid-based map is dynamically sizeable according to the user's privacy requirement.

The rest of this paper is organized by briefly summarizing related work. We will present our discussion on the privacy problem of overlapped areas and present our grid-based approach for the problem. Next, we introduce our memorizing algorithm that works on the grid for preserving user privacy in LBS in both cases, namely, fixed-grid-based map and adaptive-grid-based map. Finally, experimental results are discussed, as well as concluding remarks and future work.

RELATED WORKS

In LBS, there are three system architectures for preserving location privacy: the non-cooperative architecture, the centralized trusted party architecture, and the peer-to-peer cooperative architecture (Mohamed, 2007). In the first architecture, users are self-responsible for protecting their location privacy. The users can provide false identities or location to the service providers. They can also create many dummies to hide the true one. This is an easy way to protect location privacy but the critical foible is that it totally depends on users' knowledge (Figure 1).

In the second architecture, there is a trusted party which stands in the middle between the users and the service providers. The trusted party can be a third party server or a middleware (Bellavista, Corradi, & Giannelli, 2005). The main duty of the trusted party is to provide a location transparency mechanism. A location transparency mechanism is defined as hiding all aspects of location information from the service providers, including location values, and positioning methods (Kupper, 2005). First, it receives the location information from the user, blurring and sending the information to the service providers. Then, it filters and forwards back the results to the user. There are several algorithms that can be applied in the trusted party, namely k-anonymity (Bugra & Ling, 2008; Gruteser & Grunwald, 2003; Mohamed, 2007), k-area cloaking (Ardagna, Cremonini, Vimercati, & Samarati, 2008; Marco & Xuan, 2004; Mohamed, 2007), mix zone (Ardagna, Cremonini, Vimercati, & Samarati, 2008; Beresford & Stajano, 2003; Beresford & Stajano, 2004; Marco & Xuan, 2004; Mohamed, 2007), and so on. This architecture fulfills the weak point of the first architecture because it does not rely on users' awareness. Moreover, the architecture is flexible as it separates functional module (the service providers) and the privacy module (the trusted party). However, the disadvantages in this architecture are bottleneck problem and how trust the third party is (Mohamed, 2007).

In the peer-to-peer cooperative architecture, users gather in a group and collaborate with each other so that the service providers could not distinguish a particular user (Mohamed, 2007). The problem is that it is not always that a user has a group.

In general, the second architecture, the centralized trusted party architecture, is the most obvious one to deploy. Therefore, in our solution, we describe an algorithm that works on the trusted party, in particular the middleware.

ANONYMIZATION AREA AND PRIVACY PROBLEM

In the second architecture, when a user wants to use services, he must send his true location information to the trusted party. After anonymizing the user's location, the trusted party sends the anonymized area to the service providers. In case that the service provider is not trusted, the database of users' location information (after anonymized) is not secret. Attackers can get control the database and freely exploit it.

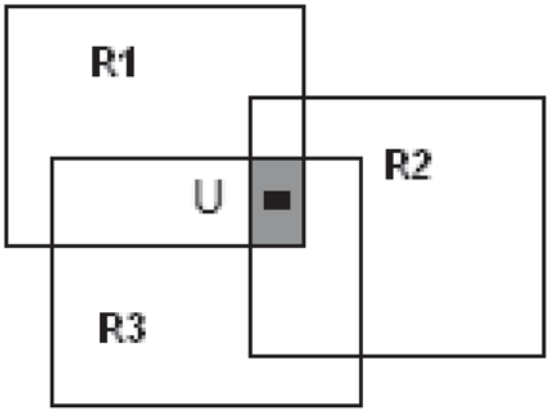
In practical, the user usually uses the services in a specific area. For example, the user lives in a certain district, and uses the location-based services to find the cheapest shopping center around him many times and at different time points. Each time he calls the service; the middleware issues an anonymization area corresponding with his true location and sends to the service providers. The problem is that the middleware does not memorize all previous anonymization areas, so it yields the anonymization areas randomly. This causes an overlapped area problem when the user uses the service many times.

Clearly, the possibility of containing the user in the intersection of these areas is very high. Intuitively, the more anonymization areas attackers catch, the smaller area they can limit. Figure 2 illustrates that the intersection of three anonymization areas R_1, R_2, and R_3 can help in narrowing down the area of a user U.

GRID-BASED SOLUTION FOR THE TRUSTED PARTY ARCHITECTURE

In this section, we will introduce an approach basing on a spatial grid. With this approach, the location of the user will be anonymized on a grid. An anonymization area includes cells and the location of the user is in one of these cells. First, we will consider the definition of the grid.

Figure 2. The overlapped area problem



Definitions

In the grid-based solution, we use a grid to divide the space into cells. The shape of the cell can be either a square or a rectangle but all of the cells must cover the whole space.

An anonymization area is an area that consists of a number of cells. It is used to obfuscate the user's true location information. Figure 3 shows a grid and an anonymization area (in grey color) for the user U. h and w are the height and the width of grid cell. In this paper, we suppose that cells are square for simplicity, i.e., w is equal to h.

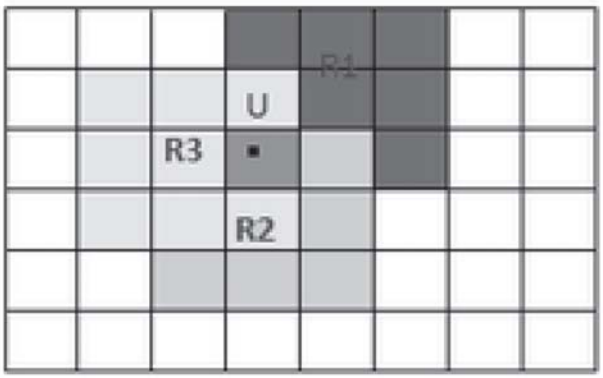
We also define a privacy level as the level that the user wants to blur his true location information. The higher the level is, the larger the anonymization area is, and the more privacy the user has. The privacy level *p* corresponds with a square-shaped anonymization area that contains *p*p* cells. For example, the anonymization area in Figure 3 has 9 cells (3*3 cells) with the privacy level 3.

In this approach, the grid needs a starting point. Depending on the real map, we can choose a proper starting point. Then, we can identify a certain cell by computing the distance between this cell and the starting point.

In this paper, we propose two types of grid-based map as well as two relevant algorithms. They are: fix-grid-based map and adaptive-grid-based map. Fix-grid-based map is the map in which the cell's size is predefined and is the same for all users. In contrast, adaptive-grid-based map allows users to customize the cell's size.

Figure 3. Grid (a) and Anonymization area (b)

Figure 4. The overlapped area



The size of one cell reflects the privacy level 1, i.e., the minimum privacy level. Since each user has a different privacy requirement. Moreover, for a specific user, he/she may even change his/her minimum privacy level when using services. Therefore, it is difficult to decide how large the cell is. If the size of a cell is too small, it is not enough to preserve the location privacy of the user. Otherwise, if the size of a cell is too large, it will decrease the service's quality. To solve this problem, we can design a grid, called adaptive-grid-based map, in which the cells can be resized dynamically according to user's request. However, because of its simplicity, the fixed-grid-based map is useful in the case that the minimum privacy requirement is considered the same for all users.

For algorithms associated with each grid type, firstly, we introduce a simple algorithm for fixed-grid-based map to show the basic idea of the solution. After that, we present a more complicated algorithm for adaptive-grid-based map.

Architecture

First, we will review the second architecture which has a trusted middleware. In the following parts, we call the trusted middleware as middleware for short. When the user wants to use the services, he sends his true location information to the middleware with the required privacy level. When the middleware receives this information, it embeds the location infor-

Figure 5. A problem with grid-based solution

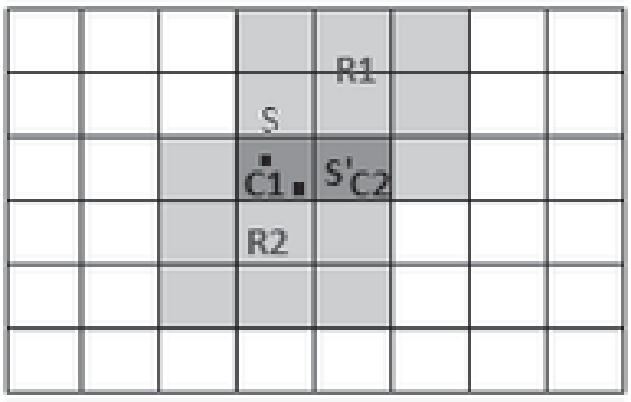
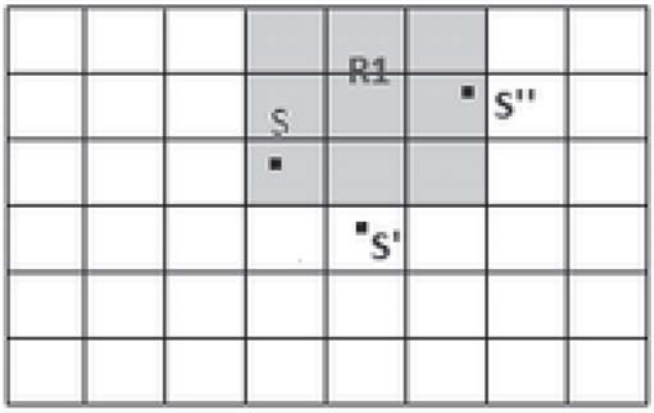


Figure 6. Decision of choosing a rectangle



mation into an anonymization area according to the privacy level. After that, it sends the anonymization area to the service providers. After receiving the results from the service providers, it filters the reasonable results and sends the results to the user.

The grid-based solution also bases on the second architecture. In this architecture, the grid will be put in the middleware. The middleware will choose cells from this grid to form a rectangle according to required privacy level.

In Figure 3, the user *U* sends his true location information to the middleware with the privacy level 3. Then the middleware finds the grid covering the space of the user's location. It also chooses 9 cells to form a 3*3-rectangle and sends this rectangle to the service providers. The anonymization area is colored in Figure 3.

With this grid, it is simple to satisfy a required privacy level of the user. When the user requires a higher level, the anonymization rectangle will be extended and when the user requires a smaller level, the rectangle will be reduced.

Overlapping Problems and the Grid-Based Solution

In the random approach, when the user sends the true location information to the middleware, the middleware will embed this location information into a random area. If the user uses the service at different times, the middleware can send different areas to the service server. Attackers can use these areas to limit the space containing the user's location. If the user uses services more times, the attackers have more changes to find the true location of the user.

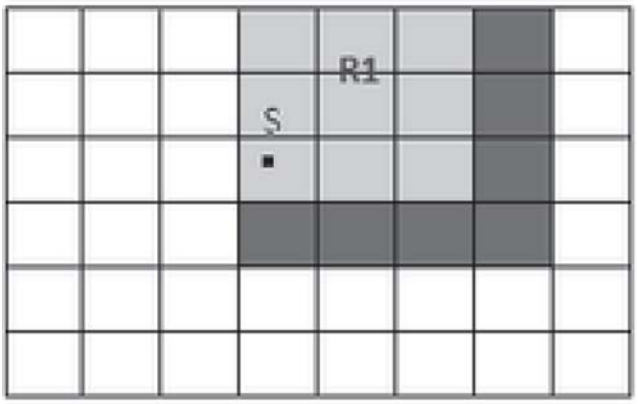
With grid-based solution, the user can uses the services many times but the smallest space which attackers can limit is a cell containing the location of the user. For example, a user *U* uses service at time t_1 and sends his location to the middleware. Then the middleware will send the rectangle R_1 to the service server. Similarly, at time t_2 and t_3, R_2 and R_3 are sent to the service server (Figure 4).

When the attackers have three rectangles, they can infer the true location of the user by getting intersection of these rectangles. The smallest area that the attackers can limit is a cell because the smallest intersection of these rectangles is a cell (Figure 5).

Moreover, we will see an example: at time t_1, the user *U* is at *S* and uses the service with the required privacy level 3 (3*3 rectangles). The middleware will embed his location into R_1. At time t_2, the user is at *S'* and also uses the service with the privacy level 3; his location is embedded into R_2:

However, the attackers can take two rectangles and find that the actual level is 2 cells because the space containing the location of the

Figure 7. Problem when users require the higher privacy level



user is embedded into two cells C_1 and C_2. Thus, it does not satisfy the required level of the user.

Both fixed-based-map and adaptive-grid-based map have the overlapping problem. The only difference is about the cell's size. In the fixed-grid-based map, the cell's size is predefined and is the same for all users, while in the adaptive-grid-based map, the cell's size may vary from user to user or from time to time of a particular user.

In the next section, we introduce an approach to solve this problem. This approach requires that the middleware memorize the anonymization rectangles.

MEMORIZING ALGORITHM FOR GRID-BASED SOLUTION

The two problems have the same cause. It is randomization. At different times, the middleware will create different rectangles and send to the service server. The rectangles are created randomly basing on the user's location. Therefore, the more rectangles are created, the more accurate attackers can find the user's location. To solve these problems, we can use a database to save the rectangles. At different times, the middleware checks the database and finds the proper rectangle(s) if any. Thus, the middleware creates only one rectangle at the first time and uses it for the next times.

Figure 8. Problem when users require the smaller privacy level

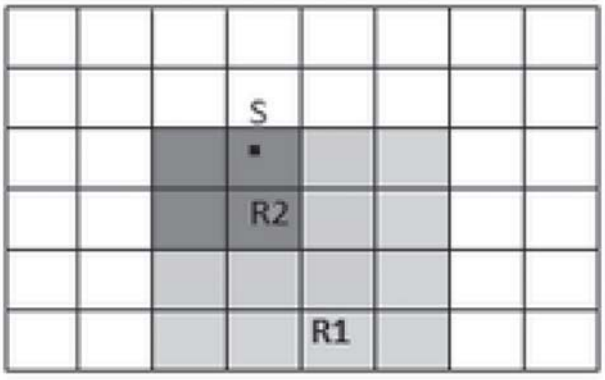
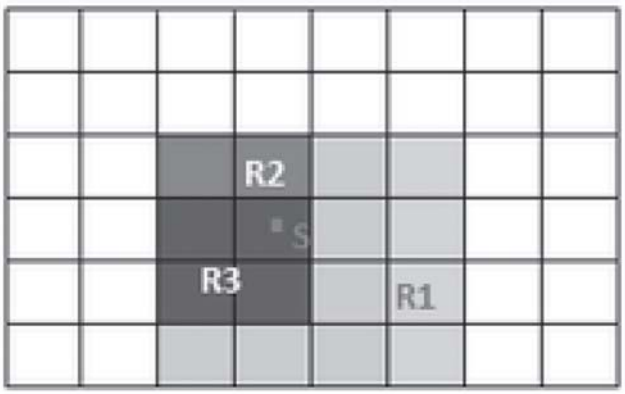


Figure 9. Problem when users require the smaller privacy level



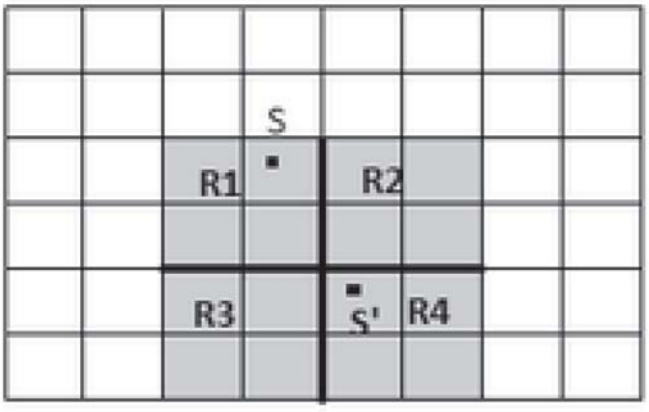
Algorithm with Fixed Grid-Based Map

In short, the mechanism of this solution is as follows:

- When the user wants to use service, he sends his true location information to the middleware and a required privacy level.
- The middleware checks the database to verify whether this is first time the user uses the service in the area or not:
  - If the answer is yes, a rectangle is created randomly according to the required privacy level. Then, the middleware sends this rectangle to the service server and also saves this rectangle to its database.
  - Otherwise, the middleware gets the rectangle returned from the database and sends this rectangle to the service server.

In this algorithm, when the middleware queries the database to find down if this is first time or not, it will need the true location information of the user. For example, at location S (see Figure 6), the user uses the service and, the middleware creates the rectangle R_1 and stores to its database:

Figure 10. A dividing solution



```
if (first time) {
        get random rectangle covering the user's position based on the privacy level;
        save this rectangle and the privacy level;
        return this rectangle;
}
else {
        if (less level){
                perform dividing function and get a proper rectangle;
                save this rectangle;
                return this rectangle;
        }
        else if (greater level){
                get the greatest saved rectangle;
                add some cells to this rectangle to satisfy the privacy level;
                save the added rectangle and the privacy level;
                return the added rectangle;
        }
        else { //equal level
                return the saved rectangle;
        }
}
```

At location *S'*, the middleware checks and finds that the user used the service in the past, but the current location *S'* of the user is not in the area R_1. In this case, the middleware will consider that at location *S'*, the user uses the service for the first time and creates a new rectangle. However, when the user is at *S''* and wants to use the service again, the middleware will find that it is not the first time the user calls the service in this area because *S''* belongs to the area R_1. Therefore, the middleware will return the rectangle R_1 instead of creating a new rectangle.

In summary, when the user requests the service, if this is the first time he uses the service at this position, the middleware will cloak him in a random rectangle which satisfies his required privacy level. Otherwise, it will reuse the anonymization rectangle stored in the database in the previous usage having the same privacy level. Clearly, when the user requires a same level as the first time, the middleware does not need to do anything; it will return the same rectangle saved in the first time. However, the user may change his privacy level comparing to the previous call of the service.

Figure 11. Partial overlap area (a) and Total overlap area (b)

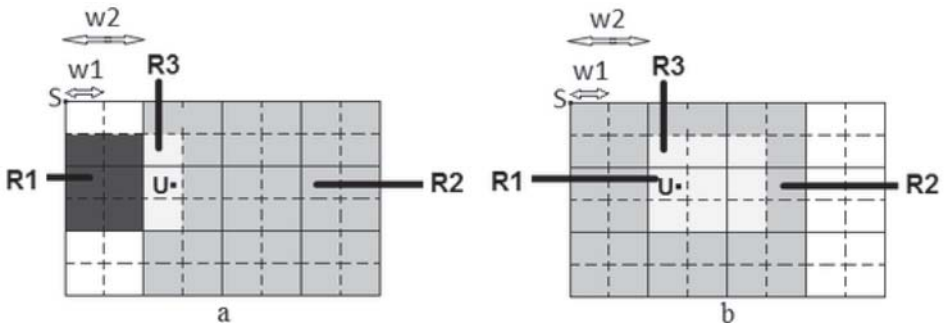
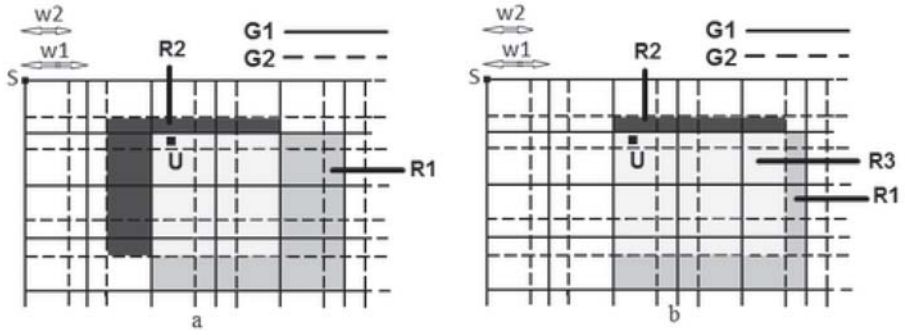


Figure 12. Example for overlap area (a) and Maximal overlap area (b)



At the first time, the user is at S and wants to use the service with the privacy level 3 (3*3-rectangle). Then, the middleware creates a 3*3-anonymization rectangle R_1. It also memorizes R_1 for the next use. After that, he wants to use the service again at a certain location in the rectangle R_1. However, his required privacy level is 4. It means that he requires a bigger anonymization rectangle. The middleware finds that this time is not the first time, so it will get the rectangle saved in the database. Because this rectangle is not big enough to satisfy the required privacy level, some cells should be added to this rectangle to meet the privacy requirement. In this step, we can get some cells randomly and add to the saved rectangle to form a new rectangle. In Figure 7, 7 cells are added to R_1 to form 4*4-rectangle.

In contrary, we will also examine the situation when the user wants a smaller privacy level in same area. At the first time of using service, the rectangle R_1 with privacy level 4 is returned. After that, the user wants to use service again but with the smaller level, for example, level 2. As a result, R_2 is created but not randomly. It must be inside R_1. In other words, it means that we will "reduce" R_1 to R_2 (see Figure 8).

In the next time, the user uses service again and requires the same privacy level as the second time. R_3 is created inside R_1. However, as we mentioned above, the combination of two rectangles can reduce the privacy level of the

Figure 13. Very small overlap area

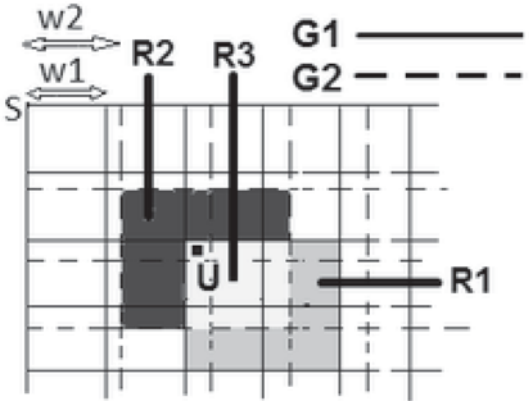
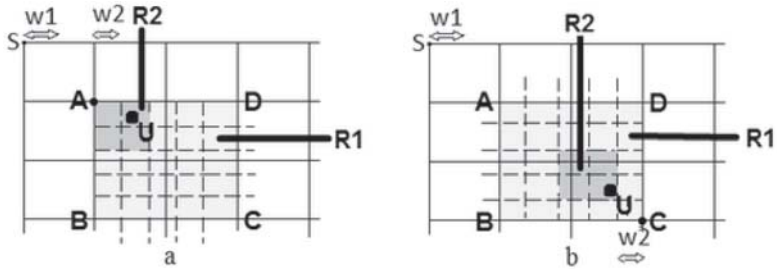


Figure 14. Roving starting point



user. Therefore, the required privacy level of the user is not satisfied. In our example, the combination of R_2 and R_3 can limit the area containing S to 2 cells whereas the required privacy level is 4 cells.

As we discuss before, the reason of these problems is the random in choosing cells for a rectangle. R_2 and R_3 are created randomly so the intersection of them can reduce the area containing S. To solve these problems, the middleware should store rectangle R_2 to the database and returns R_2 if the user is in the area R_2 and requires the service. Similarity, when the user is in R_2 and requires the smaller level, we can solve it as before. We can see it as a recursive process (Figure 9).

However, the saving rectangle will make the database more complicated. It raises a problem related to the service performance. We have to design a data structure which adapts to our solution, i.e., storing effectively and finding quickly the proper rectangle for the service. In addition, when the user moves to a location in R_1 and wants to use service again, the overlapped rectangle can limit the area as we discuss before. To avoid these cases, we can divide the rectangle R_1 to smaller rectangles according to the user privacy level but not overlapping. For example, we can divide R_1 to 4 rectangles (Figure 10).

When the user is at S, the middleware will send the rectangle R_1 to service server, but when the user moves to S', the middleware will send R_4.

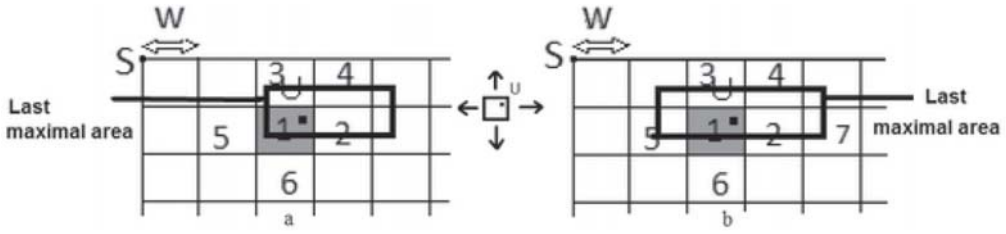
To sum up our solution, we describe the algorithm in the following pseudo code:

Algorithm with Adaptive Grid-Based Map

We will discuss problems with the fix-grid-based map. At the first time, the user uses the service and the middleware chooses the anonymization area, but at the second time and so on, the grid cell size can be resized, so the anonymization area may be changed. Because

```
        if (this is the first time the user uses the service) {
        Get random anonymization area which contains the true location of the user;
        Save this anonymization area;
        Send this area to the service server;
}
else {
        Query the last maximal overlap area of the user;
        Perform overlap_area_getting function;
        Save the anonymization area which have just found in the overlap_area_getting function;
        Save the maximal overlap area;
        Send this area to the service server;
}
```

Figure 15. Roving starting point



the grid cell can be resized, the anonymization area may not overlap totally. The partial overlap is the cause of these problems. The smaller the overlap, the easier attackers can find the location of the user.

To solve these problems, we should combine information from previous times when the user used the service. The middleware will combine information from previous uses of the user to create the anonymization area for the current time. We can describe the mechanism of this algorithm as following:

- The user will send his requirement information when he uses the service to the middleware. As discussion before, the requirement information includes his true location, the required grid cell size and the required level of privacy.
- The middleware will receive the user's requirement information. With the starting point, it will create the grid according to the required grid cell size. Then, it will query the database to check if this is the first time the user uses this service or not:
  - If this is the first time, depending on the location of the user and the required level of privacy, the middleware will choose the anonymization area and save it to the database for future references.
  - If no, the middleware will find all information from previous uses; combine them with the current information to choose the appropriate anonymization area. Then, it will save the current information to the database.
- The middleware will send the anonymization area that has been just created to the service server.
- The middleware will receive return results, choose the acceptable result and return this result to the user.

Figure 16. True and false overlapped area

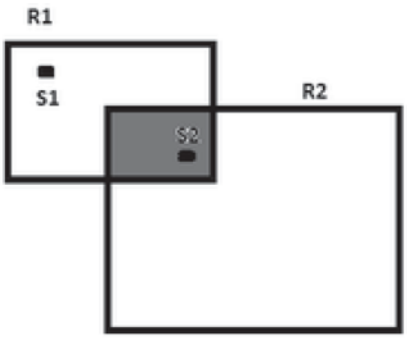
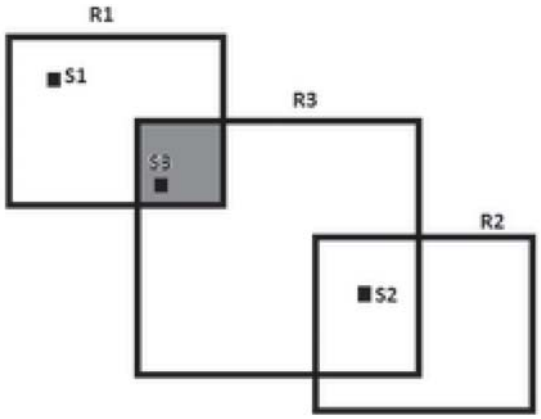


Figure 17. Other example of True and false overlapped area



Clearly, the anonymization area for the current time should totally overlap with previous areas. The total overlap will help our against the information mining of attackers to find the user's location. Reality, if the anonymization area which is created at the second time and so on does not overlap the first anonymization area totally, attackers will limit the area which contains the true location of the user. See the case in Figure 11a, the anonymization area *R1* overlaps partially with the anonymization area *R2*, so attackers can limit the area that contains the user's location to *R3*. In this case, the total overlap area as in Figure 11b is better.

However, the total overlap may not occur at any time. As we discussed before, the grid cell size may change, so the bigger overlap area may not fill all space of the smaller one. For more details, we will consider the example as in Figure 12a: at the first time, required privacy level is 9 cells. Anonymization area *R1* is created and at the second time, required privacy level is 16 cells. Grid *G2* is created and anonymization area *R2* is chosen.

We see that we cannot choose the anonymization area *R2* in order to overlap *R1* totally. In this case, the algorithm should choose the anonymization area *R2* in order to the overlap area between *R2* and *R1* is maximal. The maximal overlap area is R3 in Figure 12b.

As we see in the mechanism, when the anonymization area is created and sent to the service server, the middleware also save this anonymization area to its database for future references. However, what does information need to save? To choose the anonymization area, the middleware will consider all previous anonymization areas. It will choose the anonymization area for current time in order to the overlap area is maximal. Therefore, the middleware should also save all previous anonymization areas. Intuitively, we need just the information of the last overlap area. So, the middleware will choose a new anonymization area so that new overlap area between this anonymization area and the last overlap area is maximal.

Clearly, the partial overlap area will limit the space that contains the users' location. In above examples, the maximal overlap is acceptable because the space, which contains the users' location, is big enough. However, not all of maximal overlap area is acceptable. We will consider the example in Figure 13: anonymization area *R1* is created at the first time and *R2* is created at the second time the user uses the service.

In this case, the maximal overlap area, which intersects between *R1* and *R2*, is *R3*. Actually, the user wants the space, which contains his true location, is *R2* at the second time.

Table 1. Information for each test

Grid cell's size (m)	Frequent area size (m^2)	Number of service calls (times)
50m	2000*2000	100
	1000*1000	
	500*500	
100m	2000*2000	100
	1000*1000	
	500*500	
200m	2000*2000	100
	1000*1000	
	500*500	

However, attackers can find out that the true location of the user is in *R3*. Clearly, *R3* is very small when comparing with *R2*. Intuitively, to solve above problem, we can define the minimal anonymization area. When the maximal overlap area at the current time is smaller the minimal anonymization area, the middleware will choose the previous maximal overlap area and send this area to the service server. However, it is difficult to decide the size of this area because we cannot know how big the minimal anonymization is enough.

We also propose another approach to solve above problem. It is to use a roving starting point. The idea of this approach as follow:

- When the user uses the service for the first time, the middleware will save to its database the information about 4 vertexes of the anonymization area. In Figure 14, they are vertex A, B, C and D.
- At the second time and so on, the middleware will choose one of four vertexes as the starting point. It will create new grid according to the new starting point and return the anonymization area.

As shown in Figure 14a, the vertex *A* is chosen as new starting point. The new grid is created and the new anonymization area (*R2*) will totally overlap with the previous anonymization area (*R1*). In Figure 14b, the vertex *C* is chosen as starting point and the anonymization area is *R2*. It also overlaps totally with R1. The details of this approach will be left as future works.

In short, we can describe the algorithm in pseudo code as follows:

Create the grid according to the user's requirement information;

In this algorithm, the overlap_area_getting() function is very important. The goal of this function is to find the new anonymization area so that the overlap area between this area and the last maximal overlap area is maximal. Therefore, we can describe the mechanism of this function as follow:

- Query the last maximal overlap area from the middleware's database
- Based on the grid that has just been created and the required privacy level of the user. The middleware will choose all anonymization areas according to the user's required privacy. The condition is that these anonymization areas must contain the true location of the user.
- Choose the anonymization area that the overlap area between it and the last maximal area is biggest.
- Return the anonymization that has just found and new maximal overlap area.

To limit the number of anonymization areas, we notice that these anonymization areas must contain the location the user. So we will start at the cell contains the location of the user, we will go forward to four directions from this cell as in Figure 15. At each direction, choose cells that are "the most suitable". We will consider the example in Figure 15a: the starting cell is cell 1. Assume that we want to get a 2*2-anonymization area. With the width, two cells 2 and 5 are considered. We will choose cell 2 because the overlap area between cell 2 and last maximal area is bigger. With the height, it is similar to the width's process. The anonymization area with cells 1, 2, 3, 4 is the best one for 2*2-anonymization area. Another example is in Figure 11b; in this case, we want to choose a 3*3-anonymization area. At the step 1, similar to the Figure 11a, the cell 5 and cell 2 will be considered; we will choose cell 2. At the next step, cell 5 and cell 7 are considered, cell 5 will be chosen. The process for the width is stopped because three cells have been chosen. At the next step, the process for the height will be started and it is similar to the width process.

Finally, we can see that an efficient structure data is important. For a long time, anonymization areas, which are stored to database, is increased. So, a sufficient structure data for saving these anonymization areas is needed.

EVALUATION

Theoretical Evaluation

The main requirements for the location cloaking are Accuracy, Quality, Efficiency and Flexibility as shown in (Mohamed, 2007):

- Accuracy: the system must satisfy the requirement of the user as accuracy as possible.
- Quality: the attacker cannot find out the true location of user.
- Efficiency: the computation for the location cloaking should be simple.
- Flexibility: the user can change his requirement of privacy at any time

However, these criterions should be trade off. The requirement for the best quality will lead to increase the complexity of the computation and so on. In our approach, the user can require the level of privacy to protect his private location. The middleware will choose the anonymization area to hide the true location of the user according to user's privacy level. Furthermore, the user can define the smallest area (cell size) or use the default cell. He can also change his required level of privacy at any time when he wants to use the service. Indeed, the approach can easily satisfy the privacy requirement of the user. When the user wants a high level of privacy, the middleware will expand the anonymization area that contains the true location of the user. Conversely, the anonymization area will be smaller if a lower level of privacy is required. Because the true location of the user is embedded in an area, it is difficult to find the true location of the user. When the anonymization area is enough big, the attacker maybe make more effort to find out the true location of the user.

Moreover, in the algorithm, overlap_area_getting()is the main function and it takes much time to finish. This function will find the anonymization area so that the overlap area between this anonymization area and the last overlap area is the largest. The function will take two loops, one for find cells in the vertical and another one for horizontal. So the complexity of the this function is $O(n)$.

Besides, the complexity of the algorithm also depends on the database access. So, the data structure for saving anonymization areas is needed to decrease the complexity of this algorithm, we discussed it before.

Experimental Evaluation

In this section, we show the evaluation we conducted in order to evaluate the effectiveness of our algorithm. In fact, we only do experiment with the algorithm on the adaptive-grid-based map because the fixed-grid-based map can be considered as a particular case of the adaptive-grid-based map.

We will measure and compare the true overlapped area issuing by our solution, the memorizing algorithm, and by a random algorithm. Firstly, we define the concept "true overlapped area" as follow:

Definition: The true overlapped area is the overlapped area which really contains the user at the present call. Conversely, the false overlapped area is the overlapped area which does not contain the user at the present call

In Figure 16, at t_1, the user is at S1 and uses the service. The true location S1 is blurred in R1. In the next time t_2, the user calls the service again at S2 so R2 is created. The overlapped area (colored area) between R2 and R1 contains the user (or S2) so it is called the true overlapped area.

In Figure 17, R1, S1 are the rectangle and the position of the user for the first time when the user uses the service. At t_2, the user is at S2 and R2 is also created. There is no overlapped area happened. However, at t_3, R3 is created to cover the true location S3 of the user. At this time, there are two overlapped areas. The overlapped area containing S3 is the true overlapped area and the other is the false overlapped area.

Since we want to measure compare the true overlapped area when the user calls services for many times at different positions. We put the user in a particular area, then simulating many service calls from the user. We see that the user maybe use the service many times but in a limited area. So, we set a square-shaped area where the user uses the service more frequently. We set a probability to this area. For example, a frequent area with probability 80% means that if the user uses the service for 100 times, 80 times the user is in frequent area and uses the services and other times is not in this area.

Firstly, we define the grid with a defined-cell size. The area of each cell reflected the minimum security level. This means that the larger the cell is, the more security is provided, but the less quality the service is. We exam the algorithm with the grid cell's size can be changed in each evaluation time. In each evaluation time, the frequent area is also changed and in each case, the user uses the service 100 times. We also notice that the user can change the privacy level at anytime. Table 1 lists the information for each test.

In each case, we run this dataset with our algorithm and with the random algorithm. For each time, we calculate the average of true overlapped area of each time the user uses the service.

Figure 18, Figure 19, and Figure 20 show results of the test 1 in which the grid cell's size is 50m. Figure 18 shows the result for 2000*2000 frequent area sizes. Figure 19 and Figure 20 are the result for 1000*1000 and 500*500 frequent area size respectively. Similarly, Figure 21, Figure 22, and Figure 23 show results of the test 2 in which the grid cell's size is 100m and Figure 24, Figure 25, and Figure 26 are for test 3 in which the grid cell's size is 200m.

Following charts show the relationship between the average of true overlapped area of the user and the number of service calls. In the chart, the black line with lozenge-points represents for the average values of our algorithm while the grey line with square-points stands for the random algorithm.

These results show that the average of true overlapped area yielded by our algorithm is bigger the one yielded by random algorithm. This means that the attacker takes more difficulty to discover the location of the user. The reason for these results is that our approach is to find the anonymization rectangle which the overlap area between previous rectangle and current rectangle is biggest while random algorithm is to find rectangle randomly. These results confirm that our approach is reasonable in reducing the number of overlapped areas. In general, these experimental values can change slightly depending on the dataset but the overall conclusion keeps valid.

CONCLUSION

In this paper, we proposed a grid based solution and a memorizing algorithm that the trusted middleware can use to anonymize the location of the user. This solution solved the overlapping problem occurring when the user uses the services many times a specific location. Moreover, the solution is also flexible as it gives users capabilities to change the cell's size depended on their requirements.

The paper also proposes the solution when the user wants to change his desired privacy level to preserve privacy. However, when the user moves in a trajectory, the area contains the location of the user may be limited but this area is not smaller than a grid cell as we discuss above. To the best of our knowledge, our approach is new and the introduced memorizing algorithm over the grid-like partitioned space is among the vanguard ones that try to minimize the ability that the user's location can be discovered.

Our newly proposed solution opens some potential research issues. As we discussed before, the adaptive grid with fixed starting point will result in some problems. Therefore, the design an adaptive grid with a roving starting

*Figure 18. Test 1: 2000m*2000m, 50m*

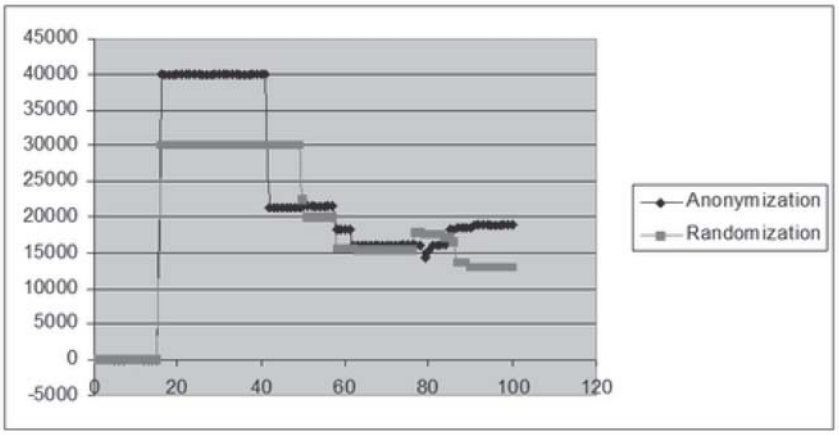


*Figure 19. Test 1: 1000m*1000m, 50m*

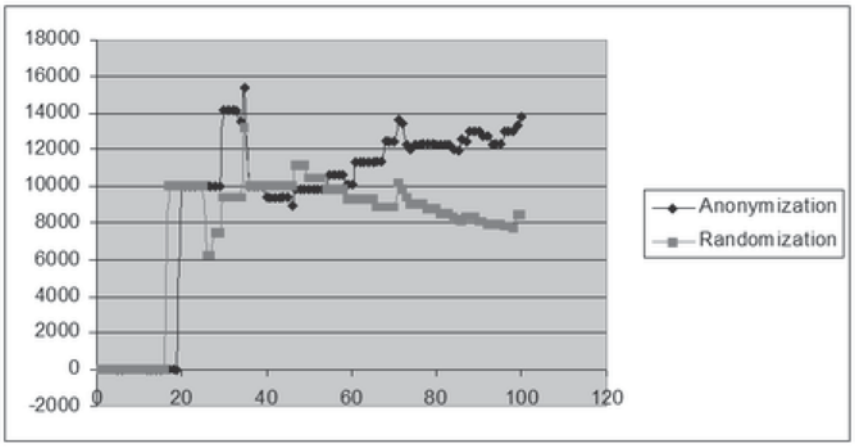


*Figure 20. Test 1: 500m*500m, 50m*

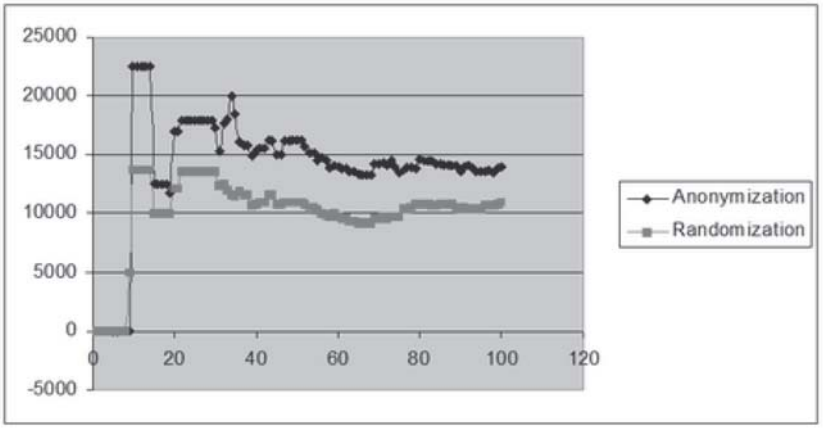


*Figure 21. Test 2: 2000m*2000m, 100m*

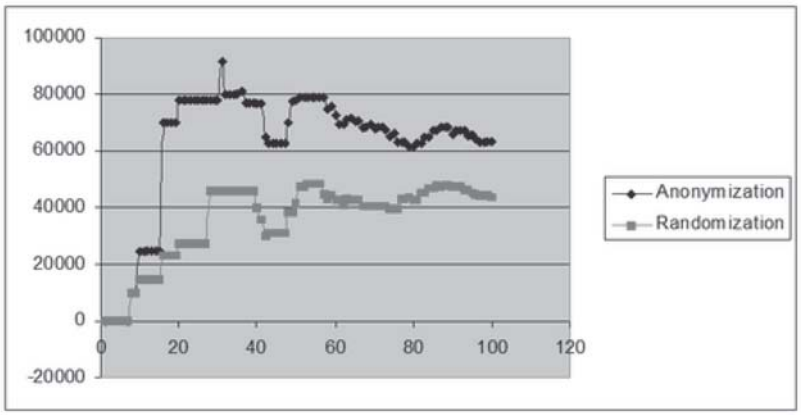


*Figure 22. Test 2: 1000m*1000m, 100m*

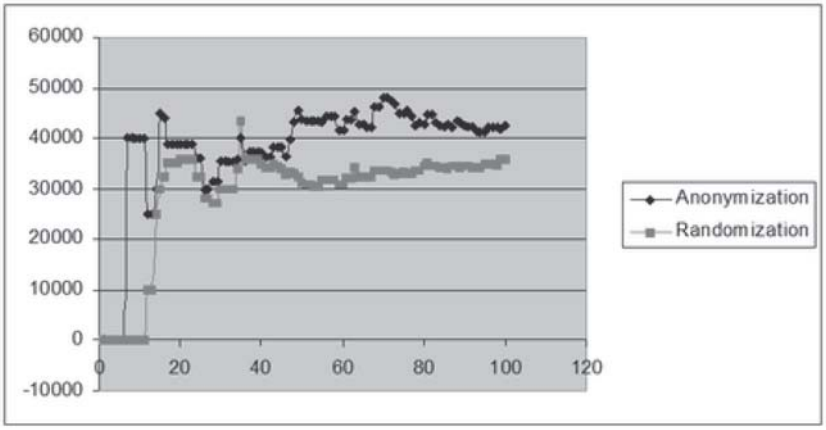


*Figure 23. Test 2: 500m*500m, 100m*

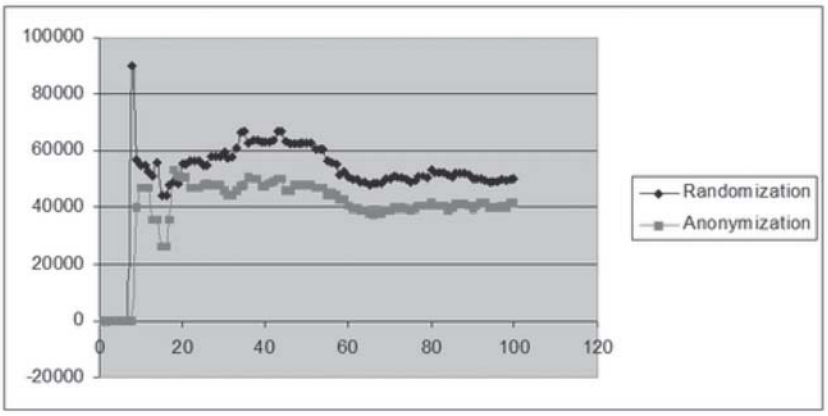


*Figure 24. Test 3: 2000m*2000m, 200m*

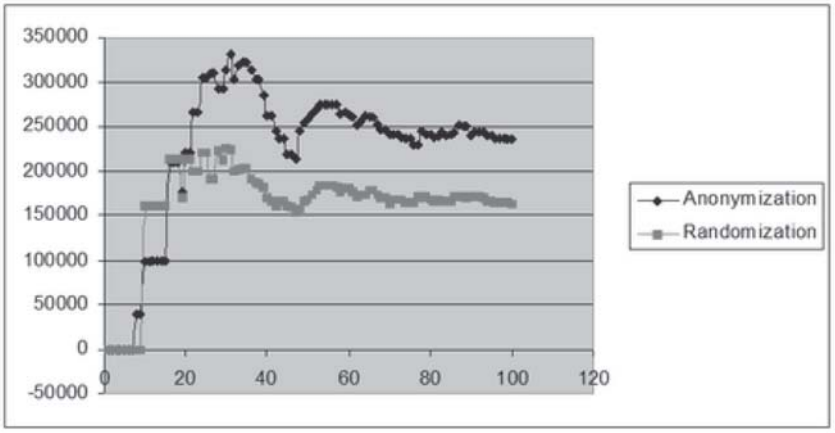


*Figure 25. Test 3: 1000m*1000m, 200m*

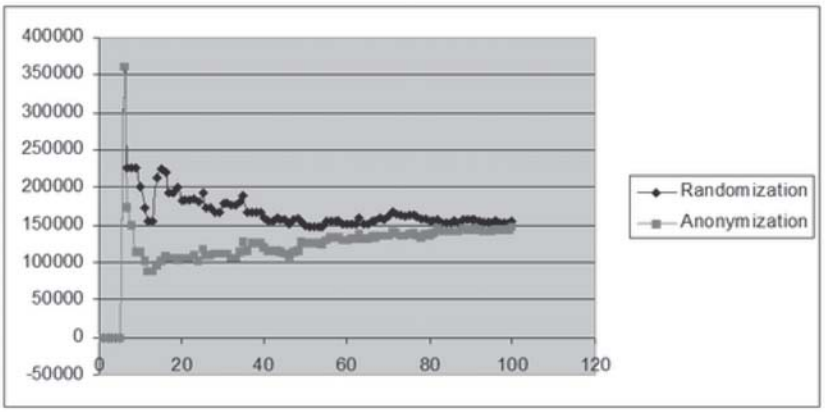


*Figure 26. Test 3: 500m*500m, 200m*

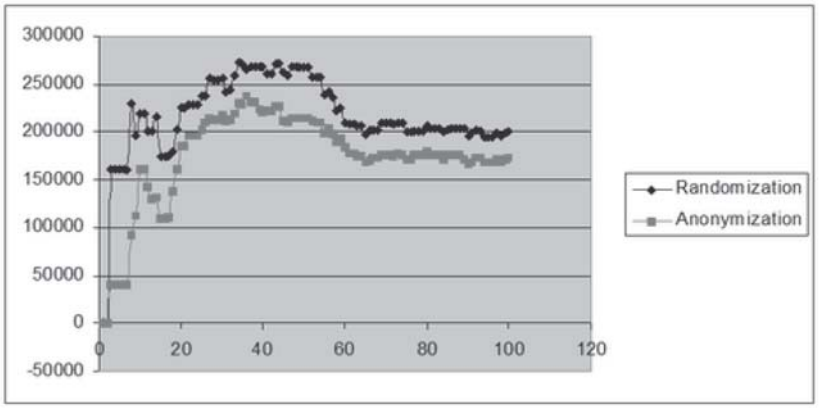


Figure 27. Probability of an anonymity area

$$P = \sum_{1}^{required\_privacy\_level} Pi + R$$

point will make the middleware to protect the privacy of the user sufficiently.

In some case, the anonymization area, which is chosen, will not "big" enough to hide the location of the user. For example, assume that the anonymization area includes four cells; they are cell 1, cell 2, cell 3 and cell 4. However, cell 1, cell 2, cell 3 are regions that the user may not be there, for example, a lake or a swamp, so attackers can limit the area which contains the user's location to the cell 1. A new direction in investigating a new algorithm or a method to eliminate the anonymization area, which contains "dead" regions, should be considered. The probability *P* of an anonymity area which has been chosen should be:

P is the probability of an anonymity area that does not have the "dead" regions. P_i is the probability of a cell that is not a "dead" region. R is the priority of this anonymity area (Figure 27). R should depend on the overlap area between this anonymization area and previous anonymization areas of previous uses. When the middleware wants to choose an anonymity area, it will choose the anonymity area that has the biggest value of *P*. A combination between the grid approach with an algorithm, which helps to find P_i and *R*, will increase the efficiency in protecting the user's privacy.

Again, we notice that the time to carry out the algorithm also depends on the database structure. So, an efficient data structure is needed. The efficient data structure will help us to save the anonymization area efficiently. This will help to reduce the time to get the anonymization area when the middleware wants to query the database. A new direction in designing a new data structure should be also considered.

REFERENCES

Andreas, G. (2006). *Coordinate Transformation - A Solution for the Privacy Problem of Location Based Service*. Washington, DC: IEEE Computer Society.

Andreas, P., & Marit, K. (2000). Anonymity, unobservability, and pseudonymity, a proposal for terminology . In Hannes, F. (Ed.), *Designing Privacy Enhancing Technologies (LNCS 2009)*. New York: Springer.

Ardagna, C. A., Cremonini, M., Vimercati, S. D. C., & Samarati, P. (2008). Privacy-enhanced Location-based Access Control . In Michael, G., & Sushil, J. (Eds.), *Handbook of Database Security – Applications and Trends* (pp. 531–552). New York: Springer.

Atluri, V., & Shin, H. (2008). Effiently Enforcing the Security and Privacy Policies in a Mobile Environment . In Michael, G., & Sushil, J. (Eds.), *Handbook of Database Security – Applications and Trends* (pp. 553–573). New York: Springer.

Bellavista, P., Corradi, A., & Giannelli, C. (2005). Efficiently Managing Location Information with Privacy Requirements in Wi-Fi Networks, a Middleware approach, Wireless Communication Systems. In *Proceedings of the Second International Symposium on Volume Issue* (pp. 91-95).

Beresford, A. R., & Stajano, F. (2003). Location privacy in pervasive computing. *IEEE Pervasive Computing / IEEE Computer Society [and] IEEE Communications Society*, 46–55. doi:10.1109/MPRV.2003.1186725

Beresford, A. R., & Stajano, F. (2004). Mix zones: User privacy in location-aware services. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (CoMoRea '04)*, Orlando, FL.

Bettini, C., Mascetti, S., & Wang, X. S. (2008). Privacy Protection through Anonymity in Location-based Services . In Michael, G., & Sushil, J. (Eds.), *Handbook of Database Security – Applications and Trends* (pp. 509–530). New York: Springer.

Bettini, C., Wang, X., & Jajodia, S. (2005). Protecting privacy against location-based personal identification. In *Proceedings of the Second VLDB Workshop on Secure Data Management* (LNCS 3674). New York: Springer Verlag.

Brinkhoff, T. (2002). A framework for generating network-based moving objects. *GeoInformatica*, 153–180. doi:10.1023/A:1015231126594

Bugra, G., & Ling, L. (2005). A Customizable k-Anonymity Model for Protecting Location Privacy. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, Columbia, OH (pp. 620-629).

Bugra, G., & Ling, L. (2008). Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. *IEEE Transactions on Mobile Computing*, *7*(1).

Clay, S., & Brian, N. L. (2000). A protocol for anonymous communication over the internet. In *Proceedings of the 7th ACM conference on Computer and communications security (CCS-7)*, Athens, Greece (pp. 33-42). New York: ACM Press.

Cuellar, J. R. (2002). Location Information Privacy . In Srikaya, B. (Ed.), *Geographic Location in the Internet* (pp. 179–208). Dordrecht, The Netherlands: Kluwer Academic. doi:10.1007/0-306-47573-1_8

Dieter, S., Marco, C. M., & Siani, P. (2008). *PRIME Architecture Version 3*. Retrieved October 14, 2009, from http://www.prime-project.eu

Gidófalvi, G., Huang, X., & Pedersen, T. B. (2007). Privacy-Preserving Data Mining on Moving Object Trajectories. In *Proceedings of the 8th International Conference on Mobile Data Management (MDM '07)*, Germany.

Gruteser, M., & Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys '03)*, San Francisco, CA.

Kupper, A. (2005). *Location-based Services – Fundamentals and Operation*. New York: John Wiley & Sons Ltd.doi:10.1002/0470092335

Langheinrich, M. (2002). A Privacy Awareness System for Ubiquitous Computing Environments. In *Proceedings of the 4th International Conference on Ubiquitous Computing (UBICOMP '02)*, Sweden (pp. 237-245). New York: Springer Verlag.

Marco, G., & Xuan, L. (2004). *Protecting Privacy in Continuous Location - Tracking Applications*. Washington, DC: IEEE Computer Society.

Mohamed, F. M. (2007). Privacy in Location-based Services: State-of-the-art and Research Directions. In *Proceedings of the 8th IEEE International Conference on Mobile Data Management (MDM '07)*, Germany.

Myles, G., Friday, A., & Davies, N. (2003). Preserving Privacy in Environments with Location-Based Applications. *IEEE Pervasive Computing / IEEE Computer Society [and] IEEE Communications Society*, *2*(1), 56–64. doi:10.1109/MPRV.2003.1186726

Panos, K., Gabriel, G., Kyriakos, M., & Dimitris, P. (2007). Preventing Location-Based Identity Inference in Anonymous Spatial Queries. *IEEE Transactions on Knowledge and Data Engineering*, *19*(12).

Sastry, D., Marco, G., Xuan, L., Paul, M., Ronald, P., Moninder, S., & Jung, M. T. (2002). Framework for security and privacy in automotive telematics. In *Proceedings of the second international workshop on Mobile commerce (WOMM '02)*, GA (pp. 25-32). New York: ACM Press.

Schiller, J., & Voisard, A. (2004). *Location-Based Services*. San Francisco, CA: Morgan Kaufmann. ISBN: 1558609296

Xiao, X., & Tao, Y. (2006). Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD '06)*, IL (pp. 229-240). New York: ACM Press.

Tran Khanh Dang received his BEng degree from the faculty of CSE/HCMUT (Vietnam) in 1998. He achieved the medal awarded for the best graduation student. From 1998-2000, he had been working as a lecturer and researcher in the same faculty. Then, he got a PhD scholarship of the Austrian Exchange Service from 2000-2003, and finished his PhD degree in May 2003 at FAW-Institute, Johannes Kepler University of Linz (Austria). Afterwards, he had been working as a lecturer and researcher at the School of Computing Sci¬ence, Middlesex University in London (UK) since August 2003. In October 2005, he returned home and has continued working in HCMUT. Dr. Dang's research interests include database/information security, modern database applications, and MIS. He has published more than 60 scientific papers in international journals & conferences. Dr. Dang has also participated in and managed many research/commercial projects. Currently, he is the head of the IS Department and director of Advances in Security & Information Systems Lab at CSE/HCMUT.

Ms. Quynh Chi Truong obtained her bachelor degree in Computer Science and Engineering (CSE) in 2008 at Ho Chi Minh City University of Technology (HCMUT). After graduation, she has been employed as a lecturer and researcher at Department of Information Systems, Faculty of CSE, HCMUT. Besides teaching, she is also a key member at Advances in Security & Information Systems Lab (ASIS-Lab) since it was established in 2006. She has also pursued a master degree at the faculty of CSE, HCMUT since 2008. Her research interests are Information Systems, Database Security, and Privacy Preservation in Location-based Services.

Mr Anh Tuan Truong finished his bachelor's degree in Computer Science and Engineering in 2008 at the Ho Chi Minh City University of Technology (HCMUT). He was then an assistant lecturer of the Faculty of Computer Science and Engineering at HCMUT. From 2007 to 2008, he was a software engineer in CSC (Computer Science Corporation), an US-based company. After that, he returned to the Faculty of Computer Science and Engineering, HCMC University of Technology. Currently, he is a lecturer and researcher in the faculty. He has also been a master student at the faculty from 2008. His research interests are Database Security, Privacy preservation, Location-based Service and Software Engineering.