Dual-Resolution Friend Locator System With Privacy Enhancement Through Polygon Decomposition

Bin Zan, *Student Member, IEEE*, Fei Hu, *Member, IEEE*, Ke Bao, and Qi Hao, *Member, IEEE*

***Abstract*—Online social networks have become increasingly popular. One of the interesting applications is the friend locator, in which the application server informs a user through a mobile device if some of his/her listed friends are close enough in terms of geographical locations. However, in such services, it is challenging to protect the privacy of the individual users. Previous solutions for the friend locator do not guarantee a high level of privacy and do not maintain efficiency. In this paper, we propose a dual-resolution system structure to guarantee both strong privacy and efficiency. Additionally, we use the polygon decomposition method to achieve both accuracy and flexibility. To be more specific, in the coarse resolution level, each regular mobile user uploads his/her encrypted coarse location information to a central server periodically for comparisons. If a regular mobile user is found to be in the same grid block as an active mobile user, then the friend locator procedure with a higher resolution level will be conducted. Finally, through numerical analysis and simulations, we show that the proposed system design and algorithm can achieve high privacy, efficiency, accuracy, and flexibility.**

***Index Terms*—Friend locator, location privacy, polygon decomposition, social networks.**

I. Introduction

A friend locator is a location-aware social network application. In such an application, users with a Global Positioning System (GPS)-enabled mobile device will periodically update their current locations to a central server. By comparing a user's location information with his/her friends' one, the server can determine if any of his/her friends is close by and then inform this user. The user has to trust the central server and discloses his/her own location information to the server periodically. However, this is somehow in contradiction with a user's location privacy requirement. A user's location information is often associated with critical 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, they may raise rates accordingly.

Prior works on preserving privacy of general location-based services (LBS) do not fit the friend locator application 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 for 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 two users are adjacent. Some specific algorithms also have been developed for friend locator privacy preservation [10], [11], [17], [20]. However, Šikšnys *et al.* [11], [17] achieved privacy by sacrificing the accuracy of the results. To the best of our knowledge, none of the previous solutions can handle areas of interest with irregular shapes, as in our proposed algorithm.

In this paper, we aim to develop a new privacy-preserving system design and algorithm to achieve high privacy, accuracy, efficiency, and flexibility. To achieve this goal, we develop a dual-resolution system structure in which the coarse resolution corresponds to a relatively low accuracy result, and the fine-resolution level corresponds to a high accuracy result. The coarse level requires a small on-the-fly overhead, and the fine level requires a little more overhead. When combining these two levels, the total overhead can be reduced since fewer users from the coarse resolution level will be selected into the finer level for the succeeding procedure. Privacy protection is provided through differential permutation encryption method and the entropy-based multilevel grid in the coarse resolution level process. In the fine resolution level, an active user who is looking for nearby friends live provides more detailed information about his/her interested area to the selected friends. The interested area is described as a combination of multiple convex polygons. A friend is determined to be nearby if he/she is inside the interested area. By using convex polygon decomposition method on the interested area, we could exploit the property of linear operations to achieve strong privacy preservation in the high resolution level.

This paper is organized as follows. Section II describes the system model and problem statement. Section III provides the details of the proposed dual-resolution algorithm. Section IV evaluates the schemes through numerical analysis and simulations. Section V presents some options to enhance our scheme. Section VI has the security strength discussions. Section VII discusses related work. Section VIII concludes this paper.

Manuscript received March 17, 2014; revised September 23, 2014 and December 2, 2014; accepted January 31, 2015. Date of publication February 5, 2015; date of current version February 9, 2016. The review of this paper was coordinated by Prof. C. Zhang. *(Corresponding author: Qi Hao.)*

B. Zan is with the Wireless Information Network Laboratory, Rutgers University, North Brunswick, NJ 08902 USA (e-mail: zanb@winlab.rutgers.edu; gruteser@winlab.rutgers.edu).

F. Hu and K. Bao are with the Department of Electrical and Computer Engineering, The University of Alabama, Tuscaloosa, AL 35487 USA (e-mail: fei@eng.ua.edu; kbao@crimson.ua.edu).

Q. Hao is with the Department of Electrical Engineering, South University of Science and Technology of China, Shenzhen 518055, China (e-mail: hao.q@sustc.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2015.2400464

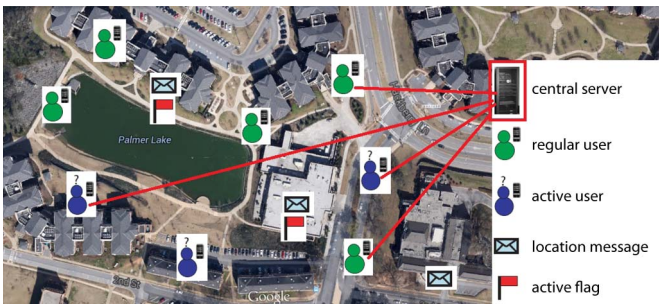


Fig. 1. System model.

II. System Model and Problem Statement

A. System Model

As shown in Fig. 1, the friend locator service has three parties: a central server, regular users, and active users (user with question mark on head). The users carrying mobile devices are called regular users. Each user periodically updates his/her location information to the server that stores the location information in its database. The active users are the users who are currently interested in finding their nearby friends. When a user wants to become an active user, he/she sends the server an active flag and the latest updated location information. The server will either help determine the nearby friends or start a procedure to help identify if they are nearby.

B. Problem Statement

The biggest challenge is to keep an individual user's exact location information unknown from others (including the central server and the other users) without affecting the execution of friend locator applications. Privacy is a general requirement for our daily lives. 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 private information from the location data of a particular user. However, for friend locator applications, the server has to collect user's location information to identify if two users are close to each other.

To summarize, we target the following objectives.

1) **Privacy:** Any location information of a user should not be directly known by the central server. The central server is assumed untrustworthy. The privacy should be preserved under passive attacks from the server.
2) **Efficiency:** The system should try its best to shift computation load to the central server, i.e., minimize the computation in each individual.
3) **Accuracy:** The system should find friends of a user based on its requirement of accuracy. For example, if a user wants to find all friends in the area of 5 mi^2, we should not include his/her friends 20 mi away.
4) **Flexibility:** The system should offer the active user flexibility in choosing his/her interested area, which means he/she might want to determine if a nearby user is in a particular area of interest.

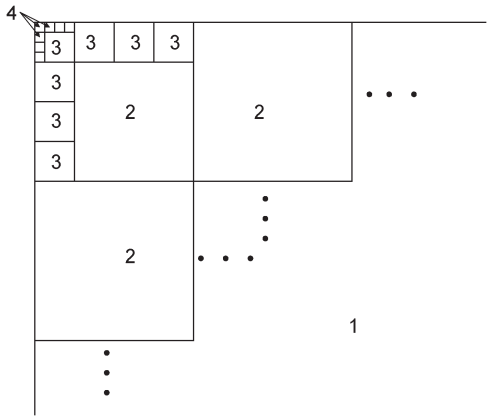


Fig. 2. Multilevel splitting. Numbers indicate the levels.

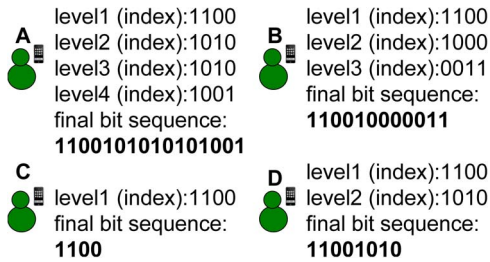


Fig. 3. Different users can choose their privacy preferences.

III. Privacy-Preserving Algorithm

Here we describe the proposed privacy-preserving algorithm (PPA) in detail. 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 2, after filtering all the friends who are not in the same geographical block, the active user further determines if a particular friend is inside his/her interested area by providing him/her with a convex polygon representation of the interested area.

A. Phase One: Multilevel Splitting

The entire map is viewed as different levels of blocks, as shown in Fig. 2, and the location information of a user is represented by a bit sequence. For example, A = 1100101010101001. We assume that each level occupies 4 bits in the bit sequence. Then, this example means that A is in: level 1 (block 12), level 2 (block 10), level 3 (block 10), and level 4 (block 9). Every user defines the length of his/her location bit sequence according to his/her 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 his/her friends. A short bit sequence results in more chances to be checked by others. As an example, in Fig. 3, user C only gives the first level information. Thus, for all other users A, B, and D, if any of them wants to figure out if user C is

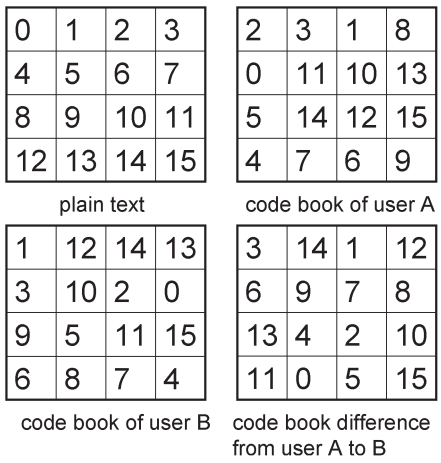


Fig. 4. Codebook of a user indicates the encrypted values the user will generate for real location indexes.

nearby, he has to ask user C to offer more information. On the other hand, the information given by user B is enough for users A and D to determine whether they are not nearby, or *vice versa*.

A user periodically updates his/her location information to the server. In addition to that, the bit sequence is encoded by a secret key that permutes the block numbers at a particular level. This key is only shared between this user and his/her friends. To find if any friend is nearby, a user sends multiple copies of his/her 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 the comparison of two bit sequences. Recall in the original scheme that, for every friend, a user has to provide a copy of his location bit sequence with special encryption through the common key agreed by that friend. If this user has a large number of friends, it clearly creates a lot of overhead. To reduce the overhead, we propose a differential method. The idea is to let each user encrypt (permute) his/her bit sequence using his/her own secret key, and let every pair of friends publish the distance of their secret keys. After knowing the "distance," the server could transform the permuted location bit sequence of a user, and make it comparable with another user's sequence.

For example, the bit sequence, 1100101010101001 (12,10,10,9), of user A, as shown in Fig. 3, can be encrypted by user A through codebook of A in Fig. 4, and becomes 0100110011001110 (4,12,12,14). The bit sequence, 110010000011 (12,8,3), of user B, becomes 011010011101 (6,9,13). The encrypted value 0100110011001110 and 011010011101 received by the server cannot be compared directly. However, after converting 0100110011001110 into 0110101110110101 according to the codebook difference table in Fig. 4, the server can identify that the users A and B have the same level 1 location; however, from level 2, they are at different blocks. Without knowing the codebook of A or of B, the server cannot figure out the real location index of user A or B.

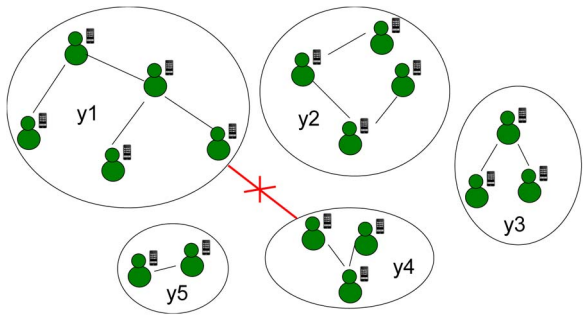


Fig. 5. Population attack. Note that the icons shown in each group represent the rough population size, which is not exactly means one user.

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

Assume the adversary is the server itself, and there are four blocks at certain level. The general population in each block is (1, 10, 2, 2) millions. As shown in Fig. 5, 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, then these two groups cannot be combined again. By finding the optimal way to allocate each group into a block, some of the block indexes can be disclosed with high probability. For example, it is reasonable to consider groups y_1 and y_2 to represent block 2, y_5 to represent block 1, and y_3 and y_4 to represent 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. Generally, this attack can be formalized into a nonlinear optimization problem. After linking all the pairs of users in the same block, the server obtains y number of groups. Moreover, it also has the population distribution of the m blocks. The goal of the server is to tag all the groups with real block indexes and achieve a distribution of the items in the bags close to the known population distribution. This is similar to a fractional bin packing problem, in which we have $\sum y_i x_i$ number of items and need to put into m bags, in which x_i is the group size. An extra condition is some of the items cannot be put into the same bag.

We can write down the above problem as

$$\text{Minimize} \quad \sum_{j=1}^{m} \left\| \sum_{i=1}^{n} y_{ij} x_i - C_j \right\|_2 \tag{1}$$

$$\text{Subject to} \quad x_i \geq 0, y_{ij} \in \{0,1\}, \sum_{j=1}^{m} y_{ij} = 1$$

$$E_{uv} = 0, u, v \in B_j$$

where x_i is the group size, and y_{ij} is a binary variable; $y_{ij} = 1$ if the group i is assigned to block j, and $y_{ij} = 0$ otherwise. C_j is the expected population size of the block j based on the observed users. One group can only be assigned to one block, but one block may have multiple groups. B_j is the jth block. Any two groups assigned to the same block should have no conflict edge; thus, $E_{uv} = 0$. For example, in Fig. 5, $y1$ and $y4$ has an edge between them, which indicates that they cannot be in the same block.

Generally, the above problem is NP hard, but there are numerous heuristic solutions, such as the one in [8]. Some users' privacy may be breached easily under certain circumstances.

We also observe that, if the block size in terms of population is divided evenly, although the optimal matching is perfect, the adversary would not know the right index value. As in Fig. 5, 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, we still do not know which one should be in block 3 or 4 exactly. When the population sizes of blocks are similar to each other, it is hard to distinguish them. This can be seen by using entropy to represent the whole system's uncertainty as

$$H = -\sum_{i=1}^{m} p(x_i) \log p(x_i) \tag{2}$$

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

Since phase one only uses the location bit sequence of a multilevel grid, it does not provide high accuracy for friend locator application. To achieve higher accuracy, the nearby users need to enter the second phase, i.e., the fine resolution level, to be discussed in the following.

B. Phase Two: Point Inclusion to a Convex Polygon

Through phase one, it is expected that only a small number of friends from a user's friend list will be selected for further exploration. The friends or buddies who are in different blocks are no longer considered.

Representing Interested Area by Convex Polygons: As will be shown later, our algorithm achieves strong privacy preservation through linear operations. By converting user's interested area into convex polygons, the original friend locator problem can be transferred to the point inclusion to a convex polygon problem. This problem then can be represented by matrix and solved through linear operation. While this conversion is for privacy purpose originally, we will show later that by solving the friend locator problem through point inclusion to convex polygon, it can also achieve higher accuracy and become more flexible than general Euclidean distance methods.

The first step is to convert a user's interested area into convex polygons. We first introduce some basic definitions and concepts.

(Half-space): A half-space in $\mathbb{R}^n$ is a set of the form

$$H = \left\{x \in \mathbb{R}^n : \quad p^T x \leq \alpha\right\} \tag{3}$$

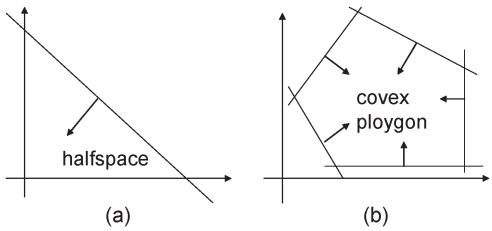


Fig. 6. Concept of half-space and convex polygon.

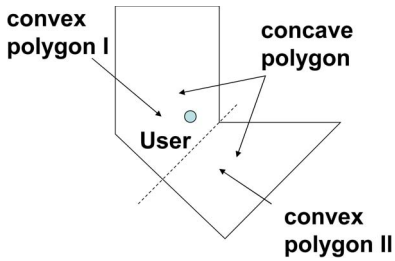


Fig. 7. Interested area is represented by a polygon.

where $p \in \mathbb{R}^n$ is a fixed nonzero vector, and α is a fixed real number. Fig. 6(a) gives a half-space example in 2-D space.

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

$$P = \left\{x \in \mathbb{R}^2 : \quad Ax \geq b\right\}. \tag{4}$$

Fig. 6(b) shows an example of convex polygon.

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

$$P = \left\{x \in \mathbb{R}^2 : \quad a_i^T x \geq b_i, \quad i = 1, \ldots, m\right\}. \tag{5}$$

The interested area of a user can be represented as a polygon. As shown in Fig. 7, a polygon is either a convex polygon itself, or it can be divided into a set of convex polygons. A polynomial-time algorithm to find a decomposition into as few convex polygons as possible is described by Chazelle [2]. Thus, we can write $P = P^1 \cup P^2 \cup \cdots \cup P^z$, and

$$P = \left\{x \in \mathbb{R}^2 : (A^1 x \geq b^1) \vee (A^2 x \geq b^2) \vee \cdots \vee (A^z x \geq b^z)\right\} \tag{6}$$

Preserving Privacy Through Matrix Operation: By representing interested area into a set of z convex polygons, we then determine if a user B's location $x = (x_1, x_2)$ is inside current user A's interested area. We check if x is a feasible point of P. Without privacy consideration, current user A only submits a set of matrices A^i and vectors b^i, which describe its interested area as a set of convex polygons 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 since a user's location

information will be disclosed to his/her friends. One common method to hide information is transformation. To be specific, let both A and B transform their interested area information or location information into a different coordinate system that 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 offer 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 [20], and user A has a higher flexibility in choosing its interested area instead of just using a circle of fixed radius.

Achieving Strong Privacy: When the central server cooperates with one of the users, neither the above method nor the homomorphic encryption method can protect the privacy of the second user. For example, with available information, i.e., 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 induced to a secure multiparty computation problem [3], [5], [19]. There are some methods to solve such problem as early as Yao's millionaire problem [19]. However, considering the complexity, many of them are impractical for the friend locator application. In the proposed algorithm, we develop a method that is inspired by [3]. In this method, we only disclose partial data in A^i by user A and x by user B. To be more specific, at user A, we have

$$\tilde{A}^i = A^i H = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ . & \\ . & \\ . & \\ a_{m1} & a_{m2} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} h_{11}a_{11} + h_{21}a_{12} & h_{12}a_{11} + h_{22}a_{12} \\ h_{11}a_{21} + h_{21}a_{22} & h_{12}a_{21} + h_{22}a_{22} \\ & . \\ & . \\ & . \\ h_{11}a_{m1} + h_{21}a_{m2} & h_{12}a_{m1} + h_{22}a_{m2} \end{bmatrix} \tag{8}$$

and at user B, we have

$$\tilde{x} = H^{-1}x = c \begin{bmatrix} h_{22} & -h_{12} \\ -h_{21} & h_{11} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{9}$$

$$= \begin{bmatrix} c(h_{22}x_1 - h_{12}x_2) \\ c(-h_{21}x_1 + h_{11}x_2) \end{bmatrix} \tag{10}$$

where $c = 1/(h_{11}h_{22} - h_{12}h_{21})$. It is easy to see that, given only one column of matrix $\tilde{A}^i$ or one row of vector $\tilde{x}$,the additional knowledge of matrix H cannot help an adversary to obtain the value of A^i or x. Therefore, let users A and B exchange part of their own information through the server. User A sends $\tilde{A}^i_{*1}$, i.e., the first column of matrix $\tilde{A}^i$, to user B, and user B sends $\tilde{x}_2$, i.e., the second row of vector $\tilde{x}$, to user 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

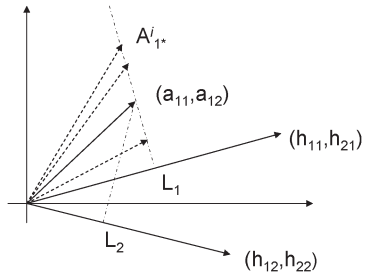


Fig. 8. Strong privacy protection in geometric theorem.

$$A^i x = \tilde{A}^i\tilde{x} = \tilde{A}^i{}_1\tilde{x}_1 + \tilde{A}^i{}_2\tilde{x}_2$$

$$= U^i + V^i \geq b^i \tag{11}$$

$$\Rightarrow U^i - b^i + V^i \geq 0 \tag{12}$$

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

We further illustrate the given ideas in Fig. 8. Assume H and H^{-1} are public known. When user A sends $\tilde{A}^i_{*1}$, i.e., the first column of matrix $\tilde{A}^i$, to user B through the server, it discloses the vertical intersection point L_1. That is because $\tilde{A}^i_{11}$ is the inner product of the vector $A^i_{1*} = (a_{11}, a_{12})$, and (h_{11}, h_{21}). If user A also sends $\tilde{A}^i_{12}$, which is equivalent to publicize the value of L_2, then, since two straight lines determine one point if they cross over, the real A^i_{1*} will be known. However, knowing that L_1 alone is not enough to recover the original vector A^i_{1*}, it could be any point on the vertical line passing the real value. The same argument also works for user B. Next, from the result of $U^i_1 = \tilde{A}^i_{12}\tilde{x}_2 = (h_{12}a_{11} + h_{22}a_{12})c(-h_{21}x_1 + h_{11}x_2)$, user B and the server can still derive the vertical intersection point L_2 of vector A^i_{1*} and (h_{12}, h_{22}). It is true that L_2, together with the knowledge of L_1, can reveal the original vector A^i_{1*}. However, user A does not send U^i_1 exactly; instead, the sum of U^i_1 and $-b^i_1$ is sent. Because b^i_1 is unknown, this is equivalent to say that L_2 can be any point on the vector (h_{12}, h_{22}) and can push the value L_2 back to an unknown status. Therefore, the exact value of A^i_{1*} can still be any point on the straight line passing by L_1 and the real point. On the other hand, since user B only replies with the final decision, there is not enough information disclosed at user B for an adversary or others to figure out the real value of (x_1, x_2).

To conclude, through the given method, we can achieve a strong privacy in phase two. Assume user A and the central server are colluding since 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. This information constructs a

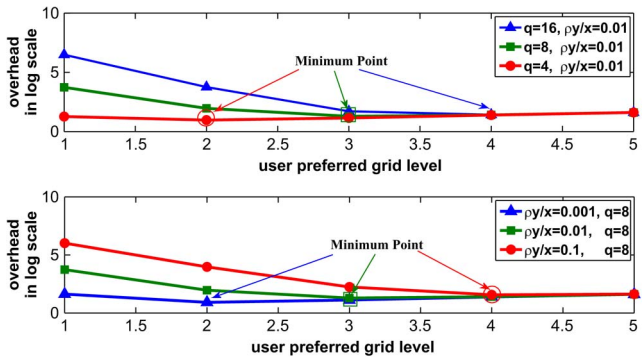


Fig. 9. For a regular user, a small grid level results in small communication overhead in each updating period; however, it increases the chance to enter phase two.

$2m^i$-equation system with $3m^i$ unknown variables, which in general has no unique solution.[1]

IV. Evaluation

We evaluate the proposed system through two steps: The first step is numerical analysis, and the second step is simulation.

A. Numerical Analysis

First, we study the advantage of using various lengths of sequences. We can generalize the communication overhead for a regular user during a location update period as

$$O = l * x + q^{L-l} * \rho * y \tag{13}$$

where l is the specific grid level the user preferred, x is the number of bits each level needs (assume the same length for all levels), L is the total level, ρ is the probability the user will be checked by his/her friend who is in the same block of the lowest level during the update period, q is the probability to be selected to enter phase two, and y is the communication overhead for phase two operations. In Fig. 9, we assume $L = 5$. It can be seen from the first subplot, while fixing the value of $\rho y/x$, that the smaller the value of q, the more communication overhead we save while reducing 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 computing the continuous relaxation version, we can minimize

$$l * x + q^{L-l} * \rho * y \tag{14}$$

subject to

$$l \in R \tag{15}$$

which has an optimum value at

$$l = L - \frac{\ln\left(\frac{x}{\rho y}\right) - \ln\left(\ln(q)\right)}{\ln(q)}. \tag{16}$$

[1]By choosing special matrix H, it is possible to determine the original values through partial information; however, in such a case, users can decide if the public H is safe for using.

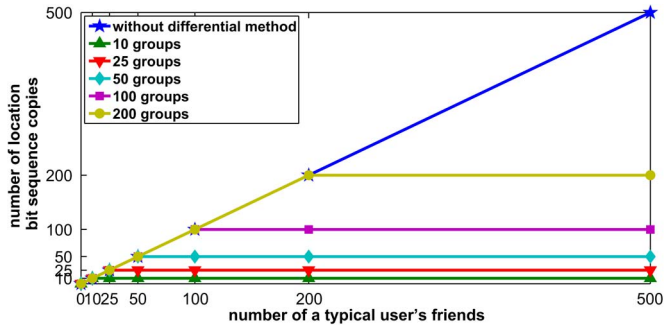


Fig. 10. Constant number of groups for a user.

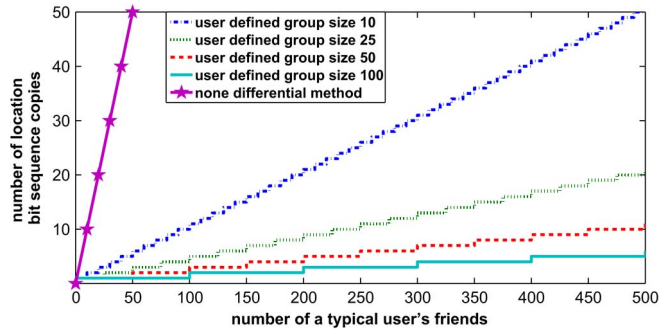


Fig. 11. Fixed size of a group for a user.

Then, we can further determine the integer solution.

Next, we study the advantage of using differential permutation. As we know, through differential permutation, a regular user can avoid uploading multiple copies of his/her location bit sequence. The only disadvantage of this method is that if one user leaks his/her secret key, the server or adversary can also break his/her friends' secret keys. Therefore, we offer two methods based on differential permutation to avoid ripple effect. First, we allow a user to have a 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 a user's own preference, he/she assigns his/her friends 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. 10, while we fix the number of groups, the overhead is first increasing as in nondifferential methods, and until the user has all groups activated, then the number of copies stays in the same level. In Fig. 11, when the size of each group is fixed, the overhead will keep increasing; however, the speed is far lower than nondifferential methods.

B. Simulations

Here, we further study the proposed algorithm through simulations. The data set of user movements is obtained through the MilanoByNight simulation [13] by EveryWare Laboratory of the University of Milan, Milan, Italy. The data set represents 100 000 users' movement in the city of Milan during a weekend night. The whole data set is collected over 5 h, and locations are sampled every 20 s. The total size of the map is 174 mi^2, and the average density is 572 users/mi^2.

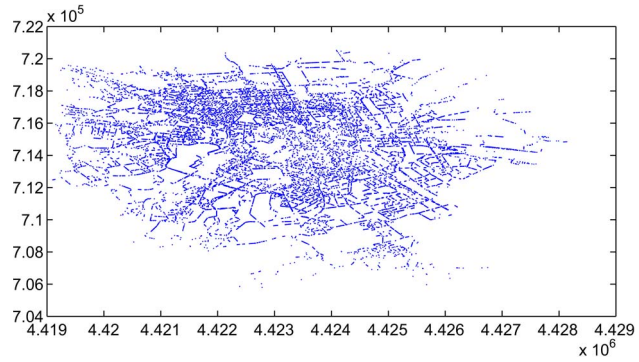


Fig. 12. Map of application users.

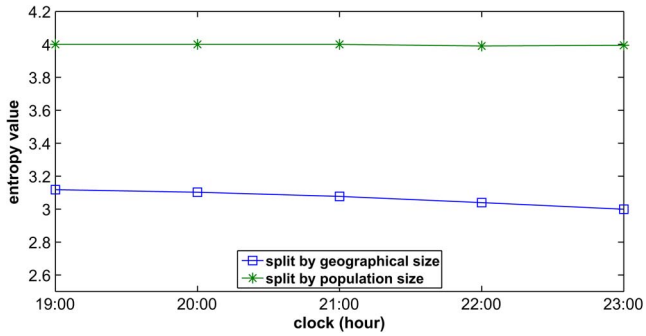


Fig. 13. Population-based split achieving higher entropy values.

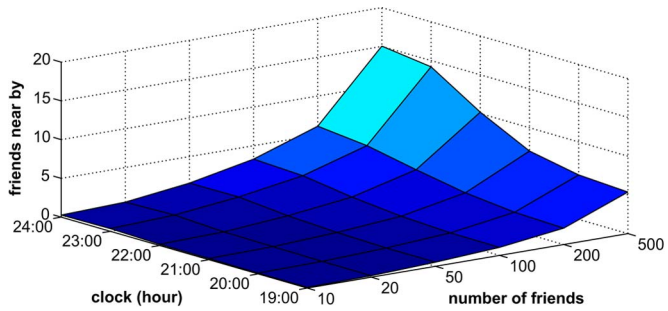


Fig. 14. Whole map is divided into two levels, and each level has 16 blocks. The number of friends each user has varies from 10 to 500. As shown, in average, a user has far fewer nearby friends than the total number of friends he/she has. Therefore, a two-phase structure is necessary, and it can dramatically reduce the overhead of each user.

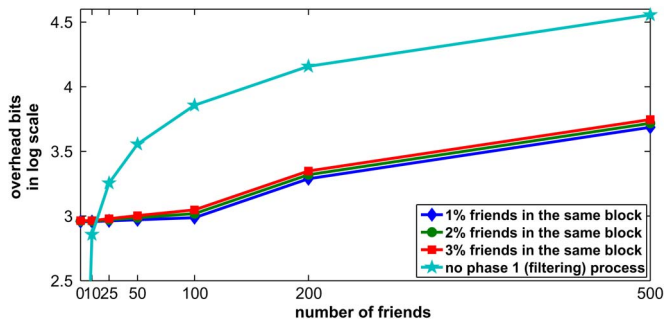


Fig. 15. Active user's total communication overhead during the 5-h period.

Fig. 12 shows a snapshot of the user distribution at a random time.

In the first experiment, we split the whole map into 16 blocks 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 scheme. As shown in Fig. 13, by splitting the map 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 friends (buddies) varying from 10 to 500, and we assume there are two levels of grids. In the first level, the whole map is divided into 16 blocks, and in the second level, each block in the first level is again divided into 16 blocks. As shown in Fig. 14, after phase one, the proposed algorithm eliminates most friends of each user and dramatically reduces the overhead in phase two operation. Most times, 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 show the maximum of 15 friends nearby. Following this result, we further show an active user's total communication overhead during the 5-h period in Fig. 15. We assume each user updates his/her location every 20 s, and every user uses the most accurate level, which is 2 in this case. The length of location bit sequence is 1 B for everyone. We also assume that, for every 100 friends, a user uses a special secret key, and the active user's interested area is a rectangle. According to the proposed algorithm, the communication overhead for the active user with every friend in the same block is 72 B.[2] As shown in the figure, by using two phases, we can dramatically reduce the communication overhead for an active user except when a user has only a very small number of friends because, at that time, periodically uploading location information causes high communication overhead.

Next, we show the accuracy of using the proposed polygon decomposition method, compared with previous work [20], which is based on homomorphic encryption. The homomorphic encryption method can be used to determine if two users have a Euclidean distance that is less than a predefined value. We use the square shape as example. In the simulation, we randomly select 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 the accuracy is defined as follows: When a friend is found inside the circle area, what is the probability that it is actually inside the real interested area? In Fig. 16, we show that, when the size of the square changes (side length varies among 100, 200, and 400 m), the accuracy changes. For small side length, the chance to get accuracy of either 0 or 1 is higher than the large side length. This is because when the interested area is small, the probability that none of a user's friends is inside both the interested area and the circumcircle becomes large, which results in more appearance of accuracy of 1. On the other hand, a small interested

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

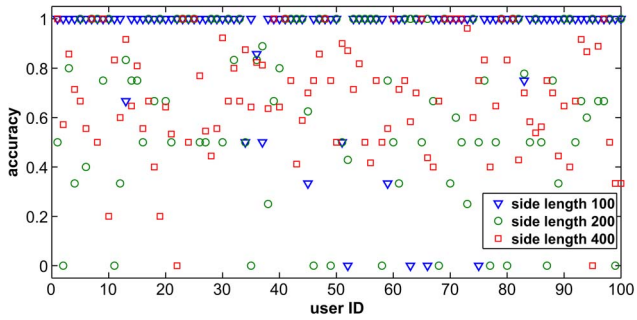


Fig. 16. Accuracy observed from the simulation data, in which we assume among the 100 000 users that 2% of users are a particular user's friends. The side length of the square interested area varies from 100, 200 to 400 m.

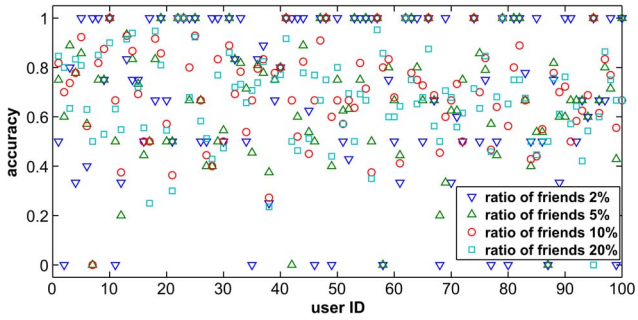


Fig. 17. Accuracy observed from the simulation data, in which we assume the user interested area is the square area centered at his current location, and the side length is 200 m.

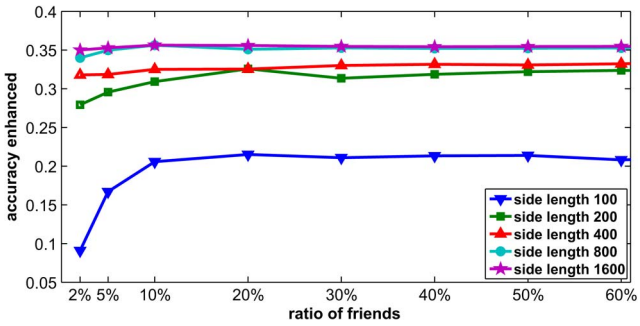


Fig. 18. By using a polygon-based method, the accuracy of the friend locator application can be improved. When the size of the interested area and/or the density of friends increases, the enhancement finally reaches the analysis value.

area also results in higher probability of having friends inside the circumcircle but not in the real interested area. In Fig. 17, we fix the size of intersected area but vary the ratio of friends. It can be seen that a small ratio of friends results in more accuracy of 0 and 1. The reason for this is similar to the reason for a small interested area.

Finally, in Fig. 18, we show the accuracy improvement by the polygon decomposition method. As shown in the figure, when the size of the interested area and/or the friends density increases, the improvement of accuracy becomes closer to the theoretical value.

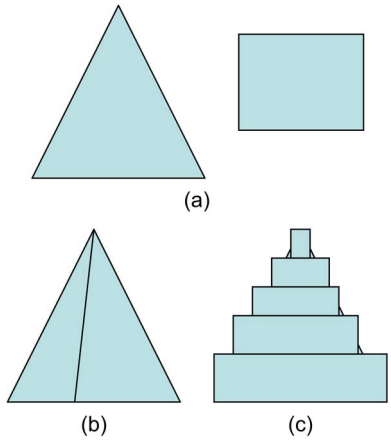


Fig. 19. Improve privacy by different types of decompositions. (a) Incorrect convex polygon. (b) Extra splitting. (c) Approximate representation.

V. Discussions

Recall the polygon decomposition uses matrix to represent convex polygons. For different friends, we require the active user to use different sets of convex polygons to represent an interested area to prevent the attacks from the collusion of multiple users and the server. Here, we give some suggestions on how to choose different types of decompositions.

1) Multiply a row of A by random numbers. For example, the active user has $x_1 + x_2 \geq 3$ in matrix A for one friend. However, for another friend, the active user uses $3.44x_1 + 3.44x_2 \geq 10.32$ to replace the row in matrix A. Note that this method alone does not work; however, combining with other methods will cause confusion at the adversary side.
2) Include incorrect convex polygon. An incorrect convex polygon can be added in matrix A to confuse the adversary. An example is shown in Fig. 19(a).
3) Split a convex polygon into multiple convex polygons. The active user could do unnecessary splitting on the convex polygon to confuse the adversary, as shown in Fig. 19(b).
4) Use convex polygons to approximately represent the exact area. As shown in Fig. 19(c), this method sacrifices a little accuracy, However, since this method can replace multiple rows of the matrix A completely, it is hard to break.

When combining all the given methods, it becomes even harder for the adversary to figure out the real interested area. Note that we assume a statistical attack is impossible in such case since it is not normal for an active user to have thousands of friends nearby.

VI. Privacy Analysis

A. On Level 1 (Coarse Resolution) Privacy

We now formally proof that the privacy of our friend locator scheme can be achieved. We first analyze the security strength of the coarse level process, which is based on differential permutation.

Lemma 1: Denote the server as S and each mobile user as U_i. Assume users are authenticated through third-party methods. We also assume that an adversary is not interested in spoofing attack (such as masquerading as a friend of the target user). The adversary's only interest is to break the location privacy, i.e., it tries to deduce the user's exact location or the distance between users. The proposed first-level friend-finding scheme, i.e., a multilevel location splitting coding and a U_i-to-S location update protocol based on differential permutation, can achieve the following two effects in friend locator applications: 1) It greatly reduces the communication overhead among U_i and S, and 2) it achieves desired privacy level for U_i.

Proof: The purpose of using a differential permutation method is to achieve the privacy and improve the efficiency of the multilevel splitting algorithm. We define a set as a collection of distinct objects. The proposed multilevel splitting algorithm defines a user's location information as an object in a set of plaintext. Being encoded with a secret key for a location bit sequence is equal to transfer the object into the corresponded object in another set, which is a set of encrypted text. We call such a process as transfer mapping. The objects in two encrypted text sets are one-to-one mapping. However, the objects in plaintext and encrypted text sets may not necessarily be one-to-one mapping.

Originally without a differential permutation method, every pair of friends shares an encryption set. Thus, each user U_i has to prepare k copies of its location information since he/she needs to have one copy in every set of encrypted text that he/she shares with a friend for encrypted data. Then, the server S can compare the location information between two friends in the same set.

Communication Overhead Reduction: With a differential permutation method, however, U_i only publishes one copy of its location information, i.e., an encrypted object in its own set. Since S knows the distance between two sets, in another words, it knows how to convert an object in one set to the corresponded object in another set, and S can make the comparisons between two location bit set sequences. This reduces the total message from k to 1 for U_i.

Privacy Protection: S does not know how to transfer an object in the set of encryption to the set of plaintext of U_i. Although it knows how to convert location information between two sets of encrypted text, it still is not able to convert an object in an encrypted set to an object in the plaintext set. Of course, if S steals the information from U_i and knows one plaintext with location information and its corresponding encrypted text, it is able to find out the specific encrypted text location information represented by a friend of U_i through the distance information that it knows. However, depending on the complexity of the mapping, it still has no way of knowing other cases. For example, if the mapping is n complexity, which means that the rule of mapping can only be solved when n corresponding mapping results are known. One location record stolen by S gives it only $1/n$ knowledge of the mapping, but it still cannot solve the mapping fully. The higher complexity n, the more difficult S can reversely figure out the mapping rules.

When S knows every possible object in the plaintext set of U_i and its matched object in the encrypted text set, it is possible for S to also know each object in a friend of the users. However, this will be also true for any method without using differential permutation. The differential permutation method maintains the same security level as without the differential permutation, as long as the conversion from U_i's plaintext set to his/her corresponding encrypted text set is not fully known to S.

B. On Level 2 (Fine Resolution) Privacy

Lemma 2: Following the assumptions in Lemma 1, by using our proposed convex polygon decomposition scheme with only partial information exchange among user A, user B, and server S, we can achieve privacy protection of location locator application. The privacy level has similar privacy strength as in ideal case, i.e., user A knows whether user B is in its interested area, without information exchange.

Proof: In Section III we have shown that both users A and B's privacy levels can be preserved as long as users A and B only disclose part of their location matrix elements. We have used geometric theorem (see Fig. 8) to show that it is computationally infeasible to deduce the exact location or distance information from the transformed matrix parameters. Therefore, our friend locator algorithm in the fine resolution level preserves strong privacy. Here, we provide more analysis on the privacy performance of our convex-polygon-based scheme.

The most common way to figure out if two users are geometrically close by is to publish or exchange both users' location information. This method does not consider the privacy requirement of the users. To avoid exact location information disclosure to the public, a pair of friends could exchange their information with encryption. However, we still cannot protect a user's location information from his/her friend. To satisfy the privacy requirement for users, even friend should not know more than what should they need to know. In a friend finder application, a user only needs to know if a friend is nearby or not; he/she does not need to know the exact location of a friend.

The proposed convex polygon method achieves strong privacy by transferring secure friend-finding problem into a secure multiple-party computation problem. There is certain downside to achieve strong privacy—it reduces the dimension of the secret. For example, originally, a user B's location may be represented by two numbers (x, y) on a map or a plane. As explained in Section III, when messages are exchanged during phase 2, user B's location will be known to be limited within a particular line. From a point on a plane to a point on a line, this reduces a secret from a 2-D space into a 1-D space. Theoretically, there are still an infinite number of points on a line. If, originally, a secret in a 2-D space can be broken in 10 min, then the 1-D case can be broken in 5 min. On the other hand, if the original 2-D space is not breakable, the 1-D space does as well. Although moving from 2-D down to 1-D is a security degradation, it is the sacrifice we are willing to make in order to achieve a high privacy level: Even friends should not know more than what they need to know.

Fig. 8 shows that our convex-polygon-based privacy scheme is based on the impossible decomposition of a matrix operation

if only partial parameters are disclosed to each other. However, if too many parameters are revealed, or too many neighbors are colluding with each other, we cannot achieve desired privacy. As a matter of fact, since any point is limited to a line, if the server or a user A change the value of $u_i - b_i$ for many times, user B's location will be disclosed. However, this kind of attack is also true even in other location security schemes [21]–[25].

Lemma 3: The proposed multilevel splitting coding in a coarse level process, and the matrix transformation in a fine level process can, together, prevent the colluding attack between S and U_i.

Proof: In the ideal case, the server only knows whether user B is in the interested area of user A. Without colluding with one of the friends, when the location information is updated, the server may get a sequence of yes or no. This does not help the server to know where they are. Furthermore, this special "yes" or "no" messages can be exchanged in an encrypted way. Thus, the server will not obtain any information. However, the proposed method does leak other information. As discussed in Section III-B, since the value of H is publicly known, a location information will be limited to be on a line. To discover the true location of a user, however, we need another nonparallel line to get the intersection point. For the same pair of friends, since they will use the same H for multiple comparisons, if a users location is not changing at all, then the server will also observe the same line. On the other hand, if the user's location is changed, it is still hard to figure out two different points on two lines. Furthermore, we can protect user location from leaking to the server by keeping the value of H to be a common secret between friends. This way, the server will not be able to gather any information from the information exchange of two users.

VII. Related Works

A. Privacy in Common User Query LBS

In common user query LBS, an LBS server stores a large number of public information such as point of interest (POI). A user tells the LBS server on his/her location information in order to find the nearby POIs. The target of privacy protection is to hide the location information of the user from the server without decreasing the functionality of the query application. A common technique for privacy preserving is anonymization, e.g., in [1] and [6]. In particular, in the k-anonymity model [4], [7], [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. The spatiotemporal cloaking [7] and the clique-cloak algorithm [14] are both based on k-anonymity. However, k-anonymity is not suitable to protecting privacy in friend locator applications.

B. Privacy in Friend Locator

In the friend locator [17], Šikšnys *et al.* proposed a PPA for detecting proximity among friend pairs within a client–server architecture. The friend locator 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 users' requirement. Continuing effort from the same group is shown in [15]. However, the same issue still exists.

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. Through geometrical transformations, Longitude [11] hides the real location information of a user from the server while still allowing the server to calculate the Euclidean distance between two users. However, it is not as flexible as our algorithm. Furthermore, Longitude cannot guarantee privacy if the server and one of the user cooperate.

In [10], Manweiler *et al.* uses k-ID anonymity to protect user privacy. In addition to its huge overhead (when the server forwards a query probe to the source of an update, it must broadcast the message to the entire identifier set), the privacy is not guaranteed in the situation where some of clients and the server cooperate.

C. Secure Multiparty Computation

The proposed convex polygon decomposition approach converts a friend locator problem to a secure multiparty computation problem. However, it is inefficient to use a secure multiparty computation method to solve the friend locator problem with no modifications. For example, the original solution in the famous Yao's millionaire problem [19] is impractical if the range of unknown variables is large.[3]

VIII. Conclusion

We have proposed a dual-resolution system architecture and algorithm that can protect user's privacy in the friend locator application. Compared with previous work, this paper has the following advantages. First, it uses multilevel grid and varied length of bit sequence to represent a user's coarse location, which reduces the overhead. Second, a dual-resolution system helps reduce the total communication overhead of all users. Third, the proposed differential permutation method achieves strong privacy while balancing the overhead at the same time. Fourth, dividing the geographical location into small blocks based on population is better than traditional geographical splitting. Fifth, by converting a user's interested area into multiple convex polygons, we could exploit the property of linear operation to achieve high privacy preservation.

References

[1] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.
[2] B. Chazelle and D. P. Dobkin, "Optimal convex decompositions," in *Computational Geometry*, G. T. Toussaint Ed. Amsterdam: The Netherlands, Elsevier, 1985.

[3] In our case, the unknown variables are the exact location (x, y) of one user and the interested area $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ of another user's; the range can be as large as R^2 and R^{2n}, respectively.

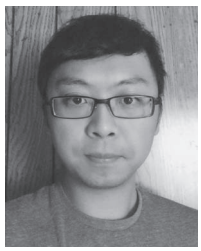
[3] W. Du and Z. Zhan, "A practical approach to solve secure multi-party computation problems," in *Proc. New Security Paradigms Workshop*, 2002, pp. 127–135.
[4] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *Proc. 25th IEEE ICDCS*, 2005, pp. 620–629.
[5] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. 19th Annu. ACM STOC*, New York, NY, USA, 1987, pp. 218–229.
[6] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing for anonymous and private Internet connections," *Commun. ACM*, vol. 42, no. 2, pp. 39–41, Feb. 1999.
[7] M. Gruteser, D. Grunwalddepartment, and C. Science, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. ACM MobiSys*, 2003, pp. 31–42.
[8] S. A. M. John Oommen, O.-C. Granmo, and M. G. Olsen, "Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 166–175, Jan. 2007.
[9] F. Kandahl, Y. Singh, W. Zhang, and C. Wang, "Mitigating colluding injected attack using monitoring verification in mobile adhoc networks," *Security Commun. Netw.*, vol. 6, no. 4, pp. 539–547, Apr. 2013.
[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 *Proc. 10th Workshop HotMobile Comput. Syst. Appl.*, New York, NY, USA, 2009, pp. 1:1–1:6.
[11] S. Mascetti, C. Bettini, and D. Freni, "Longitude: Centralized privacy-preserving computation of users' proximity," in *Proc. 6th VLDB Workshop SDM*, Berlin, Germany, 2009, pp. 142–157.
[12] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, and S. Jajodia, "Privacy-aware proximity based services," in *Proc. Mobile Data Manage. Syst., Serv. Middleware*, 2009, pp. 31–40.
[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 *Proc. 1st Int. Workshop Privacy Loc.-Based Appl.*, 2008, pp. 1–21.
[14] M. F. Mokbel, C. Yin Chow, and W. G. Aref, "The new casper: Query processing for location services without compromising privacy," in *Proc. VLDB*, 2006, pp. 763–774.
[15] L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu, "Private and flexible proximity detection in mobile social networks," in *Proc. 11th Int. Conf. Mobile Data Manage.*, 2010, pp. 75–84.
[16] D. Stehle and R. Steinfeld, "Faster fully homomorphic encryption," Cryptology ePrint Archive, Rep. 2010/299, 2010. [Online]. Available: 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 *Proc. 11th Int. SSTD*, Berlin, Germany, 2009, pp. 405–410.
[18] X. Wang, L. Qian, and H. Jiang, "Tolerant majority-colluding attacks for secure localization in wireless sensor networks," in *Proc. 5th Int. Conf. WiCom, Netw. Mobile Comput.*, 2009, pp. 3427–3431.
[19] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. SFCS*, Washington, DC, USA, 1982, pp. 160–164.
[20] G. Zhong, I. Goldberg, and U. Hengartner, "Louis, Lester and Pierre: Three protocols for location privacy," in *Proc. 7th Int. Conf. PET*, Berlin, Germany, 2007, pp. 62–76.

Bin Zan (S'12) received the B.S. degree in communication and information engineering from the University of Electronic Science and Technology of China, Chengdu, China; the M.S. degree in computer science from Clarkson University, Potsdam, NY, USA, in 2006; and the Ph.D. degree in electrical and computer engineering from Rutgers University, North Brunswick, NJ, USA, in 2013.

His research interests include location-aware systems, mobile networking, privacy, and security.

Fei Hu (M'06) received the Ph.D. degree in signal processing from Tongji University, Shanghai, China, in 1999 and the Ph.D. degree in electrical and computer engineering from Clarkson University, Potsdam, NY, USA, in 2002.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL, USA. He is the author of over 200 journal/conference papers and books. His research has been supported by the U.S. National Science Foundation, the U.S. Department of Defense, Cisco, Sprint, and other sources. His research interests include wireless networks and machine learning.



Ke Bao is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, The University of Alabama, Tuscaloosa, AL, USA.

His research interests include wireless networks, multimedia quality of service, and wireless test beds.

Qi Hao (M'06) received the B.E. and M.E. degrees from Shanghai Jiao Tong University, Shanghai, China, in 1994 and 1997, respectively, and the Ph.D. degree from Duke University, Durham, NC, USA, in 2006, all in electrical engineering.

His postdoctoral training with the Center for Visualization and Virtual Environment, The University of Kentucky, Lexington, KY, USA, was focused on 3-D computer vision for human tracking and identification. From 2007 to 2014, he was an Assistant Professor with the Department of Electrical and Computer Engineering, The University of Alabama, Tuscaloosa, AL, USA. He is currently an Associate Professor with South University of Science and Technology of China, Shenzhen, China. His research has been supported by the U.S. National Science Foundation and other sources. His current research interests include smart sensors, intelligent wireless sensor networks, and distributed information processing.