Friendship-based location privacy in Mobile Social Networks

Wei Chang*, Jie Wu and Chiu C. Tan

Department of Computer and Information Sciences,
Temple University,
Philadelphia, PA 19122, USA
E-mail: wei.chang@temple.edu
E-mail: jiewu@temple.edu
E-mail: cctan@temple.edu
*Corresponding author

Abstract: Location privacy in Mobile Social Networks (MSNs) has generated significant interest in recent years, with many proposed methods to address the problem. Commercial solutions to this problem have suggested designing better ways for users to determine when to report their locations, while academic researchers have proposed solutions that involve deploying trusted third party servers to protect user privacy. In this paper, we showed that simply omitting location updates does not provide adequate privacy protections, especially in situations where the friendship relationships between users are known. We proposed a fake location update algorithm that allows a user to protect his privacy. A key feature of our approach is that it can be adopted without the use of any third party services, making them more practical. We evaluate our approach using extensive simulation experiments.

Keywords: closeness; inference attack; location privacy; MSNs; mobile social networks; trajectory estimation.

Reference to this paper should be made as follows: Chang, W., Wu, J. and Tan, C.C. (2011) 'Friendship-based location privacy in Mobile Social Networks', *Int. J. Security and Networks*, Vol. 6, No. 4, pp.226–236.

Biographical notes: Wei Chang received BSc in Computer Science in 2009 from Beijing University of Posts and Telecommunications, China. He is a PhD student at Computer and Information Sciences department, Temple University, USA. His research interests include security and privacy issues in mobile social networks.

Jie Wu is Chair and Professor at the Department of Computer and Information Sciences at Temple University, USA. He is an IEEE Fellow. He is on the editorial board of IEEE Transactions on Mobile Computers and Journal of Parallel and Distributed computing. He was a Distinguished Professor in Department of Computer Science and Engineering, Florida Atlantic University. He served as a Program Director at US NSF from 2006 to 2008. He has served as a Distinguished Visitor of the IEEE Computer Society, and he is also an ACM Distinguished Speaker. His research interests include wireless networks and mobile computing, parallel and distributed systems, and fault-tolerant systems.

Chiu C. Tan is a Research Assistant Professor in the Computer and Information Sciences Department at Temple University. He received his PhD in Computer Science from the College of William and Mary. His research interests include wireless security (802.11, vehicular, RFID), cloud computing security, and security for mobile health (mHealth) systems. He is a member of IEEE.

1 Introduction

The increasing popularity of smartphones has led to the rise of Mobile Social Networks (MSNs). An MSN is a combination of online social networks and location-based services, and can be used to provide many new services, such as a friend notification service that alerts a user when his friends are near his location. In order to provide these services, the MSN provider has to collect location information from users and their friends. This has led to the concern that such information may pose a privacy threat since users may be unaware that they have revealed some sensitive information until after the fact.

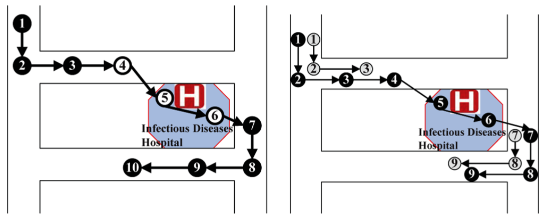
One technique to protect privacy is to give users more control over when their locations are updated. The intuition is that individuals are the best arbiters of what locations are private. Therefore, simply by allowing a user the option to *not* upload his location at such sensitive locations, we can achieve location privacy. However, this intuition may not adequately provide location privacy against an adversary that knows both a user's historic location information *and* his social relationships.

We can illustrate this by using the following example. Let us consider the map shown in Figure 1(a), where the center region is a hospital. A user may decide that the hospital is a sensitive area and may choose not to upload his location information when he is near the hospital. Now let us assume that the MSN provider is an adversary that is seeking to learn more information about the user. The adversary can use the previously reported locations to predict the user's movements and infer that he has visited the hospital. In addition, we see in Figure 1(b) that the user is travelling with his friend when he visits the hospital. The user is careful to not upload his location information when nearing the hospital, but his friend does not remember to do so. Since the adversary is aware of the friendship information, the adversary can use the friend's location information to estimate any gaps in the user's reporting.

In this paper, we consider the problem of providing location privacy against an adversary that has access to the data collected by the MSN provider. We make the following contributions:

- we are the first to consider an attack where the adversary uses both the historical trace data and friendship information to predict a user's movements
- our proposed solution utilises Kalman filters to determine the best fake location to upload so as to defeat the MSN's location prediction algorithm

Figure 1 In (a), the dark cells represent reported locations, while the white cells represent unreported positions. The number in each cell indicates the observation sequence. We can see that we still can guess the positions of a user even if he omits reporting certain locations. In (b), there are two friends, represented by black and grey circles. The two friends are walking together, but the first user (grey circle) does not report his location from time step 4–6 while his friend (black circle) does. Using the location information from the friend, we can still estimate the user's location (see online version for colours)



(a) single user (b) two users

- our solution does not require the use of trusted third parties, which may be difficult to deploy in the real world, to provide location privacy protection.

The rest of the paper is organised as follows. Section 2 summarises related works, and Section 3 provides an overview about Kalman filters, the adversary model, and our approach to the problem. Our proposed solution is found in Section 4, and Section 5 contains an additional discussion. In Section 6, we evaluate our approach, and Section 7 presents the conclusion of this paper.

2 Related work

We adopt the classification used in Chao and Dongyu (2010) and divide location privacy research into two major categories: those that provide anonymity and those that introduce obfuscation.

Location privacy can be provided via the frequent changing of pseudonyms (Marco, 2003; Huang et al., 2005; Freudiger et al., 2009; Jiang et al., 2007) to make it difficult for adversaries to estimate a user's trajectory. The system first defines several special regions named 'mix zones'. In a mix zone, a number of nodes will enter the zone and leave the zone. Within each mix zone, nodes change their pseudonyms at the same time and do not report their locations within the zones. Since the changing of pseudonyms among nodes is performed simultaneously, the adversary cannot get the complete trajectories of users, thus preserving location privacy. However, for MSN-type applications, the MSN provider will still have to know the friendship relationships for each pseudonym so as to deliver the correct information. As such, the frequent changing of user IDs may not mask the user's identity.

Another technique of providing anonymity is to use k-anonymity techniques (Kulik et al., 2009; Chow and Mokbel, 2009; Ouyang et al., 2008; Damiani et al., 2009). In these solutions, a user will only upload the location of a region which contains $k - 1$ neighbours. This will ensure that the adversary cannot pinpoint the exact location of a user. The use of k-anonymity in an MSN is not feasible because the user's friends can only receive an approximate area and cannot determine the location of the user. This will make an MSN less useful.

The other category of location privacy-preserving techniques is obfuscation. In obfuscation-type solutions, the user will try to confuse the adversary through techniques such as injecting noise (Hidetoshi et al., 2005; Krumm et al., 2007; **?**), reporting fewer locations (Hoh et al. (2007)), reporting false locations (Hidetoshi et al., 2005), or increasing the intervals between reported locations (Hoh et al., 2006). Our approach follows this category of location privacy-preserving techniques. The difference is that we consider a more powerful adversary that can utilise *friendship* information to better predict user movements. As shown earlier, friendship information can be combined with location information to predict user movements.

A recent paper (Chang et al., 2011) considered a similar problem involving social relationships and location privacy. However, that solution only considers the case of one or two friends travelling together, and does not provide a solution for multiple friends travelling as a group. Moreover, paper (Chang et al., 2011) paper also does not consider the quality of service and energy issues when there is a group of users. In this paper, we will present an energy and friendship-based location privacy-preserving method, and we will also discuss methods of improving the quality of service if the users are using fake locations to protect their privacy.

Finally, while some researchers have proposed cryptographic techniques (Puttaswamy et al., 2010; You et al., 2007) to protect user privacy, these solutions generally require the cooperation of the MSN provider. Since there appears to be little incentive for the MSN provider to perform such operations, it is unclear how practical such solutions will be. We do not consider these solutions in this paper.

3 Overview

In this section, we first briefly describe the system model and the adversary model. Then, we provide our location privacy-preserving metric and illustrate how our proposed method defends against the adversary's attack.

3.1 System model

Our system consists of an MSN provider and many users. We divide time into discrete time units; at each time unit, a user can report his actual location, a fake location, or no location at all. We assume that friends of a user will be able to distinguish between a real or fake location. For instance, a user could label his locations with unique IDs so that his friends can filter out fake locations. A user can query the MSN provider for his friends' location at any time. However, the request will only be transmitted to the MSN at a predefined time step. If there are multiple requests at a time step, the phone will only provide the nearest location. Table 1 contains the notations used.

Table 1 Notations

K	Kalman gain (updating ratio)
H	Observation model matrix
F	State transition model matrix
I	Identity matrix
Δd	Extra distance provided by a user at a time step
$I(L_A; L_B)$	Mutual information between two sets of locations
$H(L_B)$	Entropy of user B's locations set
$H(L_B\|L_A)$	Entropy of B's locations set, given user A's locations set

We consider an adversary that has control over the operations of the MSN provider. Thus, the adversary will have access to all of the location updates and friendship information stored in the MSN. The adversary will use this information to filter out fake locations and to improve any the missing-location predictions.

3.2 Kalman filter

The adversary will use the Kalman filter to estimate users' unreported locations. The Kalman filter is a set of mathematical equations that provide an efficient computational (recursive) means for estimating the state of a process in a way that minimises the mean of the squared error (Welch and Bishop, 1995). Forward and backward Kalman filling (Lavanya and Elaine, 2004) is a method, derived from the Kalman filter, to estimate the missing data within a linear system.

The Kalman filter consists of a *predict* phase and an *update* phase in each time step. In the predict phase, the filter estimates the state of the current time step from the estimated state in the previous time step. After the filter obtains the observation of the current time step, the algorithm enters into the update phase where it refines the predicted values by real observations. By recursively running the two phases, the Kalman filter can compute the optimal values of the current state in the presence of noise (Welch and Bishop, 1995).

Kalman filling is a technique which uses the Kalman filter's estimated states to represent the missing data. The Kalman filling algorithm first uses some reasonable data to pre-fill the missing data at the corresponding time steps. Then, it applies the Kalman filter to the data and obtains a set of estimated states for every time step. Finally, it replaces the pre-filled data by these estimations. *Forward Kalman filling* (*backward Kalman filling*) uses the data in the ascending (descending) order of time.

3.3 Adversary location estimation

Algorithm 1 illustrates how an adversary estimates a user's location. Since the trajectories are continuous, and the moving pattern of people can be modelled by a linear process with noise, the unreported locations in a single user's trajectory can be estimated by using forward and backward Kalman filling.

Algorithm 1 Tracking algorithm based on Kalman filling

```
1: Input: $H_R$ reported locations at each time step, $H_E$
   external knowledge for pre-fill the unreported ones in $H_R$
2: Use $H_E$ to pre-fill the empty items in $H_R$, obtained $H_F$
3: Apply forward Kalman filter to $H_F$, obtained $H_{KF}$
4: Apply backward Kalman filter to $H_F$, obtained $H_{KB}$
5: for Each unreported location in $H_R$ do
6:     Replace it by $\alpha * H_{KF} + \beta * H_{KB}$ ($\alpha, \beta$: weighted
       values)
7: Return Kalman filling result $H_R$
```

The adversary will first provide some synthetic locations to pre-fill the locations set, based on some outside knowledge. For example, the missing data can be linearly interpolated. Then, the interpolated data will later be used as measurements for the Kalman filter.

The adversary can use social relationships to predict a location as follows. The adversary will first determine a distance R_S. If the distance between user A and his friend, user B, is less than R_S, the adversary can deem them as being together. For all MSN friends of A, the adversary can calculate the percentage of them being together with A. When A applies an unreported location in a protected trajectory, the adversary can guess that A is staying with one of his friends. The partial traces of a user's friends can be used as an external location source to estimate the hidden locations during the use of Kalman filling.

3.4 Location privacy preserving metric

The evaluation metric is the average Kalman filter error between the predicted results and the actual locations at each reporting time. An adversary's error degree can be calculated as $W_{\text{adv}} = \frac{\sum_T \|Loc_G, Loc_R\|}{\sum amount(T)}$, where W_{adv} represents the average mistake degree of an adversary's guessing; $\| \bullet \|$ represents the Euclidean distance between a guessing location Loc_G and its corresponding real location Loc_R; T is the total time of observations; *amount* represents the amount of reported locations. If there are no fake or unreported locations in the observations, the value of W_{adv} is 0. The higher the adversary error degrees are, the safer a user's location privacy is.

3.5 Example of proposed Solution

Figure 2 outlines the basic idea of our solution. We give sequence letters from (a) to (e) to the pictures in Figure 2 from left to right. Figure 2(a) shows the original trajectory of a user, and Figure 2(b) shows the result after using the location omission strategy. After observing the reported locations of a user, the adversary can use Kalman filling to estimate the unreported ones and to guess the whole trajectory, shown as Figure 2(c). However, instead of not reporting some locations, the user can report some plausible fake locations, which makes the estimation of the Kalman filter inaccurate. If the positions of these fake locations are plausible, then the adversary cannot filter them out. Although the adversary can apply the Kalman filter to minimise the influence of fake locations, the position of our fake locations can bring about the maximum estimation error among all other positions. Figure 2(d) and (e) represent the trajectory after using the fake location strategy and the Kalman filter estimation result.

4 Proposed solution

Our approach towards providing location privacy is to report fake location information to the adversary such that the adversary cannot obtain the user's real trajectory. We begin by considering the case of a user travelling alone, followed by a situation where the user is travelling with friends.

4.1 User travelling alone

Here, the user is travelling alone without any of his friends nearby. When choosing fake locations to update, the user wants to select locations that cannot be easily filtered out by the adversary. Figure 3 illustrates the appropriate area of fake locations: reported points (x_0, y_0), (x_1, y_1), and (x_2, y_2) are three places. As for place (x_1, y_1), we want to use another fake location to replace it. When considering the speed limitation, the fake location should be located at a point in the intersection of two speed rotundities.

4.1.1 Relationship between estimation error and fake location distance

Now suppose that in order to protect the privacy of his own trace, a user wants to use N% of his locations as fake

Figure 2 An adversary's estimated results with and without fake locations. (a) real trajectory; (b) reported trajectory; (c) estimation without fake locations; (d) trajectory of using fake locations and (e) corresponding estimation result (see online version for colours)

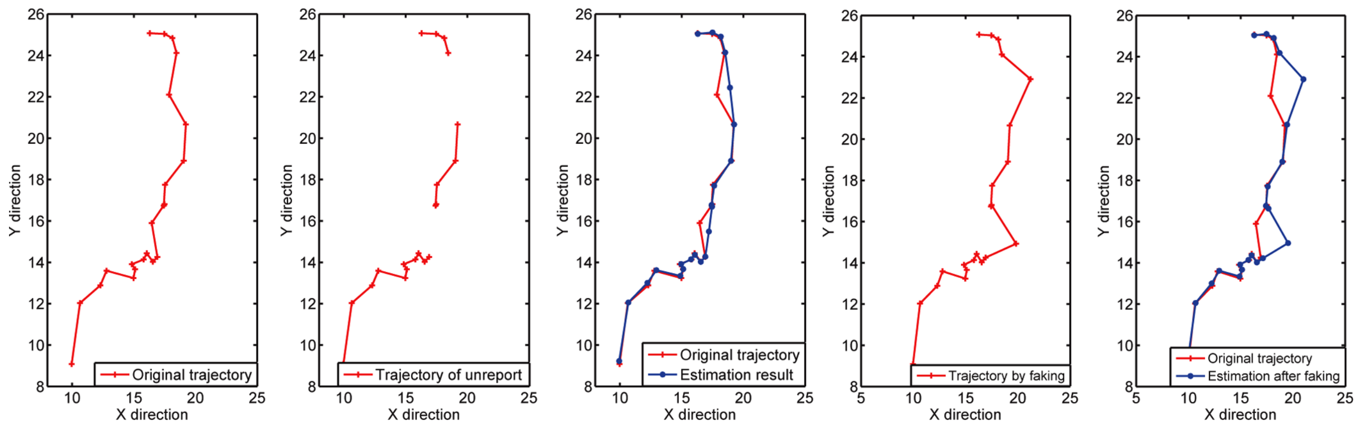
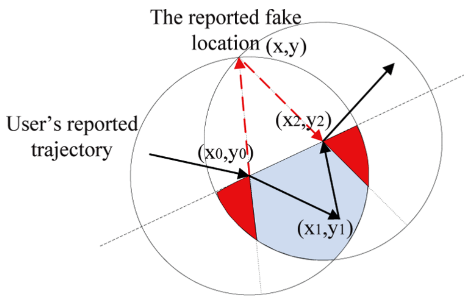


Figure 3 The position of an optimal fake location (see online version for colours)



locations. We assume that at time k, the user uses fake location $(x + \Delta x, y + \Delta y)$, where x and y represent the real location. Δx and Δy can be thought as some user-specified *noise*. Now, we want to calculate the relation between fake locations and their corresponding estimated locations. The estimation error of the Kalman filter can be evaluated by $P_{k|k}$. $P_{k|k}$ is a covariance matrix, which stores the covariance of the state at time k based on observations at K. According to the definition of the Kalman filter, the covariance can be written as follows:

$$P_{k|k} = (I - P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}H)P_{k|k-1}, \tag{1}$$

where I is an identity matrix, $P_{k|k-1}$ is the covariance of the state at time k based on the past observation at time $k-1$, H is an observing matrix, R is the covariance of noise, k is a time instance, and H^T is the transposed matrix of H. From this equation, we can see that if the covariance of *noise* in a user's historical location set is relatively larger, the covariance between observation and prediction will be greater. In other words, the farther a fake location is from the real location, the greater the estimation error will be.

4.1.2 *Estimation error of Kalman filter*

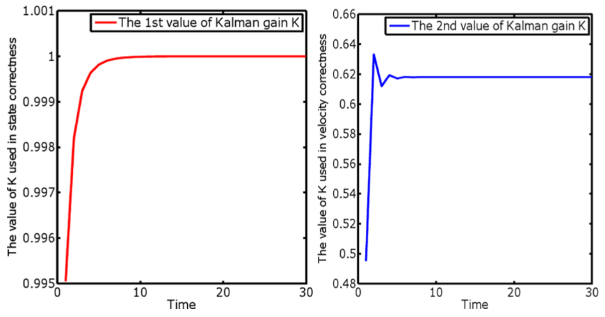
The Kalman gain (updating ratio) K determines the trust of a predicting system to its observation and its predicting result (Welch and Bishop, 1995). From Welch and Bishop (1995), we know that under conditions where R is constant, both the estimation error covariance $P_{k|k}$ and the Kalman gain K will stabilise quickly and then remain constant. Figure 4 shows the change pattern of the Kalman gain of a 1-D Kalman filter.

Next, we consider the total estimation error led by *noise* in a 1-D space. Assume that we add Δd to one location in either the x or y direction. Then, the additional estimation error Δe in the following n^{th} step can be represented as follows:

$$\Delta e = ((I - KH)F)^n K\Delta d, n = 0, 1, 2, \dots , \tag{2}$$

where I is an identity matrix, K is the Kalman gain, H is an observing matrix, and F is the state transition model. The trace of a user can be divided into x and y directions, which means the moving trace of a user can be seen as the combination of two traces from the 1-D space.

Figure 4 The change pattern of Kalman gain (see online version for colours)



The Kalman filter has the capacity to deal with noise because the expected value of noise is zero. As a result, if we provide *noise* in the same direction as a real location, the influence of fake locations can accumulate and disturb the estimated result of the Kalman filter. However, the value of Δe will decrease exponentially when n goes up: if we provide a fixed value Δd to a group of traces, the extra estimation error is a fixed value: $\sum_n((I - KH)F)^n K\Delta d$. We use $\lambda\Delta d$ to represent this fixed value. Suppose that we use N% of our reported locations as the fakes. Since a fake location can be considered as a real location adding $\overline{\Delta d_i}$, the average estimation error of a trace is $N\% \times \lambda\overline{\Delta d_i}$.

4.1.3 *Fake locations reporting pattern*

There are two options for reporting fake locations. The first is to continuously report fake locations for some time period followed by continually reporting the real locations. The alternative is providing the fake locations separately: the reporting time of the fake locations and the real locations is mixed together. The estimation error of Kalman filling at a location is the sum of errors caused by previous fake locations. However, as the parameter decreases rapidly, intensive reports of fake locations will cause the remainder of the locations to be less protected. Therefore, we will separately report fake locations to protect the whole trace. A user will only provide intensive fake location updates when protecting a specific sensitive location and not the entire trace. Therefore, if a user wants to protect a certain location, she can intensively report some fake locations. Though, as for using fake locations to distort the whole trajectory of a user, it is better to report the fake locations separately.

4.1.4 *The optimal fake location*

We can view a person's moving trajectories as a group of points in two dimensions. Therefore, to find the best fake locations at each point, we have to find the maximum value of $\Delta x^2 + \Delta y^2$ for a particular speed restriction. Suppose that (x_0, y_0), (x_1, y_1), and (x_2, y_2) are three consecutive points in a trajectory. Assume that (x, y) represents the corresponding fake location to point (x_1, y_1). With the limitation of speed, the maximum distance a user can reach in the time interval is R. We first join two points, (x_0, y_0) and (x_2, y_2), together, as shown in Figure 3. The third location (x_1, y_1) would reside on either the joined line or on one side of the line. The fake location (x, y) should satisfy the following restrictions:

$$R = S_{\max} * \Delta t \tag{3}$$
$$(x - x_0)^2 + (y - y_0)^2 \leq R^2 \tag{4}$$
$$(x - x_2)^2 + (y - y_2)^2 \leq R^2 \tag{5}$$
$$\text{Goal : maximise}(\sqrt{(x - x_1)^2 + (y - y_1)^2}). \tag{6}$$

If the current real position (x_1, y_1) is located in the lightly shadowed region of Figure 3, then the best fake location is at the point of intersection of two speed limitation circles. Normally, a user makes a U-turn far less than others. In order to reduce the complexity of the algorithm, we still use the intersection point as an approximate solution when the point (x_1, y_1) is in the darkly shadowed region. Therefore, the best fake location (x, y) can be computed as follows:

$$a = (\frac{y_2 - y_0}{x_2 - x_0})^2 + 1 \tag{7}$$
$$q = \frac{(x_2 - x_0)(x_2 + x_0) + (y_2 - y_0)(y_2 + y_0)}{2(x_2 - x_0)} - x_0 \tag{8}$$
$$b = -2(y_0 + \frac{y_2 - y_0}{x_2 - x_0} q), c = y_0^2 - R^2 + q^2 \tag{9}$$
$$(x, y) = (q + x_0 - \frac{y_1(y_2 - y_0)}{x_2 - x_0}, \frac{-b + \sqrt{b^2 - 4ac}}{2a}) \tag{10}$$
$$\text{or } (x, y) = (q + x_0 - \frac{y_2(y_2 - y_0)}{x_2 - x_0}, \frac{-b - \sqrt{b^2 - 4ac}}{2a}).(11)$$

However, the best fake locations are obtained based on the assumption of knowledge of the next location. If we name current time t, then the best fake location of t is determined by the real locations of $t - 1$, t, and $t + 1$. Although we have found the best fake locations of each reported point in a trace, realistically, users will not provide each (x_2, y_2) to the application. We assume that there is an application that can predict the next position (x_2, y_2) by using a current real location and a current instantaneous speed. The corresponding fake location generation algorithm is given by Algorithm 2.

Among all of the optimal fake locations of a trace, we note that the fake locations, which increase the errors of Kalman filling, are always the fake locations at turning points. In our method, we let each user's application record the distribution of the distances between each of the real locations and their corresponding fake ones. Therefore, reporting $N\%$ of the locations as fake means finding a distance threshold D such that $\sum Dist(d \geq D) = N\%$. At each time interval T, if the distance between a fake and real location is greater than D, the application will report this fake location, otherwise it will report the real location. The corresponding algorithm is presented by Algorithm 3.

Algorithm 2 Fake location generation algorithm

```
1: Input: Previous location $(x_0, y_0)$, current location$(x_1, y_1)$, current speed $S_C$, time interval $T$, maximum speed $S_{max}$
2: Use $(x_1, y_1)$ and $S_C$ to predict next location $(x_2, y_2)$
3: Compute the two potential fake locations based on formula 9-10, and calculate the distances from them to $(x_1, y_1)$
4: Return the furthest potential fake location $(x, y)$
```

Algorithm 3 Single user trace protection

```
1: Input: Real location $(x_1, y_1)$, fake reporting percentage $N$, fake distance distribution $Dist$ and candidate fake location $(x, y)$
2: Compute the distance $\Delta d$ between $(x_1, y_1)$ and $(x, y)$
3: if $\sum_d Dist(d > \Delta d) \leq N$ then
4:     Return the fake location $(x, y)$
5: else
6:     Return the current location $(x_1, y_1)$
```

4.2 *Users travelling in a group*

Now, we consider the scenario where a group of friends are moving together. We need to address the following issues:

- Who will report the fake locations
- How many fake locations should be reported
- How a group of users' devices cooperatively protect location privacy while guaranteeing quality of service.

We will first present the solution for the simplest case of two friends travelling together before expanding our solution to multiple friends in a group.

4.2.1 *Group of two friends*

The probability of two users appearing together is proportional to the closeness of them. We use the distance distribution between two users (A and B) to measure the closeness of them. We apply the concept of mutual information to analyse the problem. The historical location sets of A and B can be represented by two random variables L_A and L_B. L_A represents the location set that A visited, and L_B stands for the visited location set of B. We use $I(L_A; L_B)$ to represent the mutual information between A and B's location sets. A high amount of mutual information between two users means that they always appear in the same regions at the same time.

The closeness of two users is subjective. We use a predefined threshold to determine the state of *being together*. We first compute the distance's distribution of two users. Then, we calculate the probability that two users appear in the same region at the same short period of time, based on a predefined threshold. The probability represents the chance that the two users stay together.

According to Information Theory, mutual information can be equivalently expressed as $I(L_A; L_B) = H(L_B) - H(L_B|L_A)$, where $H(L_B)$ means the entropy of user B's location set and $H(L_B|L_A)$ means, given A's location set, the entropy of B's location set. This formula shows how much uncertainty in guessing L_B has been removed by knowing L_A. Therefore, if user A reports his location at time t while B doesn't, the uncertainty of those unreported locations of B will be reduced since adversaries can get more information from A's locations. According to the definition of mutual information, by increasing the uncertainty of getting together, or by increasing the probability of arriving at some location alone, the location-binding mutual information between users will become less.

A high amount of mutual information between A and B also means a high value of $Prob(L_B|L_A)$. The adversary can guess the position of B at a certain time t by $Prob(L_B) = Prob(L_B|L_A) \times Prob(L_A)$. If we can increase the uncertainty of A's reported locations, then B's locations can be protected better and vice versa.

Our fake location generating method works as follows: when the distance between A and his friend, B, is less than a predefined *being together* distance threshold, two users will report their real location and fake location at the same time, in turns. For example, A reports his real location at time t_1, and B reports a fake location. Then, at time t_2, B reports the real location, and A reports the fake one. Through this method, we reduce the mutual information between two users and also increase the uncertainty of reported locations.

4.2.2 Group of multiple friends

Now we consider the case where the user is travelling with a group of friends. Unlike an online social network, we can assume that the friends in this group have a higher level of trust since they are friends in both cyber-space and physical space. We can exploit this fact to improve energy efficiency. We note that short range data transmission between nearby nodes can consume less energy than transmitting to the AP, and some users in the group can enjoy the service provided by the server without submitting any information simply by obtaining it from their friends. Table 2 contains the notations used in this part.

Assuming that there are k friends moving together, the simplest solution is randomly choosing one user to report real locations while others report the fake ones. However, this solution contains the following problems.

- Since the user that is reporting the real location needs to retransmit the service data to his friends, how to assign the different roles (real location reporter, fake reporter, and the silent user) to different users is a problem.
- How to hide the generation pattern of fake locations is another problem.

Table 2 Table of notations for group's fake location reporting policy

k	*Number of users in a group*
$e(i)$	Remaining energy of user i
$RP(i)$	Probability of letting user i to submit real position
Tol	Users' tolerance degree for service failure
ρ	A parameter determined by k and Tor
$Threshold_E$	A dynamically changed threshold, below which the user does not need to report locations
μ	A predefined constant
R_s	Service radius

- If the current real position submitter leaves the group, who will take over the job?

When k friends make up a group, the users will periodically exchange the remaining energy, $e(i)$, $(1 \leq i \leq k)$. Based on the exchanged consecutive remaining energy and corresponding time stamps, the users can predict the remaining energy of any others in the group at any time. We want the real position reporting policy to be related with remaining energy. Hence, if $e(i)$ is the remaining energy of user i, the probability of i reporting the real location (RP(i)) to the server is:

$$\text{RP(i)} = \rho \times \frac{e(i)}{\sum_{j=1,\ldots,k} e(j)}, \tag{12}$$

where ρ is a constant number determined by users. We suppose that users of different services all have a service failure tolerance degree, a percentage, assuming Tol. The probability that no one reports the position at time t should be smaller or equal to Tol since under such case there is definitely no service.

$$(1 - \rho \times \frac{e(i)}{\sum_{j=1,\ldots,k} e(j)})^k \leq Tol. \tag{13}$$

As result, the parameter ρ can be computed as follows:

$$\rho = k(1 - Tol^{\frac{1}{k}}). \tag{14}$$

Hence, to any node i in the group, the reporting probability is:

$$\text{RP}(i) = \begin{cases} 1 & \text{for } \frac{\rho e(i)}{\sum_{j=1}^{k} e(j)} \geq 1 \\ \frac{\rho e(i)}{\sum_{j=1}^{k} e(j)} & \text{for others} \end{cases}. \tag{15}$$

When the maximum distance among k users is less than the predefined threshold, we consider that the k users become a group. We assume that the service's returning data covers a circular region whose center is the reported location and whose radius is R_s. In most applications, the initial distance among users can be ignored, compared with the R_s. In the very beginning, after k users join a group, each user, whose remaining energy is greater than an energy threshold $Threshold_E$, will virtually determine a random walk path from the location. The energy threshold is a dynamically changing value, which is determined by the remaining energy of all of the users at time t.

$$Threshold_E = \mu \frac{\sum_{j=1,\ldots k} se(j)}{k}, \tag{16}$$

where μ is a predefined constant. For example, if we define that the users whose remaining energy is less than 10% of the current average of the remaining energy, the less remaining energy users do not need to make a virtual random walk, then $\mu = 0.1$. The maximum distance between the end of the random walk and the user's current real position is $\gamma \times R_s$, where γ is a constant between 0 and 1. A large γ means having to protect more privacy, but it also causes more transition time (the time interval for letting user j report the real position instead of the user currently submitting real locations i).

Our fake location reporting algorithm works as follows. We first select the user with the most remaining energy to be the real location reporter. The remaining users whose remaining energy is greater than $Threshold_E$ make a bias random walk. A bias random walk is when the center of the random is not the current real location. The advantage of doing this is that the adversary cannot use the average value of the reported positions as the group's real location. After arriving at the determined random location, the user begins to report the fake trajectory. Considering that these k users may not join the group at the same time, it is hard for the adversary to determine which reported trajectory is the real one.

After a user is selected to report the real location, he will continuously report the real position for a random amount of time. Then, he makes a prediction about the remaining energy of the other users. The next real position submitter will be randomly selected based on the distribution of $RP(i)$. The current submitter will send a special notice to the next submitter. Again, we assume that the adversary is intelligent: he can use speed limitations to filter out some fake locations, and he can also use the speed limitations and moving patterns to predict the identity of the real position submitter. After receiving the special notice, the user will report a fake trajectory that is travelling towards the real position. During the period of travel, the previous submitter will continuously report the real position. In order to disturb the change pattern, each fake location submitter will randomly make a choice between continuously moving along the current fake path or moving to another fake path. If a fake location submitter decides to transfer to another fake path, he needs to send a special notice to the other user and wait for the confirmation. If a fake submitter receives the notice, he will send a confirmation and continue walking along his current fake path.

Consider a situation in which the current real location submitter may suddenly leave the group. If this happens, the real location submitter needs to hand over the job to the nearest user (based on the reported locations) or the user whose remaining energy is less than the threshold $Threshold_E$.

5 Additional discussion

A consequence of uploading a fake location to the adversary is that information associated with the fake location will be transmitted to the user and his friends. This can be mitigated as follows.

We suppose that users' mobile-phones have been equipped with at least two kinds of wireless communication devices: one is used to access service providers, and the other is a short-distance communication device, such as Bluetooth. In order to minimise the influence of using a fake location, we will partially use the service data of nearby users. In the friend-locator application, when a user sends a fake location to the server for trace protection, the application will also send $(\Delta x, \Delta y)$ within the package. Therefore, the user's friends will know the accurate location of the user.

Our method also allows us to provide finer grained location privacy based on social relationships. We can assume that close friends share more keys than regular acquaintances. We can encrypt $(\Delta x, \Delta y)$ by multiple layers. For instance, we assume that the maximum number of keys shared by two users is U. Then, we randomly pick up U real values such that $\sum_{i=1}^{U} \Delta x_i = \Delta x$ and $\sum_{i=1}^{U} \Delta y_i = \Delta y$ ($|x_i| > |x_{i-1}|$ and $|y_i| > |y_{i-1}|$). We also sort the key based on the number of keys being shared. Then, we assign *label* 1 to the key shared the least amount times, and we assign *label U* to the key that is the most commonly used. Then, we encrypt different (x_i, y_i) by the corresponding labeled keys. Whenever decrypting a layer, the users can obtain a little more information. As a result, the location privacy will be different to different people. The closeness level between different friends can be defined either manually, by having the user assign specific weights for each of his friends, or automatically, based on the frequency of their interactions.

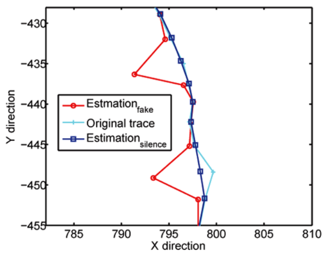
6 Evaluation

To perform the simulations, we first established a Cartesian coordinate system where the point $(10, 10)$ is the beginning of a user's trajectory. We set the initial user speed to be one unit distance per unit time, with a maximum speed of 7 units. At each step, we provide normal distributed noise as state noise and Gaussian distributed noise as observing noise. We use state noise to represent the speed and direction change of a user. We used both observation noise and state noise to randomise the movement change pattern of a user. After generating the locations at each step, we computed the statistical average speed of a user. The average speed is changed from 3.42 units of distance to 3.86 units of distance. In the experiments, we vary the observing time, continuously unreported locations, and the closeness between two friends.

6.1 Single user

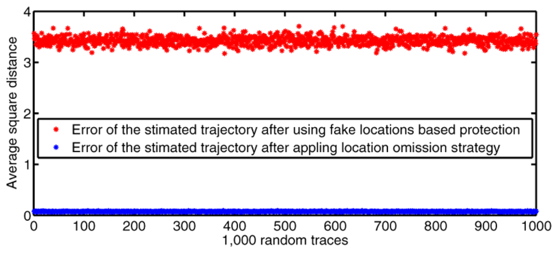
Here, we compare our scheme against the *location omission* strategy. Under the location omission strategy, the user will pick $N\%$ of their locations and not report them. In Figure 5, we intuitively presented the comparison result of our method and the location omission method. The figure shows the original trajectory of a user and the trajectory estimation result after using our fake locations-based privacy preserving method. Figure 5 is a part of a random trajectory consisting of 2000 points. We select 10% as a fake location reporting ratio, and we assume that the maximum speed is 7 units of distance. According to the statistics of our data, the average speed of a user is 3.47 units of distance. From Figure 5, we can see that the Kalman filling result is inaccurate by our choice of fake locations.

Figure 5 One user defense (see online version for colours)



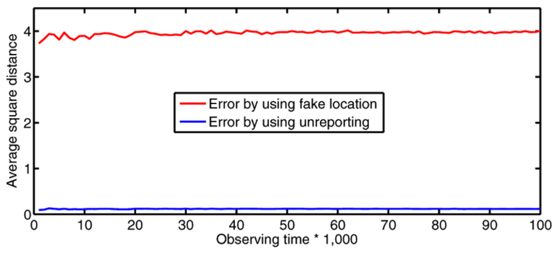
In order to evaluate the effectiveness of our proposed method to different random trajectories, we conducted the following simulation: we fixed the maximum speed of a user as 7 units of distance per time step and the initial position as $(10, 10)$. We leave the other parameters randomised, and we generate 1000 different trajectories. The average speed of these traces vary from 2.73 units of distance to 5.39 units of distance. Figure 6 shows the average estimation error per observation time between protecting via fake locations and unreported locations. In this experiment, we use 1000 random trajectories, and we observe each trajectory 2000 times. We compute both standard and squared distance by using our method and the other method. From Figure 6, we can clearly see that our proposed method can significantly improve the trace privacy of users.

Figure 6 Avg. error for uploading fake locations and not reporting (see online version for colours)



Next, we want to show that our proposed method is always effective, which means providing similarly averaged errors to the trajectory, regardless of the length of adversaries' observations. We then generate 1000 more random traces, but this time the observation time of each trace changes from 100 steps to 10^5 steps. During the simulation, we use 10% as the value of the fake location reporting ratio, and we force users' maximum speed to be 7 units of distance per time step. We compute the average error's change pattern as time goes on. From Figure 7, we can find that the average distance approaches a constant number. This result agrees with our research result in Section 4.

Figure 7 The relation between time and estimation error (see online version for colours)



6.2 A group of users

Since the simplest case of a group of users moving together is that two users are travelling together, we only use two users in our simulation. However, since the two users case results in the lowest amount of privacy by using our algorithm, if the results in the two users case is good, the solutions in the k users' group would definitely be better. Here, the adversary will compute the closeness between two users using a closeness metric. In this metric, if the distance between two users is less than a threshold, then we regard them as being together. If the closeness of two users is $k\%$, then when one user does not report her locations, there is a $k\%$ probability that she is near her friend.

In Figure 8, we let the number of continuously unreported locations vary from 1 time step to 50 time steps. We compute the average estimation error among 1000 random trajectories, and to each trajectory, we observed 1000 steps. From the image, we can see that when users do not report their locations in a short period of time, using Kalman filling produces better estimation results. However, if users do not report their current position for a relatively long period of time, using the location of users' friends has a better estimation result.

In this part, we will illustrate the results before and after using our proposed two user privacy-preserving method, shown in Figure 9. The picture in Figure 9(a) shows the original trajectories of two users, and the picture on the right-hand side is the histogram of their distance. If we defined the distance between them as less than 4 units of distance apart, then 64% of their locations are satisfied. Although after using our method, the percentage reduces to 38%. The image of Figure 9(c) shows the shape of two users' trajectories after using our method.

Figure 8 Avg. error with different continuously unreported locations (see online version for colours)

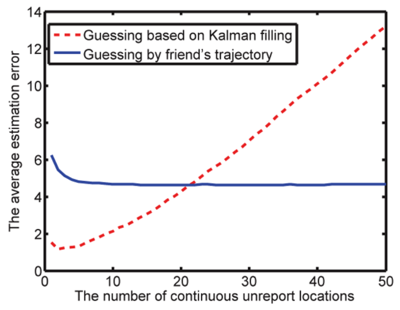
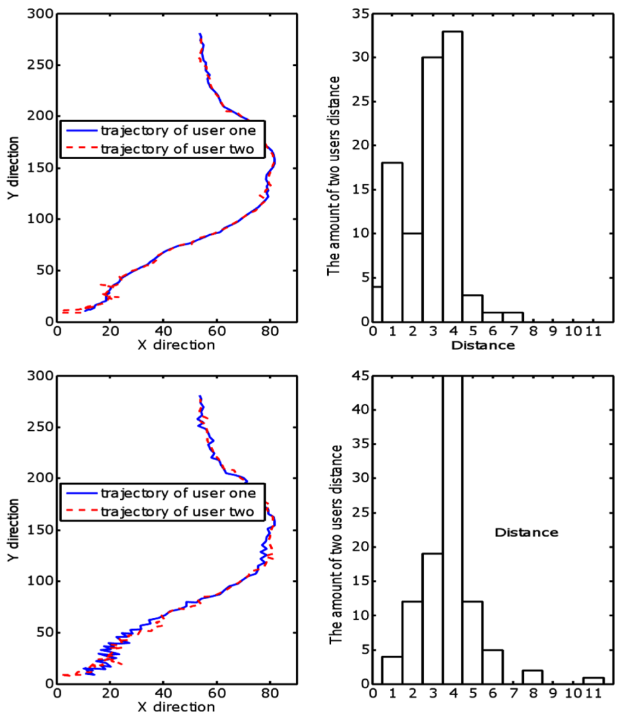
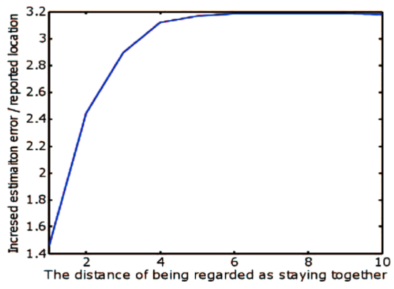


Figure 9 The trajectories and distance histogram between two friends before and after using our proposed method. Upper left (a) are the trajectories before using fake locations. Bottom left (c) is the result after using fake locations. Upper right (b) and bottom right (d) show the difference between users' closeness before and after using our proposed method (see online version for colours)



In order to see the relation between our proposed method and the distance standard used to discern two users from being together, we test 1000 pairs of users. For each pair, we observe them 1000 times. The average speed of these tested trajectories varies from 2.82 units of distance to 3.76 units of distance. During the generation of these traces, we define the maximum speed as 5 units of distance per time step. Figure 10 shows the change pattern of the average increased estimation error when the standard closeness distance changes from 1 unit of distance to 10 units of distance. When an adversary increases the distance standard, his guessing result will depend more on the locations of a user's friends since the closeness of the user and her friends will be increased, leading to an increase in the estimation error.

Figure 10 Avg. error under different closeness metrics (see online version for colours)



7 Conclusion

In this paper, we consider the problem of improving location privacy for MSNs. We introduce a new privacy attack where the adversary uses both historic movements and friendship information to estimate a user's trajectory. Our solution allows a user to upload fake locations to protect his privacy. When there is a group of friends moving together, our solution also provides the fake location uploading policy, which considers both the remaining energy of users and the privacy requirement. In our future work, we intend to consider the incorporation of mobility models, such as Levy-walks, to improve our scheme. Moreover, we also want to apply social relationship-based methods to the existing privacy-preserving techniques. Since a single user's privacy should be different to different people, we can design some social-based privacy-preserving algorithms.

Acknowledgement

This research was supported in part by NSF grants ECCS 1128209, CNS 1065444, CCF 1028167, CNS 0948184, and CCF 0830289.

References

Chao, S. and Dongyu, A. (recoverd by 2010) *Survey of Location Privacy*, http://students.csci.unt.edu/ da0097/

Chang, W., Wu, J. and Tan, C.C. (2011) 'Enhancing mobile social network privacy', *IEEE International Conference on Global Communications (Globecom 2011), Houston, Texas, USA.*

Chow, C-Y. and Mokbel, M.F. (2009) 'Privacy in location-based services: a system architecture perspective', *The Newsletter of ACM Proceedings of the 2nd International Workshop on Security and Privacy in GIS and LBS (SIGSPATIAL Special 2009)*, Vol. 1, No. 2, July 2009, pp.23–27.

Damiani, M.L., Bertino, E. and Silvestri, C. (2009) 'Protecting location privacy against spatial inferences: the PROBE approach', *ACM Proceedings of the 2nd International Workshop on Security and Privacy in GIS and LBS (SIGSPATIAL 2009)*, pp.32-41.

Freudiger, J., Manshaei, M.H. and Hubaux, J.P. and Parkes, D.C. (2009) 'On non-cooperative location privacy: a game-theoretic analysis', *ACM Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, pp.324–337.

Hidetoshi, K., Yutaka, Y. and Tetsuji, S. (2005) 'An anonymous communication technique using dummies for location-based services', *IEEE International Conference on Pervasive Services (ICPS 2005)*, pp.88–97.

Hoh, B. and Gruteser, M. (2005) 'Protecting location privacy through path confusion', *IEEE First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm 2005)*, pp.194–205.

Hoh, B., Gruteser, M., Xiong, H. and Alrabady, A. (2006) 'Enhancing Security and Privacy in Traffic-Monitoring Systems', *IEEE Pervasive Computing 2006*, Vol. 5, No. 4, pp.38–46.

Hoh, B., Gruteser, M., Xiong, H. and Alrabady, A. (2007) 'Preserving privacy in GPS traces via uncertainty-aware path cloaking', *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007)*, pp.161–171.

Huang, L., Matsuura, K., Yamane, H. and Sezaki, K. (2005) 'Enhancing wireless location privacy using silent period', *IEEE Wireless Communications and Networking Conference (WCNC 2005)*, Vol. 2, pp.1187–1192.

Jiang, T., Wang, H.J. and Hu, Y-C. (2007) 'Preserving location privacy in wireless LANS', *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (Mobisys 2007)*, pp.246–257.

Krumm, J. (2007) 'Inference attacks on location tracks', *Springer Pervasive Computing (PERVASIVE 2007)*, pp.127–143.

Kulik, L. (2009) 'Privacy for real-time location-based services', *ACM SIGSPATIAL Special*, Vol. 1, No. 2, , July 2009, pp.9–14.

Lavanya, S.T. and Elaine, C. (2004) *A Hole-Filling Algorithm for Triangular Meshes*, School of Computing, University of Utah, Salt Lake City, UL.

Marco, G. (2003) 'Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis', *Springer Mobile Networks and Applications*, Vol. 10, No. 3, pp.315–325.

Ouyang, Y. and Xu, Y., Le, Z., Chen, G. and Makedon, F. (2008) 'Providing location privacy in assisted living environments', *ACM Proceedings of the 1st International Conference on PErvasive Technologies Related to Assistive Environments (PETRA 2008)*, p.39.

Puttaswamy, K.P.N. and Zhao, B.Y. (2010) 'Preserving privacy in location-based mobile social applications', *ACM Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications 2010*, pp.1–6.

Welch, G. and Bishop, G. (1995) *An Introduction to the Kalman Filter*, University of North Carolina at Chapel Hill 1995,Chapel Hill, NC, USA.

You, T-H., Peng, W-C. and Lee, W-C. (2007) 'Protecting moving trajectories with dummies', *IEEE International Conference on Mobile Data Management (ICDM 2007)*, pp.278–282.