

Secure Proximity Monitoring in Mobile Geo-Social Services

Li Hong Ping
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
hpli@comp.hkbu.edu.hk

Abstract

Nowadays, Location Based Services (LBS) become more and more popular. According to the influence of the social network, users usually join social groups with their friend. In some group, user may want to be notified, if the user is geographically close to any users within the social group. Proximity Detection enable us to attain the goal. However, we are required to bear the risk of disclosure of location information, while we are enjoying the LBS. This privacy threat reduces the attractiveness of this kind of services. Previously, some papers contribute ideas to deal with this problem. Nevertheless, most of them assume that there is a trusted central server within their system, which is impractical in the real world. The problem under our study is to continuously monitor if any two mobile users in a social group are within a distance of D . Meanwhile, the exact location of a mobile user is not disclosed to any third party. This paper propose a computationally feasible solution in this problem, which only disclose the approximate location of the user to the authorized party. Untrusted third party (including centralized server) is not able to know the users' location.

1 INTRODUCTION

As the technology advance, more and more mobile phones and PDAs are equipped with geo-positioning capabilities (e.g. GPS). At the same time, the rise of social networking sites to narrow the gap among people. Friend-locator services (e.g., Google Latitude), which enable user to know their friends' locations, is also becoming popular. Nevertheless, friend-locator services users usually expect certain level of privacy protection rather than completely expose their position to their friends.

The existing research work in proximity detection can only protect users' location privacy in a certain degree. They are not enough to satisfy the requirement of requesting

completely location privacy.

In this paper, we can completely protect the users' location privacy, because we do not require any trusted third party. Under the protection of secure communication, server can just receive the hash value. By using those value to judge where there is any user nearby each other. Server only announces the users, when they are locating within a pre-defined distance of another user. In general, users have different location privacy requirement to other users according to different social group. Such as user Alice allow her family member to know her location when they are in the same district and allow her friend to know her location only when they are within (e.g. fifty meters) from each other.

The design of proposed solution gives three contributions at the same time. Firstly, it can preserve users' location privacy even there is no trusted third party. It outperforms many of the previous solution, which requires the existence of a centralized anonymity server. Secondly, using hash function with a set of time-varied salt gives us a better protection, because it is pretty hard for us to find the user location from the hashed value. We prevent unwanted party to know where we are and let the permitted party to know our approximate location. Thirdly, this solution employed grid base layered structure, which is similar to some of the previous approach. The major different between previous solution and our solution, is previous solution always quad-tree structured grid based layer and our solution use a nona-tree structured grid based layer (see Figure 1). This modification reduces the number of required layer.

Our solution not only gives a good protection in user location privacy, but also requires a low communication cost, which is directly proportional to the number of user. The paper is organized as follows. We briefly review related work in Section 2 and then give a problem definition in Section 3. System Operation is presented in Section 4. Section 5 presents the experimental results of our proposed solution. At last, we conclude our paper in Section 6.

2 RELATED WORK

In this section, we review the development of the location privacy technology and show the contribution of this paper to this topic. In the literature, there are many research efforts in this topic. Most of them adopt one of the three techniques (including cloaking, dummies and encryption) when handling the user location privacy problem.

The earliest proposal for location privacy protection is spatial cloaking, which is proposed by Gruteser and Grunwald [1].

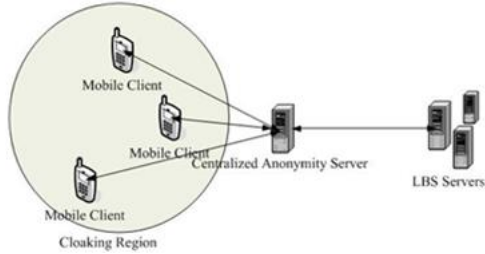


Figure 1. General Structure of Location Cloaking

Instead of sending a single user’s exact location to the server, spatial cloaking techniques collect k user locations and send a corresponding minimum bounding region to the server as the query parameter, see Figure 1. However the quality of service is highly depend on the density of users’ distribution. Also, it is time consuming for searching nearby users to form a cloaking region.

Later, location cloaking algorithms advanced from cloaking of snapshot locations to continuous location updates [3, 4]. The cloaking of snapshot locations is not secure enough to prevent the leakage of location privacy, if an attacker (e.g., the service provider) can collect the user’s historical cloaked regions as well as the user’s mobility pattern (e.g., users’ speed). Except the most common k -anonymity cloaking, there are other types of cloaking method, such as Hilbert curve [5] and Casper[6]. Both of Hilbert curve [5] and Casper[6] employ grid-based cell as their cloaking region, which also the idea that this paper has employed. The major advantage of grid-based cell is it requires less time for us to locate ourselves, comparing with the k -anonymity cloaking which require location of k nearest neighbor to find out our cloaking region. Also, grid-based cell can have a better resistant to path-tracking, as the locations of all cells have been predefined by the system. A. Khoshgozaran and C. Shahabi [5] guarantees the query anonymity even location information is disclosed to the adversary.

However, each client needs to maintain complex data structure and communication protocol as well as long range

communication among peers. Therefore additional computation and communication cost may be quite costly for practical use. For the Casper [6] solution, it blurs a user’s exact location information into a grid-based cloaking spatial region based on user specified privacy requirements. This framework uses a quad-tree data structure that maps the location information into grids with different levels and resolutions. Due to the limitations of the quad-tree structure, the calculated cloaking region is often larger than required, which may cause lower service quality.



Figure 2. Protect location privacy by using faked dummies

On the other hand, [2] suggest that we can protect our location information by faked dummies. Just like Fig. 2 show that when client X send his location service request to the LBS server, he will also send out the k faked dummies (Y,Z,...) simultaneously, so as to diversify the risk of the discovery of his actual location. Although [2] has tried to user some movement simulation technique, we cannot prevent the threat of path tracking, because the exact location must contain in the set of dummies. Therefore, it is not difficult for us to find out the user location by using the assistance of the data mining technique.

The previous solutions only provide some protection for the location privacy. However, most of them require a trusted third party (e.g. trust anonymity server) to process the users’ location data. It is not practical for us to request for a trusted third party in the real world.

Cryptography solution in location privacy can help us to protect the location privacy, while without the existence of the trusted third party. Yao’s [7, 8] present how to exchange secret by using some comparison method. More recently, G. Ghinita and P. Kalnis [9] use

Private Information Retrieval (PIR) implementation to build up a location privacy protection framework which does not require any anonymizers or collaborating trustworthy users. However, the limitation of this implementation is the cell contents have to match the query result that may cause a high storage overhead because the server required storing large amount of different content. Also, it is not easy to find the optimal size of grid partition that makes the computation and communication cost becomes huge.

Many papers have already been published for the topic of finding k-nearest neighbor (kNN).

Except the research of finding kNN, proximity detection is another important topic in location privacy application. The definition of proximity detection is the capability of a location-based service (LBS) to automatically detect when a pair of targets approaches each other closer than a pre-defined proximity distance. It is not efficient for us to do the proximity detection, if we solely use the solution of kNN. That may give us too (less/much) information when the point of interest (POI) are unevenly distributed. Ruppel [10] applies a distance-preserving coordinate transformation. By using centralized proximity detection method to detect the proximity among the transformed locations. However, Liu et al. Liu [11] show that distance preserving coordinate transformations is not safe enough, as it is easy for attacker to derive the secret mapping function. Mascetti present a solution - Hide&Crypt, presented in [12], is a privacy preserving solution which employs a filter-and-refine paradigm. Server uses the specified thresholds and computed distances to determine whether friends are in proximity. However users may need to directly communicate with their friend to check their proximity status, if they are defined as "possibly in proximity". Hide&Crypt use secure multi-party computation (SMC) protocol, which can protect the users' location privacy, but it also brings a choice between the service quality and the communication cost.

More recently, FRIENDLOCATOR [13] and VICINITYLOCATOR [14] both track users in a sparse grid while they are far away from their friends, in order to reduce the communication cost. Changing to finer grid, only when they come closer with their friend.

The limitation of the FRIENDLOCATOR is the proximity detection accuracy of it is low and uncontrollable. For VICINITYLOCATOR [14] give a solution that allow user to choose the "area of interest". In VICINITYLOCATOR, client requires to find all granules contained in his vicinity. We still have to make a tradeoff between the service quality and performance. It is because if the granule size is big, the function of the granules will become meaningless and it also scarifies the accuracy of the result. On the other hand, if the granule size is small, the computation and computation will be high. Also, VICINITYLOCATOR require disclosing the encrypted location to server, because encryption is revisable. So it still not a complete safe solution.

Similar to FRIENDLOCATOR [13] and VICINITYLOCATOR [14], our solution employs an adaptive position-update policy, which use different density layer, in order to reduce the communication cost. Comparing with traditional grid layer structure, our solution use non-tree structure rather than quad-tree structure. Therefore our solution requires less number of layers to solve the problem in some resolution. Also, our solution use hash function instead of

encryption. Hash function is irreversible because there are many different numbers map to the same value. At the same time, because the weak collision probability of hash function, it is rare to find a pair of value, which has some hash value. Furthermore, every time before our location information undergo the hash function, it will combine will a random generated salt. Users can refresh the random salt as they wish. As a result, even users stay in the same place. The message they send to the server is always different. Moreover, comparing with proximity detection, proximity monitoring is a continuous work, we are able to show whenever two users are nearby once the system gets started.

3 PROBLEM DEFINATION

This section describes the system model under our study. We assume users within the social group have reached a consensus on the acceptable distance of proximity detection. Figure 3. show us a sample of user have reached a consensus with all users in different social group. This gives system users flexibility to adjust the proximity distance, according to their need. In our system, there are 2 types of




Social Group	Proximity Distance (m)
 Colleague	20
 Friend	50
 Family	100

Figure 3. Table of acceptable distance of proximity detection for different social group

entities, client and server. Client is the users within the people within the same social group and server is just used for comparison of hashed value. Figure 4 display our system framework. The key idea to preserve the location privacy is separating the information sharing into 2 parts. Traditional centralized anonymity server act as a system center, which responsible for all client requests and analysis work. Therefore, those solutions require making an assumption, that the centralized server must be trusted. It is impractical to expect centralized server to be trusted, because the users' information is valuable. Even if centralized server is non-colluded, nobody can guarantee it will not be broken in by the attacker. Inspired by the solution of Yao's millionaire problem [7,8,15,16], our solution require users to share some standard with the other users directly and the server duty is to analyze the secret value.

Figure 4 show us the framework of our system. Firstly, one of the users in that social group shares a data processing

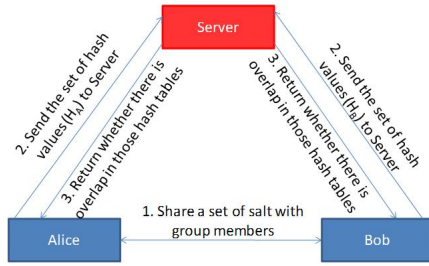


Figure 4. System Framework

standard (e.g. the shifted location of different table, proximity distance) with other group member. Then, all users will transform their location information to a secret value, according to the given standard. After the transformation finish, all users will send their hashed location to the server. Finally, server will check whether all user hashed location to find whether they are within the proximity distance of another user.

4 MECHANISM OF SYSTEM WORK

In this section, we give the detail explanation of our system's operation. Probably, our system workflow can be divided into three stages. The first stage is initialization, which has mentioned in the previous section (Figure 5 step 1), group member establish a communication standard among them in order to achieve the aim of secure multi-party computation (SMC). The following stage is the system operation, in this stage users require to send the hash value to the server. Then server is able to determine whether there is a positive result in detection. The previous stages have already helped us to finish the proximity detection. However, there is a room of improvement for us to minimize the communication cost. In order to reduce the communication cost, we employed non-tree structured grid based layer (see Figure 11). It helps us the filter out the higher probability candidate, eliminate all the unnecessary updates.

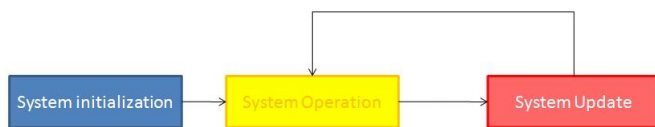


Figure 5. System Workflow

4.1 System Initialization

In the initialization stage (Figure 6), one of the users in the social group generate the random salt and shifted data and share them to other users within the group. The usage of

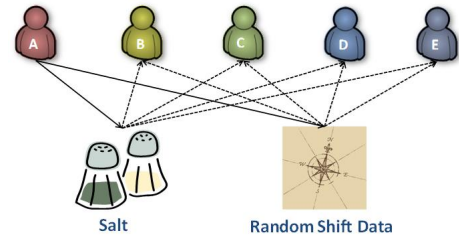


Figure 6. System Initialization

the salt is to make the hashed location become unpredictable to the third party.

4.1.1 Use of Salt

If there is no random salt and system only hashes their location directly, it may be suffer from the brute-for attack. On the contrary, if we combine the location with a randomly generated salt, the hashed location becomes unpredictable.

4.1.2 Use of Random Shifted Data

Our solution use grid based layer (cell size D^2) as a foundation, where D is a proximity distance of the social group. Every user locates in different grid cells of the layer. After that we can have proximity detection by checking whether there are users in the same grid cell. If they are in the same grid cell, we can sure their distance must be within the range of D .

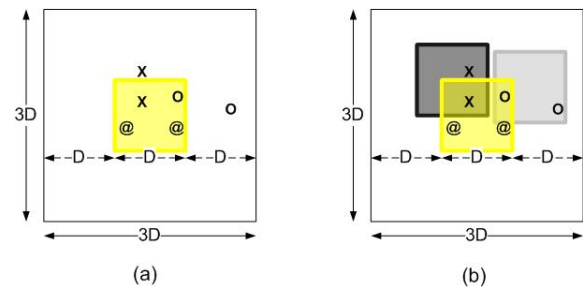


Figure 7. Single Grid Layer vs Multi-Grid Layer

However, if we only use one table to check the distance, we miss too many positive results even they are within a distance D . In Figure 7a, "O", "X", "@" pairs are within distance D with each other. However only "@" is allocated in the same cell, other pairs locate near the edge of the cell. Therefore even their distance between each other is less than D ; they are still treated as unqualified candidate in proximity monitoring. The solution of this problem is add more same size grid based layer and shift randomly to any direction within distance D . The randomly shifted data

is only a set of simple random numbers, which range is between $-(\text{cell length})$ to (cell length) . That mean if the grid layer is in Layer 0, the range of randomly shifted data is between $-D$ to D . For Layer 1, the range is $-3D$ to $3D$, etc. Figure 7b show that after adding more layers "O" pair fall in the gray color grid cell and "X" pair fall in the black color grid cell. In order to simply the demonstration, after we add gray color layer and black color layer we find that "O" and "X" are also within distance D from each other. In fact if we just solely add a few layers is not enough for us to stable our service quality.

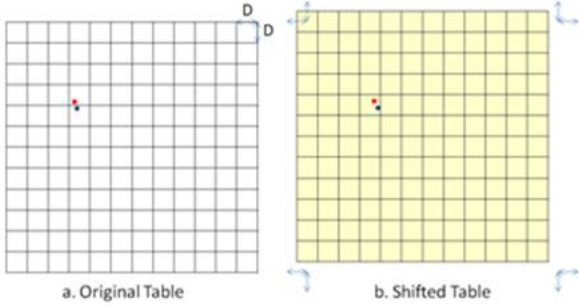


Figure 8. Original Table and Shifted Table

Figure 8. show us how randomly shifted cell can help us to solve the problem of missing case. As we can see even there are 2 dots (represent 2 users) in the original table, which are very close to each other. They have not fallen in the same cell. After add a shifted table, which shift to the left up direction. Both users fall in the same cell again and they will be treated as distance D beside his friend.

$$P = (0.75)^N \quad (1)$$

No. of Mappings(N)	Prob. of missing report(P)
1	75%
2	56.25%
3	42.19%
4	31.64%
5	23.73%
10	5.63%
15	1.34%
20	0.32%

Table 1. Relationship between the probability of missing report and number of layers

Table 1 shows us the relationship between the probability of missing report and number of layers. For a user who require for distance D proximity monitoring, its coverage area of its D distance will be $(2D)^2$. For one cell size $(D)^2$

only cover 25% of the coverage area. That's mean there have 75% chance of missing report. However, the missing rate can be reduced by adding more randomly shifted layer. As we can see when there are 20 layers, the rate of missing is only 0.3% which is only a very small chance.

As a result, we see the usage of the randomly shifted data, which solve the problem of missing report by using only one grid cell.

4.2 System Operation

After the standard is shared, the next step is using those data to achieve our major purpose - secure proximity monitoring. For the ease of illustration, we use only one table for the explanation, which is one of the tables from the bottom grid layer.

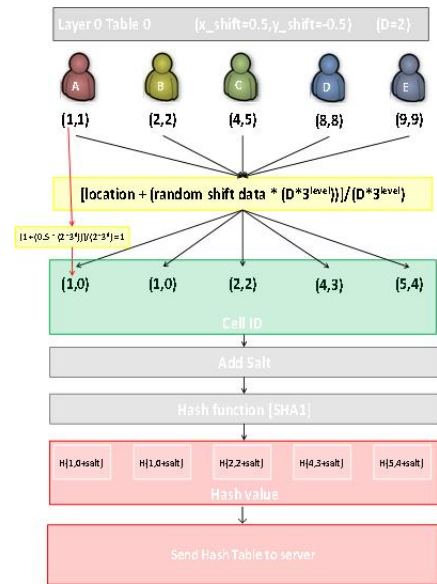


Figure 9. Workflow of System Operation

In Figure 9, we can see there is 5 users A,B,C,D,E. After A share the salt and random shifted data ($x = 0.5, y = -0.5$) in the initialization stage (Figure 7), other users are able to use those information to find cell ID and hide their location under the same standard. As the cell size is $(D \cdot D)$, if two users fall in the same cell, they must be the positive candidate in the proximity monitoring. Then, all user find the cell ID they belong to. The following step users are going to combine their cell ID and the shared salt and undergo the hash function (e.g. SHA1). Finally, all users send their hash value to the server. Then, server checks whether there is same hash value among users. In this case, server find user A,B and user D,E are in the same cell. So server announce user A, B they are nearby each other and hide the location of other users, because they are outside the bound

of proximity monitoring. User D,E will also have the same arrangement as user A,B.

4.3 System Update (Minimize Communication Cost)

It is sure that we finish our work by using one layer of hash values. Nevertheless, we need to check all users hash value frequently, which consume much more resource than we actually require.

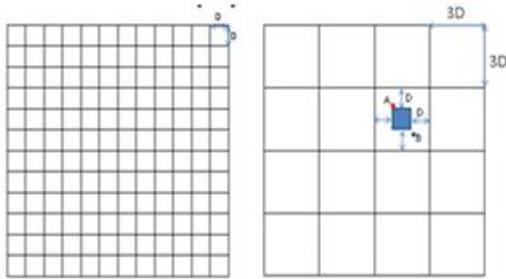


Figure 10. Sample of D^2 Layer

In the Figure 10, we show an example to explain the reason of frequent update, when only one layer has been used. In this figure, it shows us the distance between user A and B is $1.1D$. Although they are quite nearby each other, it is impossible for them to fall into the same grid cell. Therefore they are treated as away from each other. In this case, the problem come if A and B do not update frequently. It is because when there is no immediate reaction when user A and B come within distance, that show the service quality of system is not good enough. On the other hand, other users also require updating frequently, even they are far away from each other. We need some efficient update approach to improve quality of service and reduce communication cost.

There are some solutions [6, 11] suggest to create the layer based map, so as to reduce the communication by including the sparse layer. However, there are some differences between our solution and existing solution.

In existing solutions they use quad-tree structure fixed location lay, our solution use non-tree structured randomly-shifted layers. Attackers are more difficult to find the exact location of the users. Also, our solution hash the location into hash value, rather than directly transfer the grid based location to server.

Figure 11 shows the vertical view of our vertical hash structure. Our system is setup in a bottom-up manner, which starts at the bottom layer (cell size D^2) and end until the whole map is covered by a single cell.

$$Max(L, W) = 3^n D$$

That means the number of layer (n) in the system depends on the proximity distance (D) and the larger one of length (L) or width (W) of the map.

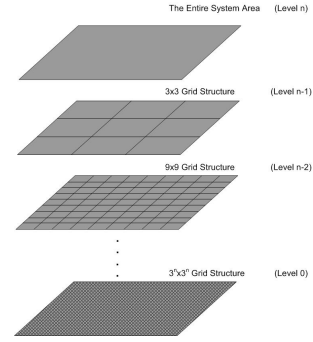


Figure 11. The Non-Tree Structured Grid Based Layer

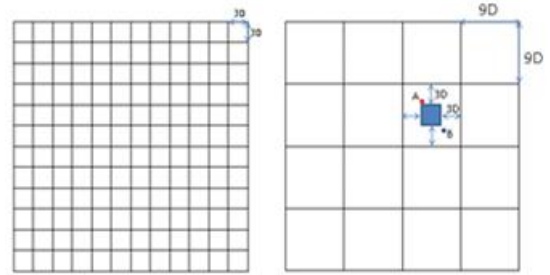


Figure 12. Sample of $(3D)^2$ Layer

We use Figure 10 and 12 as an explanation model of our work. In Figure 10, the distance between user A and B is $1.1D$, no matter how we shift the size D^2 hashed table. They never fall in the same hashed cell. Therefore, we can at least ensure they are at least apart from each other more than distance D . The similar concept can also be used in Figure 12. , the distance between user A and B is $3.1D$, no matter how we shift the size D^2 hashed table. They will never fall in the same hashed cell. Therefore, we can at least ensure they are at least apart from each other more than distance $3D$. The concept of expand layer of Figure 10 and 12 can be expanded. As the cell width and length expand a constant times per layer, so the size is also expand 9 times per layer. According to speed of expansion, a single cell in upper layer can cover a large place. That mean player require longer distance to escape the cell, thus less update is needed.

4.4 System Update in 2 Users Situation

In order to facilitate explanation, we first illustrate the system update of 2 users. After that, we will talk about the system update in practical multiuser situation. Our system do things in the initialization stage, so as to reduce the communication cost in the later stage. Therefore, in the initial-

ization stage server collect all the hashed value from users and users require to update only if necessary. In the following subsection, we will explain how our system work in different situation when there is 2 users.

4.4.1 2 Users Nearby Initially

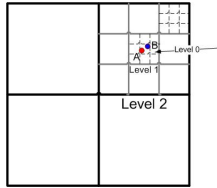


Figure 13. 2 users nearby each other

In this case, user A and B distance are less than D . That's mean they are overlap at the Level 0. So if they are required to do update when they leave any of the mappings in level 0.

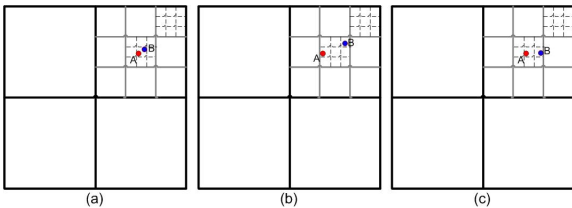


Figure 14. Different situation of 2 users in Time 1

Figure 14 shows us 3 different situation user A and B will face in later stage. Case (a) is user A and B remain same mapping as before. We do not need to do anything because nothing is changed. For the case (b), A and B move away from each other when there is change in the mappings. Update is required to be done, as the change of mapping for both users is only occurring in Level 0. Therefore both user just need to announce the server, there is change in certain layer. Then, both users need to update all layers at or below the certain layer. The solution of case C is similar to case B. The major difference is only user A requires to send the update information to the server. As user A quit the previous mapping while user B is remaining unchange.

4.4.2 2 Users Away Form Each Other Initially

In this case, the distance between user A and B is $7.5 D$. That's mean they only overlap in the higher level - level 2. Both users do not require to update their mappings if there is no change in Level 2. That's mean even there is change in level 1 and level 0 for both users, if the mapping in level

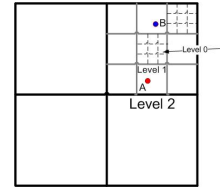


Figure 15. 2 users away from each other

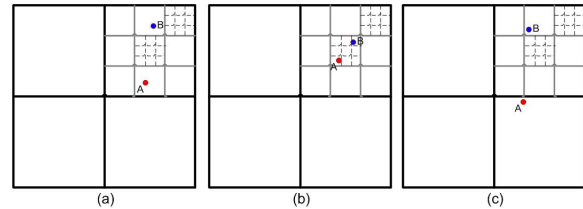


Figure 16. Different situation of 2 users in Time 1

2 is unchanged, no update is needed. So in case (a), both user A and B do not require to do any update. In case (b), both users quit their mappings in level 2, but not quit their mapping in level 3. Therefore we only need to update the mappings at or below level 2. After both users update their mappings to server, server find that both users overlap in a closer level - level 1. Then the server will focus on the update of level 1 of both users just like what is done in level 2 previously. On the contrary, case (c), both users quit their mappings in level 2 and no longer overlap in level 2. Server find their minimum overlap level is level 3. After that server will focus on the update of level 3, if there is no change in mapping of level 3, no update is required.

4.5 System Update in Multiuser Situation

All users have undergone the update process since they start the location monitoring. Multiuser solution is similar to the 2 users version. They are both share all sets of mapping during the system initialization. The major difference of two solutions is in the 2 users version update depend on the minimum overlap level (MOL) of 2 users, while multiuser version focus on a few near neighbor which overlap in lower level.

User\User	1	2	3	4	5
1		0	3	5	6
2	0		4	2	2
3	3	4		5	3
4	5	2	5		4
5	6	2	3	4	

Figure 17. User Overlap Matrix

In the multiuser solution, we construct a user overlap matrix in the server, in order to keep track with the change in mappings. Figure 14 show us a user overlap matrix, if the value in the matrix is 0 that mean that two users have some hash table overlap at level 0 (cell size = D^2), then they must be their distance must within D . The level higher, the distance longer. Minimum overlap level (MOL) is the overlap level of the nearest neighbor. We use MOL as the update check because the lower level is the more likely to change and also more important in proximity monitoring.

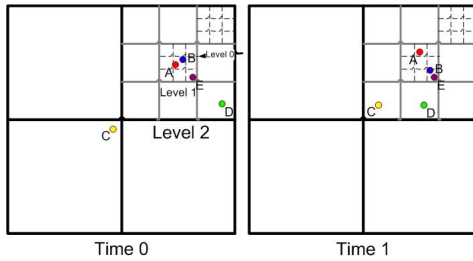


Figure 18. Users Location from Time 0 to 1

User/User	A	B	C	D	E
A		0	3	2	1
B	0		3	2	1
C	3	3		3	3
D	2	2	3		2
E	1	1	3	2	

→

User/User	A	B	C	D	E
A		1	2	2	1
B	1		2	2	0
C	2	2		2	2
D	2	2	2		2
E	1	0	2	2	

Figure 19. Matrix Change from Time 0 to 1

Figure 18 show us the location of user A,B,C,D,E from time 0 to time 1 and figure 19 is the overlap matrix of 5 users, which construct in time 0. Each time user may need to send an update to server, then server use this matrix to determine whether the user is required to update. Therefore even in the worst case only $3N$ communications is required, where N is the number of users.

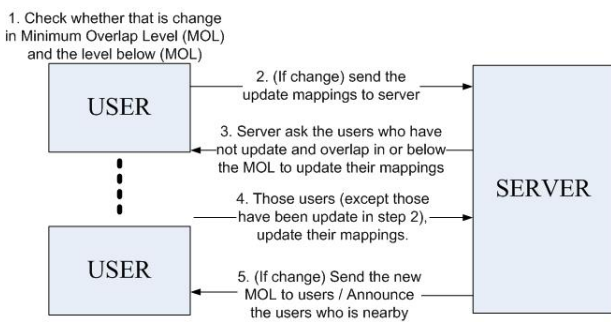


Figure 20. Update Process in Every Period

In figure 20, we have a more detail explanation in our solution and use the users from figure 18 as our sample. All users require to do step 1, check whether there is change in or below the MOL. Check the change of one level below

MOL can help us to detect the other users in the MOL come closer. In our system update process, there are 3 cases with different communication cost will occur in the process.

Firstly, for the worst case which require $3N$ communications occur when overlap with new users in or below the MOL (step 1,3,4,5), just like the case of user E. Even he does not exit his mappings in MOL, but user B overlap with user E in level 0, which is a level lower then the original MOL. Therefore user E also requires to update.

The second case is the user leaves its mapping in or a level below the MOL (step 1, 2,5) just like user A,B,C,D. They exit the mapping of their MOL. Therefore they are all required to update their new mappings to the server. Under the consideration of cost saving, users only transfer the changed level mappings rather than update all mappings blindly.

The third case is the best case which do not require any communication occur when user do not exit any mappings in or one level below the MOL and also no new user entry or old user exit the MOL. Therefore, after (step 1) is finished, it will directly jump to (step 5) and see whether there is change in MOL.

The cost of update of step 2 and 4 is depending on number of levels of mapping change. For example, if there is change in mapping of level 0,1,2 between the update period. Then 3 levels are required to update.

5 EXPERIMENT

Our experiment employ T. Brinkho[17] solution as the base of sample collection. We use the paper[17] provided city Oldenburg as our test case. In order to have a better understand of our system performance, we analyze it in three aspect number of users, proximity distance (D) and effect of speed. For the size of social group, we use (10, 20, 40, 80) as our test case. It is a reasonable size of a social group, while we can also observe the relationship between the number of users and the cost clearly. For the proximity distance (D), according to M.Gruteser and D. Grunwald [1] suggestion, they propose 100m is a suitable segment distance for Driving Conditions Monitoring. So we use (25m, 50m, 100m, 200m) as our test cases and handle the case of both walking and driving monitoring. Not only concern on different need of groups, but also the difference of performance in various speed.

The graphs in figure 21 show us, how the performance of using the multi-layer grid is based layer approach instead of the single layer approach. For the single layer version, we require to update frequently, because if one the hashed value change, we need to do an immediate update, so as to preserve the data accuracy. Just like the situation we have mentioned in Section 4.3. It is different from the multi-layer version, which separate users into sparse level. From the

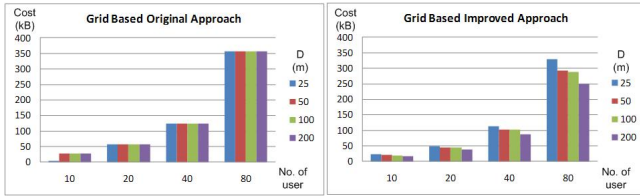


Figure 21. Comparison between Single-Layer and Multi-Layer Grid Based Approach in Average Communication Cost of Server in a Second

above graph, it shows us the performance is similar if the proximity distance (D) is small. However, proximity distance (D) larger, more cost can be save in the multi-layer approach. Normally, if proximity distance (D) is larger, users have higher chance to overlap. Thus communication cost is also higher. However, in our system the relationship between proximity distance (D) and communication cost is reversed. For $D = 25$, we use 7 layers to cover the whole map, but when $D = 200$, we only require 5 layers to do this. Multi-user approach also performs better in more users. So multi-user approach is more suitable in practical application.

In order to have a better understanding of our system performance, we make a comparison between our system and a native cloaking method.

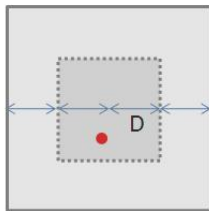


Figure 22. Cloaking Region of the Native Approach

Figure 22 show the structure of the native approach. First, we randomly generate a size $(2D)^2$ rectangle which contain the user's current location. Based on this rectangle, we will extend the size of the cloaking region to a size $(4D)^2$ rectangle, according to the center of the original $2D^2$ rectangle. When the system starts, all users send their cloaking region to the server. Then, server checks whether they is overlap in users' cloaking region. For the users who are nearby, we will continue ask the user to generate a new cloaking region everytime of update, until they are no longer nearby each other. For the users who are not nearby, server ask the user to update cloaking region, only if they have exit the original $(2D)^2$ area, otherwise no update

is required.

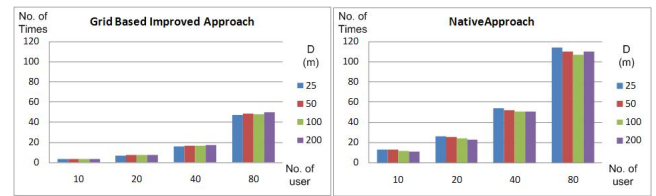


Figure 23. Comparison in Times of Location Update

Graphs in Figure 23 show number of times of location update required within a minutes for different number of users. Our system require less update than the native approach, because of the contribution of the higher level sparse layer. Users which have high minimum overlap level are far from other users. Therefore they need less update, because they require longer time to move out a grid based cell comparing with the lower level cases.

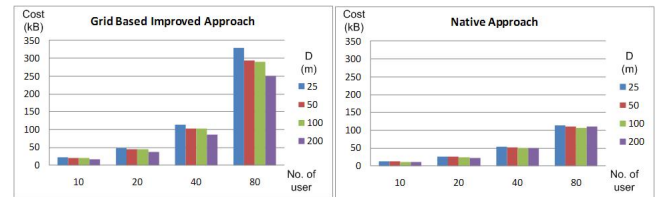


Figure 24. Comparison between Grid Based and Native in Average Communication Cost of Server in a Second

Despite less update is required. The communication cost of grid-based approach is still higher. It is because native approach only need to send the center of its cloaking region, while our system require sets of hash value for secure proximity monitoring. One time communication cost of native approach is around 1000 Bytes, including the communication overhead, but grid-based approach much more. Therefore even our system requires much less update, the cost of our system is still much higher than the native approach. However, the cost can be reduced by adjusting the number of layers and numbers of hash values per layer, but it is clearly a tradeoff between service quality and the load of communication cost.

The following graph show us the communication cost in one minute. Communication cost of the native approach increase when the speed is increasing and grid-based approach do not have direct relationship between communication cost and speed. Even the users are moving very fast. In grid-based approach, they require longer time for users to leave a large grid cell. On the contrary, for the native ap-

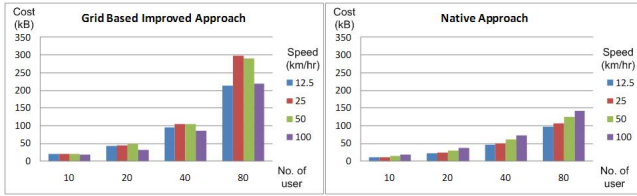


Figure 25. Comparison between Our Approach and Native in Different Speed

proach as the speed of the users increase, they always leave the cloaking region and overlap with new user. Therefore the communication cost of native approach will continue to increase and stop only when it reaches the equilibrium (every user required to update every time).

Although our approach requires more communication cost, it outperforms the native approach in better accuracy.

GRID BASED APPROACH				NATIVE APPROACH			
		Distance within D				Distance within D	
		T	F			T	F
System	T	232904	0	System	T	231724	12548
Display	F	336	7373512	Display	F	1516	7360964

Figure 26. Comparison between the Report Accuracy

Our experiment do more than 7 million comparisons in case of different number of users, speed and proximity distance (D). We come out the result, which show in Figure 26. Grid-based approach is more accurate than Native approach. Grid-based approach never gives wrong signal when user is not within the proximity distance D and gives less missing signal when there is user nearby.

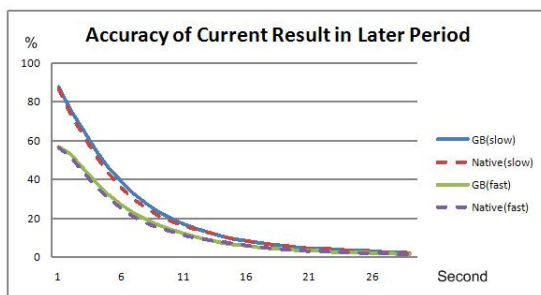


Figure 27. Accuracy of Current Result in Later Period

Figure 27 show us the degradation of accuracy for different approach in different speed by using current result in later period. In this experiment each period time is set as

1 second. We see that the performance of both approach is similar and grid based approach perform a bit better. The fast speed is set at 100km/hr slow speed is set at 12.5km/h. The result shows us fast speed have lower accuracy, especially in the easier stage.

6 CONCLUSION

In this paper, we investigate the problem of secure proximity monitoring. We propose a solution, the worst case of the solution is $O(n^2)$. Our solution can completely protect the users' location privacy with a reasonable cost.

For the future work, for communication cost we think there is still some room of improvement. For example developing some more effective location transformation. Then communication cost can be reduced.

References

- [1] M.Gruteser and D. Grunwald, "Anonymous usage of location-based service through spatial and temporal cloaking," Proc. Of the International Conference on Mobile Systems, Applications, and Services (MobiSys'03), pp163-168, Scan Francisco, USA, 2003
- [2] Hidetoshi Kido, Yutaka Yanagisawa, Tetsuji Satoh, An Anonymous Communication Technique using Dummies for Location-based Services, Pervasive Services, 2005. ICPS '05. Proceedings. International Conference
- [3] T. Xu and Y. Cai. Location Anonymity in Continuous Location-based Services. In ACM GIS'07, pages 300–307, November 2007.
- [4] Xian Pan, Jianliang Xu, Xiaofeng Meng, Protecting location privacy against location-dependent attack in mobile services, Proceeding of the 17th ACM conference on Information and knowledge management, 2007
- [5] A. Khoshgozaran and C. Shahabi. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In Proc. SSTD, 2007.
- [6] M. F. Mokbel, C. Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In Proc. of VLDB, 2006.
- [7] A.C. Yao. Protocols for secure computations. In Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982.
- [8] A.C. Yao How to generate and exchange secrets. In Proceedings 27th IEEE Symposium on Foundations of Computer Science.

- [9] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: Anonymizers are not necessary," in Proc. ACM SIGMOD Int. Conf. Manage. Data, Vancouver, Canada, Jun. 2008, pp. 121-132.
- [10] P. Ruppel, G. Treu, A. Kpper, and C. Linnhoff-Popien, "Anonymous User Tracking for Location-Based Community Services," in LoCA, 2006, pp. 116-133.
- [11] K. Liu, C. Giannella, and H. Kargupta, An Attackers View of Distance Preserving Maps for Privacy Preserving Data Mining, in PKDD, 2006, pp. 297308.
- [12] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, and S. Jajodia, Privacy-aware proximity based services, in MDM, 2009, pp. 3140.
- [13] L. Łiknys, J. R. Thomsen, S. Łaltenis, M. L. Yiu, and O. Andersen, A Location Privacy Aware Friend Locator, in SSTD, 2009, pp. 405410.
- [14] L. Łiknys, J. R. Thomsen, S. Łaltenis, M. L. Yiu, Private and Flexible Proximity Detection in Mobile Social Networks, Proceedings of the 11th International Conference on Mobile Data Management (MDM), Kansas City, Missouri, May 2010.
- [15] Artak Amirbekyan and Vladimir Estivill-Castro. Privacy-preserving k-nn for small and large data sets. In Proceedings of the ICDM Workshops, 2007.
- [16] Processing Private Queries over Private and Indexed Data
- [17] T. Brinkho. A Framework for Generating Network-Based Moving Objects. *GeoInformatica*, 6(2):153C180, 2002.