A Privacy Preserving System for Friend Locator Applications

Bin Zan
WINLAB, Rutgers University
671 Route 1 South
North Brunswick, NJ
08902-3390
zanb@winlab.rutgers.edu

Tingting Sun
WINLAB, Rutgers University
671 Route 1 South
North Brunswick, NJ
08902-3390
sunting@winlab.rutgers.edu

Marco Gruteser
WINLAB, Rutgers University
671 Route 1 South
North Brunswick, NJ
08902-3390
gruteser@winlab.rutgers.edu

Fei Hu
ECE, The University of Alabama
101 Houser Hall
Tuscaloosa, AL, 35487-0286
fei@eng.ua.edu

Yanyong Zhang
WINLAB, Rutgers University
671 Route 1 South
North Brunswick, NJ
08902-3390
yyzhang@winlab.rutgers.edu

ABSTRACT

One interesting application of online social networks is the friend locator, in which the application server informs users through mobile devices if their listed friends are nearby in terms of geographical locations. However, in such services, it is challenging to protect the privacy of an individual user. Previous privacy protection solutions for friend locators do not guarantee a high level of privacy and maintain efficiency. In this paper, we propose a privacy preserving system to guarantee both strong privacy and efficiency. Additionally, we use the polygon decomposition method to achieve both accuracy and flexibility especially for irregular areas of interest. Finally, through numerical analysis and simulation, we show that the proposed system and algorithm can achieve high privacy, efficiency, accuracy and flexibility.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*;
K.4.1 [**COMPUTERS AND SOCIETY**]: Public Policy Issues—*Privacy*

General Terms

Algorithms, Performance, Security

Keywords

Friend Locator, Privacy Preserving, Convex Polygon

MobiWac'11, October 31–November 4, 2011, Miami, Florida, USA.

1. INTRODUCTION

A friend locator is a location-aware social network application, in which users are notified if any of their friends is geographically close by. In such an application, users with a GPS enabled mobile device will periodically update their current locations to a central server. By comparing a user's area of interest with the friends' locations, the server can determine if any friend is close by, and then inform this user. Users have to trust the central server and disclose their own locations to the server first. However, this is sometimes in contradiction with privacy requirements. A user's location information is often associated with sensitive personal information. For example, frequent visits to a dental office by a user may indicate his/her dental health issues. Or, if a medical insurance company knows how often a user visits fast food restaurants, it may raise the rate accordingly.

Prior work on preserving privacy of general location based services (LBS) do not fit friend locator applications very well. For example, the k-anonymity method [7, 4, 14], which protects a user's location information by mixing it with $k-1$ other users' location information, is well-suited to location based queries, such as point of interest (POI) queries. However, it does not work in a friend locator, which requires user identities. Location obfuscation would result in incorrect judgement on whether or not two users are adjacent. Some specific algorithms also have been developed for friend locator privacy preservation [17, 10, 19, 11]. However, Šikšnys et al. [17] and Mascetti et al. [11] achieve privacy by sacrificing the accuracy of the results. Manweiler et al.'s method [10] is not so efficient. In addition to that, to the best of our knowledge, none of the previous solutions can handle area of interest with irregular shapes as in our proposed algorithm, and none of the existing approaches can guarantee privacy protection when the server and the other users are collaborating.

In this paper, we develop a new privacy preserving system and algorithm to achieve high privacy, accuracy, efficiency and flexibility for friend locator applications. To summarize, the main contributions of this paper are:

1. On Efficiency: Reducing the overall overhead by split-

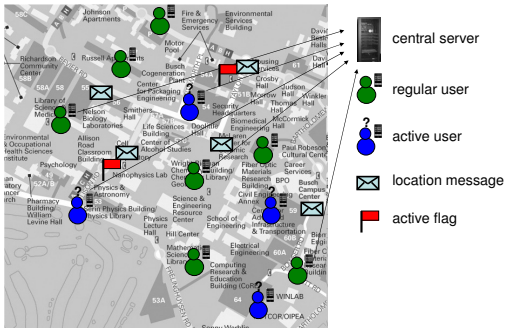

Figure 1: System Model

ting the friend locator task into two parts – a coarse and a fine resolution phase. After removing a large amount of friends during the coarse phase, only a small number of users are left to the fine phase, which has higher complexity. In the coarse resolution phase, we also exploit a multi-level grid, variable-length bit sequences, and differential permutation encryption methods to further reduce the overhead.

2. On Privacy: Exploiting entropy-based splitting to increase the uncertainty of each location index which prevents statistical attacks based on the knowledge of the population distribution. In the fine resolution phase, we exploit a property of linear operations. This allows the system to successfully hide individual user's privacy from the server or other users, while still being able to determine if a friend is inside an active user's area of interest (over a multiple-convex-polygon representation of the area).

3. On Accuracy and Flexibility: Providing a fine-resolution level for the active user to specify an area of interest in detail. Furthermore, representing each active user's area of interest through multiple convex polygons. Although this step is the basic part for the privacy mechanism, it also increases the accuracy and the flexibility of friend locator applications compared to those using Euclidean distance only methods (especially for irregular-shape cases).

This paper is organized as follows. Section 2 describes the system model and problem statement. Section 3 provides the details of the proposed privacy preserving algorithm. Section 4 evaluates the scheme through numerical analysis and simulation. Section 5 discusses related work. Section 6 concludes the paper.

2. SYSTEM MODEL AND PROBLEM STATEMENT

2.1 System Model

As shown in Fig. 1, we assume the friend locator system is formed by three parties: A central server, regular users and active users (user with question mark on head). All the users carrying a mobile device (for example cellphone) are called regular users. We assume users are communicating in infrastructure mode. Users periodically update their location information to the server and the server stores the location information in its database. Active users are the ones who are currently interested in finding their nearby friends. When a user wants to become an active user, the user sends the server an active flag and the updated location information. The server will either help identify the nearby friends for the user or start a procedure to assist the user to find friends inside the area of interest. The area of interest is defined as the geographical area in which a user would like to monitor friend positions. We assume that this area surrounds the user but the shape of the area and the user's relative position inside the area can be arbitrary. For example, the area of interest can be an irregular shaped floor plan of a shopping mall, and the user, while shopping inside a random store, is interested in knowing which friends are inside the same mall.

2.2 Problem Statement

The biggest challenge we face is to keep individual users' exact locations unknown from the central server and other users, without affecting the execution of friend locator applications. Privacy is a general requirement for human being's daily life. Sensitive location information can disclose many secrets of a particular user. For example, a female user's visits to a maternity clinic over several months may indicate pregnancy. Thus, our goal is to keep the exact location unknown from the central server. Otherwise, once an adversary compromises the central server, it can derive much private information from the location data of a particular user. However, for friend locator applications, the server has to collect users' location information in order to identify if two users are close to each other.

Zhong et al. [19] have proposed to use homomorphic encryption method to resolve this conflict. However, it is not flexible enough since this method can not handle an irregular area of interest. For example, a user would not be able to accurately define a shopping mall as an area of interest. Due to the large overhead introduced in a short time period, the idea of bypassing the central server and letting users always communicate peer-to-peer to determine if they are nearby is not practicable. Furthermore, in such case, none of previous solutions can prevent the attack from one of the users. Finally, using location information obfuscation alone does not work for users who want to have higher level of accuracy.

In summary, we target on the following problems and objectives: 1) **Privacy:** accurate location information of a user should not be directly known by the central server. The central server is assumed to be untrustworthy. The privacy should be preserved under passive attacks from the server. 2) **Efficiency:** the system should try its best to shift computation overhead to the central server. In other words, minimize the computation overhead on each individual user. 3) **Accuracy:** the system should meet a user's requirement of accuracy. For example, if a user wants to find all friends in the area of 5 square miles, we should not include the ones who are 20 miles away. 4) **Flexibility:** the system should provide active users flexibility in choosing their own areas of interest, which means a user might want to know if a nearby friend is in a particular area. Last but not the least, for a user who wants to have high accuracy and strong privacy protection, this system should provide an ultimate solution that even eavesdropping on the information from both the server and the friends of this user, an adversary is still unable to figure out the exact location.

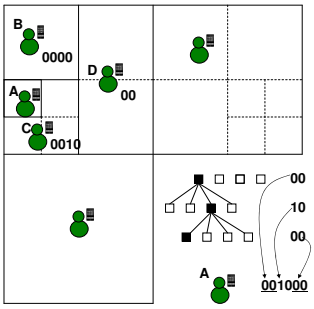

Figure 2: Multi-level splitting based on Quadtree. Users can choose their own privacy preferences, which are also corresponding to the depth of the tree.

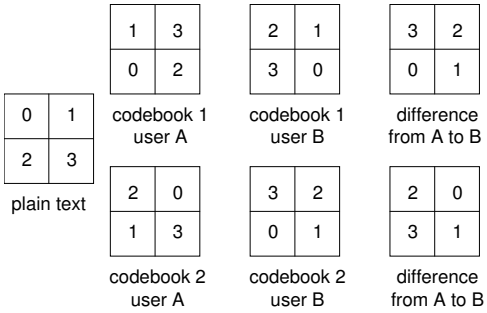

Figure 3: The codebooks of user A and B in level 1 (first row) and 2 (second row) and their difference. The codebook of a user indicates the encrypted values the user will generate for real location indexes. For example, a real location index 1 will be encrypted as 3 by user A and 1 by user B in level 1. Thus, if the server sees an encrypted value of 3 from user A, unless it also sees an encrypted value of 1 from B, otherwise the original indexes from A and B are different in level 1.

In this proposed work, we assume users are authenticated through a third party method. The adversary is not interested in spoofing attacks by masquerading as a friend of the target user. The authentication of a user's location is not the focus of this paper since we target on information confidentiality issues.

3. THE PRIVACY PRESERVING ALGORITHM

In this section, we describe the proposed Privacy Preserving Algorithm (PPA) in details. The PPA algorithm includes two phases. In phase one, the server roughly estimates if two users are close to each other by comparing their location information in terms of location bit sequences. In phase two, after filtering all the friends who are not in the same geographical block, the active user further determines if a particular friend is inside the area of interest by providing a special convex polygon representation of the area.

3.1 Phase One: Quadtree Based Multi-Level Splitting

We split the full map into multiple levels according to a Quadtree structure. The location information of a user is represented by a bit sequence which can be obtained through the tree. As an example, in Fig. 2, user A discloses three levels of the location information: level 1 (block 00), level 2 (block 10), level 3 (block 00). The length of a location bit sequence (equivalent to the depth in the tree) is defined according to the user's own privacy requirement. The longer the bit sequence is, the more accurate the location information is. This also indicates how often a user wants to be checked by friends. A short bit sequence results in more chances to be checked by others. In above example, user D only gives top level location information. Thus, if any of the users A, B and C wants to figure out if D is nearby, D has to disclose more information. On the other hand, the information given by user B is already enough for user A and C to determine that they are not nearby and vice versa.

A user periodically updates the location information to the server. In addition to that, the bit sequence is encoded by a secret key which permutes the block numbers (similar to a Block Cipher [9]) at each level. This key is only shared between a user and the friends. To find if any friend is nearby, a user sends multiple copies of the location information to the server. Each copy is encoded with a special key agreed with a friend. The server compares the pair of location bit sequences to identify if two users are in the same block. This achieves similar functionality as a Homomorphic Encryption [16] does.

Increase Efficiency through Differential Permutation: To improve the efficiency, we propose an enhanced scheme for comparison of two bit sequences. Recall in original scheme, for every friend, a user has to provide a copy of the location bit sequence with special encryption through the common key agreed by that friend. If a user has a large number of friends, this clearly creates a lot of overhead. To reduce the overhead, we propose a differential method. The idea is to let each user encrypt (permute) the bit sequence using the user's own secret key. While every pair of friends publish the distance of their secret keys. After knowing the "difference", the server could transform the permuted location bit sequence of a user and make it comparable with another user's location bit sequence.

For example, the first two-level bit sequence of user A, 0010 (0,2), as shown in Fig. 2 can be encrypted by user A through codebooks of A in Fig. 3, and becomes 0101 ($0 \rightarrow 1$ and $2 \rightarrow 1$). The bit sequence, 0000 (0,0), of user B becomes 1011 ($0 \rightarrow 2$ and $0 \rightarrow 3$). The encrypted value 0101 and 1011 received by the server can not be compared directly. However, after converting 0101 into 1000 ($1 \rightarrow 2$ and $1 \rightarrow 0$) according to the codebook difference tables, the server can identify that user A and user B have the same top level location, however from level 2, they are at different blocks. Without knowing codebook of A or codebook of B, the server can not figure out the real location index of user A or B.

Entropy-Based Splitting: In most previous work, when a map is divided into several blocks, the cutting is only mandatory based on geographical information. Even though each user's location bit sequence is encrypted with secret key unknown by the server, if the server links all the users from the same block and does some statistical analysis, it is possible to find out the real block indexes or weaken the strength of the privacy protection scheme. Consider the following example.

Assume the adversary is the server itself. At a certain

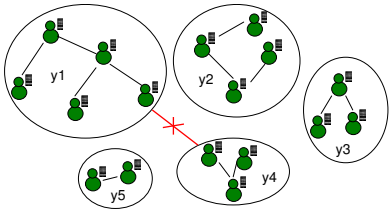

Figure 4: Illustration of the population attack. The icons shown in each group represent the rough population size.

level, the general population in each block is (1, 10, 2, 2) millions known by the server. As shown in Fig. 4, after the server links all users in the same blocks, it can form five groups (y_1, y_2, y_3, y_4 and y_5). Some groups may be able to be combined again, and some may not. For example, because some users in group y_1 and y_4 are definitely not in the same block, these two groups can not be combined again. By finding the optimal way to allocate each group into a block, some of the users' real block indexes can be disclosed with high probability. For example, it is reasonable to consider group y_1 and y_2 actually represent block 2, y_5 represents block 1, and y_3, y_4 represents either block 3 or 4, respectively. Therefore, if a user is in y_1, the server has high confidence to believe the real index number is 2.

We also observe that if the block size in terms of population is divided evenly, even though the optimal matching is perfect, the adversary would not know the right index value. As in Fig. 4, even it is known that one of the groups y_3 and y_4 is actually in block 3, and the other is in block 4, it is indistinguishable which one should be in block 3 or 4 exactly. When the population sizes of blocks are close, it is hard to distinguish them. This can be understood easily by using entropy to represent the whole system's uncertainty as:

$$H = -\sum_{i=1}^{m} p(x_i) \log p(x_i) \tag{1}$$

Where m is the total number of blocks in a level, $p(x_i)$ is the ratio of average population within block x_i compared to the average total population in this level. Clearly, H is maximized when $p(x_1) = p(x_2) = ... = p(x_m)$. It also indicates that we should divide the map into blocks based on population size.

Since phase one only uses location bit sequence of multi-level grid, it does not provide high accuracy for friend locator applications. To achieve higher accuracy, the nearby users need to enter the second step, the fine resolution phase.

3.2 Phase Two: Privacy and Polygon Decomposition

After phase one, it is expected that only a small number of friends from a user's friend list will be kept for further exploration. The friends in different blocks from the user are no longer needed to be involved.

Representing Area of Interest by Convex Polygons: The active user provides the remaining friends a special set of convex polygons which represent the area of interest[1]. Then, the friend locator problem can be represented

[1]To prevent attacking from multiple friends and server's collaboration, we require the active user provides different set of convex polygons and slightly different area of interest for each user.

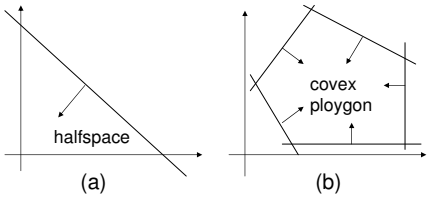

Figure 5: Illustration of the concept of halfspace and convex polygon.

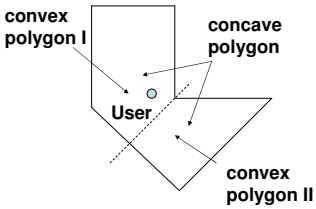

Figure 6: Area of interest represented by a polygon.

by a matrix and solved through linear operations. While this conversion is originally for privacy purpose, it also increases accuracy and flexibility when the area of interest of an active user is in irregular shape.

We first introduce some basic definitions and concepts.

(**Halfspace**) A halfspace in $\mathbb{R}^n$ is a set of the form

$$H = \left\{ x \in \mathbb{R}^n : \quad p^T x \leq \alpha \right\} \tag{2}$$

where $p \in \mathbb{R}^n$ is a fixed non-zero vector, and α is a fixed real number. Fig.5 (a) gives a halfspace example in two-dimensional space.

(**Convex Polygon**) A convex polygon is the intersection of a finite number of halfspaces. It can be defined as:

$$P = \left\{ x \in \mathbb{R}^2 : \quad Ax \geq b \right\} \tag{3}$$

Fig.5 (b) shows an example of convex polygon.

If $a_i \in \mathbb{R}^2$ represents the i-th row of A, $i = 1, \ldots, m$, then we can represent P as

$$P = \left\{ x \in \mathbb{R}^2 : \quad a_i^T x \geq b_i, \quad i = 1, \ldots, m \right\} \tag{4}$$

We represent the area of interest of a user as a polygon. As can be seen in Fig. 6, a polygon is either a convex polygon itself, or it can be divided into a set of convex polygons. Thus, we can write $P = P^1 \cup P^2 \cup ... \cup P^z$, and

$$P = \Big\{ x \in \mathbb{R}^2 : (A^1 x \geq b^1) \vee (A^2 x \geq b^2) \vee ... \\ ... \vee (A^z x \geq b^z) \Big\} \tag{5}$$

Note, for arc-shaped area, we could also use polygon to imitate it.

Preserving Privacy through Matrix Operation: After representing area of interest as a set of z convex polygons, we consider a user B's location $x = (x_1, x_2)$ is inside user A's area of interest if x is a feasible point of P. Without privacy

consideration, thus, user A needs submit a set of matrices A^i and vectors b^i to the location service provider. Then the server computes $A^i x$ and compares the results with b^i.

Obviously, for privacy concern, the exact information of matrices A^i or vectors b^i as well as the location (x_1, x_2) should not be disclosed at the server side. On the other hand, simply bypassing the server is not the solution we are satisfied with since a user's location information will be disclosed to the friends. One common method to hide information is transformation. Specifically, let both A and B transform their areas of interest information or location information into a different coordinate system which is only known by A and B themselves. For example, transforming A^i into $\tilde{A}^i = A^i H$ and x into $\tilde{x} = H^T x$. However, with the new $\tilde{A}^i$ and $\tilde{x}$, user A must provide the server a new $\tilde{b}^i$ accordingly to compare with $\tilde{A}^i\tilde{x} = A^i H H^T x$. An alternative is using $\tilde{x} = H^{-1}x$, and $\tilde{A}^i\tilde{x} = A^i H H^{-1} x = A^i x$. Through this method, the information of matrix A^i and x are not revealed to the server, but the server can still compute $A^i x$ and compare it with b^i. This solution is much simpler than the Homomorphic Encryption method used in [19], and user A has a higher flexibility in choosing the area of interest instead of just using circumcircle.

Achieving Strong Privacy: When the central server cooperates with one of the users, neither the above method nor the Homomorphic Encryption method can protect privacy of the second user. For example, with available information, H, at user A, and, $\tilde{x}$, at the central server, an adversary can figure out the location information of user B through $x = H\tilde{x}$.

This problem can be reduced to a secure multi-party computation problem [18, 5, 3]. There are methods to solve such problem as early as Yao's millionaire problem [18]. However, considering their complexity, many of them are impractical for friend locator applications. In the proposed algorithm, we develop a method which is inspired by [3]. In this method, we only disclose partial data in A^i by user A and x by user B. Specifically, at user A, we have

$$\begin{aligned}\tilde{A}^i &= A^i H = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ . & . \\ . & . \\ . & . \\ a_{m1} & a_{m2} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \\ &= \begin{bmatrix} h_{11}a_{11} + h_{21}a_{12} & h_{12}a_{11} + h_{22}a_{12} \\ h_{11}a_{21} + h_{21}a_{22} & h_{12}a_{21} + h_{22}a_{22} \\ . & . \\ . & . \\ . & . \\ h_{11}a_{m1} + h_{21}a_{m2} & h_{12}a_{m1} + h_{22}a_{m2} \end{bmatrix} \end{aligned} \quad (6)$$

and at user B, we have

$$\begin{aligned}\tilde{x} &= H^{-1}x = c \begin{bmatrix} h_{22} & -h_{12} \\ -h_{21} & h_{11} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \begin{bmatrix} c(h_{22}x_1 - h_{12}x_2) \\ c(-h_{21}x_1 + h_{11}x_2) \end{bmatrix} \end{aligned} \quad (7)$$

where $c = \frac{1}{h_{11}h_{22} - h_{12}h_{21}}$. It can be seen that given only one column of matrix $\tilde{A}^i$ or one row of vector $\tilde{x}$, the additional knowledge of matrix H can't help an adversary to obtain the value of A^i or x. Therefore, let user A and B exchange partial of their own information through the server. A sends

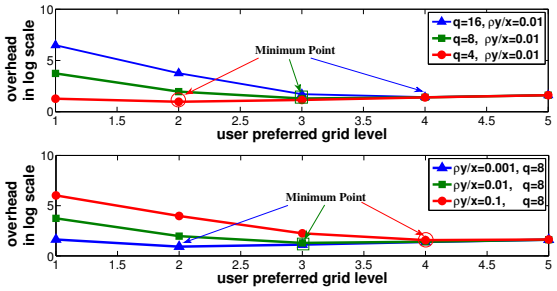

Figure 7: For a regular user, a low grid level results in small communication overhead in each updating period. However, it increases the chance to be selected to enter phase two.

$\tilde{A}^i_{*1}$, the first column of matrix $\tilde{A}^i$, to B, and B sends $\tilde{x}_2$, the second row of vector $\tilde{x}$, to A. Next, user A computes $U^i = \tilde{A}^i_{*2}\tilde{x}_2$ and user B computes $V^i = \tilde{A}^i_{*1}\tilde{x}_1$. Since

$$A^i x = \tilde{A}^i\tilde{x} = \tilde{A}^i_{*1}\tilde{x}_1 + \tilde{A}^i_{*2}\tilde{x}_2 = U^i + V^i \geq b^i \quad (8)$$

$$\Rightarrow (U^i - b^i) + V^i \geq 0 \quad (9)$$

if user A sends the results of $U^i - b^i$ to B, B can add V^i and compare with 0 to determine if B is inside the convex polygon of P^i or not. Finally user B informs A the result. During the whole process, the vector b^i should be kept secure at user A's side only, while the invertible matrix H can be known to everyone.

Assume user A and the central server are cooperating. During the whole process, user B only discloses the value of $\tilde{x}_2$, which by it alone is not enough to solve a two-variable system. Therefore, user B's privacy is preserved. On the other hand, user A discloses $\tilde{A}^i_{*1}$ and $U^i - b^i$ during the whole process. These information constructs a $2m^i$-equation system with $3m^i$ unknown variables, which, in general, has no unique solution. To conclude, through the above method we can achieve a strong privacy in phase two.

4. EVALUATION

The evaluation of the proposed system includes numerical analysis and simulation. In the numerical analysis part, we study the tradeoff on overhead between low and high grid levels and then compare the efficiency achieved by differential permutation against a non-differential method. In the simulation part, we study the impact of entropy based splitting on privacy, the advantage on communication overhead by using two-phase algorithm, and finally the improvement on accuracy through the polygon decomposition method compared to a baseline scheme.

4.1 Numerical Analysis

First, we study the advantage of using variable-length bit sequences. We can generalize the communication overhead for a regular user during a location update period as:

$$O = l * x + q^{L-l} * \rho * y \quad (10)$$

where l is the specific grid level the user preferred. x is the number of bits each level needs (for quadtree it is 2). L is the total number of levels. ρ is the probability a user will be checked by friends who are in the same block of the lowest level during the update period. q is the increasing on

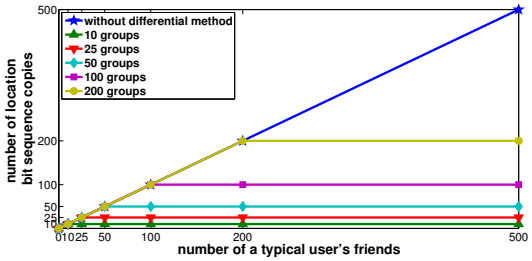

Figure 8: The number of copies required when the number of groups is fixed.

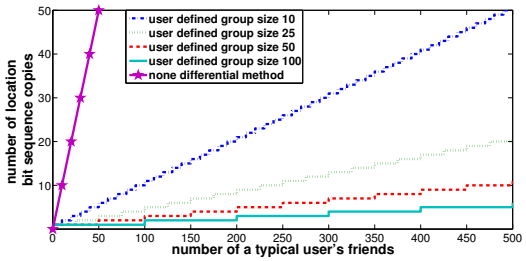

Figure 9: The number of copies required when the size of a group is fixed.

probability a user will be selected to enter phase two when the preferred grid level decreases by one level. y is the communication overhead for phase two operation. In Fig. 7, we assume $L = 5$. It can be seen from the first subplot, while fixing the value of $\rho y/x$, the smaller the value of q, the more communication overhead we save while lowering the level to a more coarse one. On the other hand, as shown in the second subplot, while fixing the value of q, the smaller the ratio of $\rho y/x$, the more coarse level is preferred in terms of reducing communication overhead. Finally, this could be formulated as an integer programming problem. By first solving the continuous relaxation version, we can get its optimum value at:

$$l = L - \frac{ln(\frac{x}{\rho y}) - ln(ln(q))}{ln(q)} \tag{11}$$

and further determine the integer solution.

Next, we study the advantage of using differential permutation. As we know, through differential permutation, regular users can avoid uploading multiple copies of their location bit sequences. The only disadvantage of this method is that if one user leaks a secret key, the server or adversary can also break the friends' secret keys through it. Therefore, we offer two methods based on differential permutation to avoid ripple effect. First, we allow a user to have fixed number of groups of friends. For each group, the user will use the same secret key, in other words, the same permutation. Based on user's own preference, friends are assigned to different groups. Second, we allow a user to have groups of fixed size. When a group is full, the user starts a new group and gives a new key to that group. As shown in Fig. 8, while we fix the number of groups, the overhead is first increasing as the non-differential method, and until the user has all groups activated, then the number of copies stays in the same level. In Fig. 9, when the size of each group is fixed,

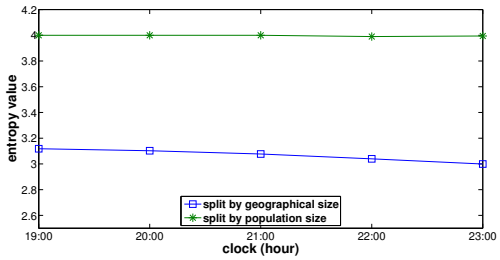

Figure 10: The uncertainty is increased by splitting the map based on population.

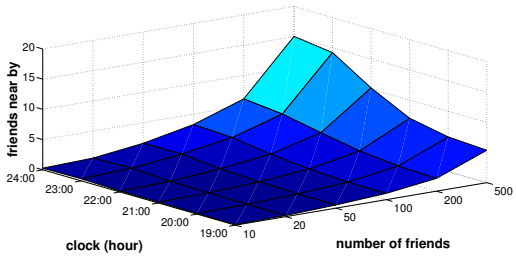

Figure 11: The maximum depth of the quadtree is 4. The number of friends each user has varies from 10 to 500. As can be seen, in average, a user has much less close by friends than total number of friends. Therefore, a two-phase design is necessary and it can dramatically reduce the overhead of each user.

the overhead will keep increasing, however, the speed is far lower than the non-differential method.

4.2 Simulation

In this subsection, we further study the proposed algorithm through simulation. The data set of users' movements is obtained through the MilanoByNight simulation [13] by EveryWare Laboratory from university of degli Studi di, Milan, Italy. It represents 100,000 users' movement in the city of Milan during a weekend night. The whole data set is collected over 5 hours, and locations are sampled every 20 seconds. The total size of the map is 174 $mile^2$ and the average density is 572 $users/mile^2$.

In the first experiment, we assume the maximum depth of the quadtree is 2. The 16 leaf blocks are obtained in two different ways. First, we use traditional method which divides the map based on geographical size. Second, we divide the map based on the proposed entropy-based splitting scheme. As shown in Fig. 10, by splitting the maps based on the population, the entropy value increases from 3 to 4 over the simulation period.

Next, we estimate the communication overhead. We assume each active user has a number of friends varying from 10 to 500, and we assume the maximum depth of the quadtree is 4. As shown in Fig. 11, after phase one, the proposed algorithm eliminates most friends of a user, and dramatically reduces the overhead otherwise will appear in the phase two. Most of the time, the average number of friends nearby is less than 5. Even when the average number of friends of a user is 500, the simulation data still shows maximum 15 friends nearby. Following this result, we further show an ac-

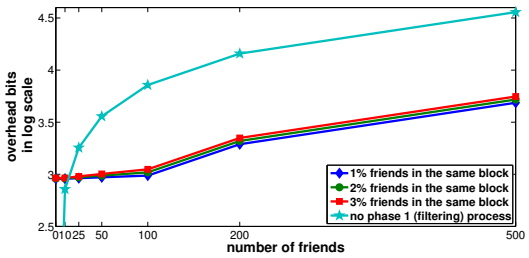

Figure 12: An active user's total communication overhead during a 5-hour period.

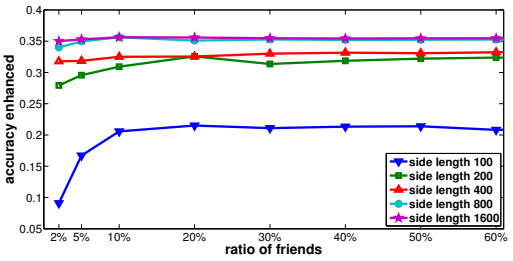

Figure 13: By using polygon based method, the accuracy of friend locator applications is improved. When the size of the area of interest and/or the density of friends increases, the enhancement increases first and then becomes stable.

tive user's total communication overhead during the 5-hour period in Fig. 12. We assume each user updates the location every 20 seconds and uses the most accurate level, which is 4 in this case. The length of location bit sequence is 1 byte for everyone. We also assume for every 100 friends, a user uses a special secret key and the active user's area of interest is a rectangle. According to the proposed algorithm, the communication overhead for the active user with every friend in the same block is 72 bytes[2]. As can be seen from the figure, by using two phases, we can dramatically reduce the communication overhead for an active user except when a user has a very small amount of friends, because at that time, periodically uploading location information occupies the most communication overhead.

Next, we show the improvement on accuracy by using the proposed polygon decomposition method compared to previous work Pierre [19] which is based on Homomorphic Encryption. The homomorphic encryption method can be used to determine if two users have Euclidean distance less than a pre-defined value. In the simulation, we randomly pick 200 users and assume they are looking for friends inside a square. For Euclidean distance based method, users always use the smallest circle to cover the square, and then the accuracy is defined as when a friend is found inside the circle area, what is the probability that this friend actually is inside the real area of interest. The side length of the square varies as shown in the figure. We also vary the ratio of friends an active user has among all the users. It can be seen in Fig. 13, when the side length increases and/or the ratio of friends increases, the proposed method improves more on the accuracy, until the improvement reaches some stable values. These values are very close to the ideal theoretical analysis results which will not be shown in this paper due to the page limit. Clearly, Euclidean distance method is not flexible enough to describe a user defined area of interest. For irregular area, even for a square, the Euclidean distance method can not achieve accuracy rate of 1. However, polygon based method is more flexible in covering different types of irregular areas.

[2]The active user sends 32 bytes $\tilde{A^i}_{*1}$, receives 8 bytes $\tilde{x}_2$, sends 32 bytes $U^i - b^i$, and ignore last 1 bit results message.

5. RELATED WORK

In this section, we review some prior work in the privacy field, starting with privacy-preserving technique in user query location based service (LBS) and followed by particular location privacy techniques in friend locator and then some theoretical work in multi-party secure computation.

5.1 Privacy in Common User Query LBS

A common and practical technique people used for privacy preserving is anonymization, for example in [2, 6]. Especially, in k-anonymity model [7, 4, 14], the user privacy is protected by k-anonymity when the information for the individual contained in the release cannot be distinguished from at least k-1 individuals whose information also appear in the release. However, k-anonymity is not suitable for protecting privacy in friend locator applications in which users' identifications have to be known by others.

5.2 Privacy in Friend Locator

In Friendlocater [17], users are considered in proximity at certain level if their current locations are within the same cell or in two adjacent cells of that level. Users' location information in terms of cell is converted into an encrypted tuple before being sent to the server. Friendlocater protects a users' privacy by reducing the accuracy of the proximity detection algorithm. Our proposed algorithm, however, could accurately detect proximity between a pair of users based on their requirements. Continuing effort from the same group is shown in [15], however, the same issue still exists. In Pierre [19], Zhong et al. proposes three privacy protocols: Louis, Lester and Pierre which are all based on cryptography method. The problem of Louis and Lester is one of the users will know other user's exact locations or distances and pierre still has issue in the accuracy of proximity detection. The Hide&Crypt protocol [12] is a hybrid approach in which a secure computation is performed only after a filtering step based on obfuscated locations. Different from our proposed algorithm, Hide&Crypt does not have privacy protection during the filtering step. Furthermore, Hide&Crypt can only detect proximity based on the Euclidean distance between two users. Finally, the secure phase in Hide&Crypt is still based on spatial granularity (minimum uncertainty region). Longitude [11] can't guarantee privacy if the server and one of the users cooperate.

In [10], Manweiler et al. uses k-ID anonymity to protect user privacy. However, the overhead is too large and the privacy is not guaranteed in the situation where some clients and the server cooperate.

5.3 Secure Multi-Party Computation

The proposed convex polygon decomposition approach converts a friend locator problem to a secure multi-party com-

putation problem. However, it is very inefficient to use a secure multi-party computation method to solve friend locator problem without any modification. For example, the original solution in the famous Yao's millionaire problem [18] is impractical if the range of unknown variables is large. In [1], Atallah and Du present a secure two-party protocol for the point inclusion problem which is equivalent to the friend locator problem. However, they convert this problem into a secure two-party vector dominance problem, which then again needs to solve Yao's millionaire problem. In [8], the authors solve the point inclusion problem by converting area of interest into polygon, which is similar to the second phase in our algorithm. However, their work can't prevent one user from using degenerate polygon to identify the location of another user. In our algorithm, since the edges of the polygon has to be multiplied by the same matrix, other users can easily detect such attacks.

6. CONCLUSIONS

We have proposed a dual-resolution system and algorithm that can protect user's privacy in friend locator applications. Compared to previous work, our work introduces the following contributions and/or improvements. First, it uses a multi-level grid and variable-length bit sequences to represent users' locations, which reduces the overhead. Second, a dual-resolution structure helps reduce the total system overhead. Third, the proposed differential permutation method also helps to achieve privacy while balancing it with the overhead of every individual user. Fourth, this is to our knowledge, the first work to point out that dividing the geographical location into small cells based on population density improves privacy compared to traditional geographical splitting. Fifth, by converting a user's area of interest into multiple convex polygons, we could exploit the property of linear operation to achieve high privacy. This approach provides a privacy protection for an individual user even when the friends and the server are collaborating. Sixth, the polygon decomposition method also provides flexible and accurate way to define a user's area of interest especially for irregular shapes. The numerical and simulation showed that the proposed privacy preserving system can achieve high privacy, efficiency, accuracy and flexibility.

7. REFERENCES

[1] M. J. Atallah and W. Du. Secure multi-party computational geometry. In *Proceedings of the 7th International Workshop on Algorithms and Data Structures*, WADS '01, pages 165–179, 2001.

[2] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24:84–90, February 1981.

[3] W. Du and Z. Zhan. A practical approach to solve secure multi-party computation problems. In *IN NEW SECURITY PARADIGMS WORKSHOP*, pages 127–135, 2002.

[4] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 620–629, june 2005.

[5] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 218–229, 1987.

[6] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42:39–41, 1999.

[7] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *ACM MobiSys*, pages 31–42, 2003.

[8] G. Kϕien and V. Oleshchuk. Location privacy for cellular systems; analysis and solution. In *Privacy Enhancing Technologies*, volume 3856 of *Lecture Notes in Computer Science*, pages 40–58. Springer Berlin / Heidelberg, 2006.

[9] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In *CRYPTO'02*, pages 31–46, 2002.

[10] J. Manweiler, R. Scudellari, Z. Cancio, and L. P. Cox. We saw each other on the subway: secure, anonymous proximity-based missed connections. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, HotMobile '09, pages 1:1–1:6, 2009.

[11] S. Mascetti, C. Bettini, and D. Freni. Longitude: Centralized privacy-preserving computation of users' proximity. In *Proceedings of the 6th VLDB Workshop on Secure Data Management*, SDM '09, pages 142–157, 2009.

[12] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, and S. Jajodia. Privacy-aware proximity based services. In *Mobile Data Management*, pages 31–40. IEEE Computer Society, 2009.

[13] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia. On the impact of user movement simulations in the evaluation of lbs privacy- preserving techniques. In *the 1st International Workshop on Privacy in Location-Based Applications*, 2008.

[14] M. F. Mokbel, C. yin Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB*, pages 763–774, 2006.

[15] L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu. Private and flexible proximity detection in mobile social networks. In *Mobile Data Management'10*, pages 75–84, 2010.

[16] D. Stehle and R. Steinfeld. Faster fully homomorphic encryption. Cryptology ePrint Archive, Report 2010/299, 2010. http://eprint.iacr.org/.

[17] L. Šikšnys, J. R. Thomsen, S. Šaltenis, M. L. Yiu, and O. Andersen. A location privacy aware friend locator. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, SSTD '09, pages 405–410, 2009.

[18] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, 1982.

[19] G. Zhong, I. Goldberg, and U. Hengartner. Louis, lester and pierre: three protocols for location privacy. In *Proceedings of the 7th international conference on Privacy enhancing technologies*, PET'07, pages 62–76, 2007.