GPS privacy: Enabling Proximity-based Services while Keeping GPS Locations Private

Athindran Ramesh Kumar, Liang Heng, Grace X. Gao
University of Illinois at Urbana-Champaign

ABSTRACT
Location-based services have revolutionized people’s everyday lives. Proximity-based services in particular, are commonly used to identify nearby friends and other landmarks. However, these services endanger the privacy of users as the location information is provided to the service provider. In this work, we propose a privacy protection scheme which leverages the low-level data used in GPS location estimation for privacy protection. The anonymous range vector of a user to visible satellites is instead provided to the service provider. The exact location of the user cannot be inferred from the anonymous range vector. An efficient algorithm for proximity detection using the anonymous range vector is presented. A proof of concept of the proximity detection scheme is illustrated using experiments with u-blox LEA-6T receivers and Android phones.

1 INTRODUCTION
Ubiquitous GPS-enabled mobile devices and ever-growing location-based services (LBSs) are becoming more and more embedded in people’s everyday lives. The advancements in low-power positioning technology (assisted GPS), LBS middleware technology and 3G mobile services led to the boom of location-based services. The evolution and the commercialization of location-based services (LBS) after the initial failure is elaborated in detail in [1]. There are several types of location-based services. Some common uses include navigation and geo-spatial crowdsourcing. Besides, there are also proximity detection services and location based services which allow users to search for friends or other points around them [2]. Further in some services such as WeChat, users can even find all nearby strangers by shaking his/her phone. However, the locations of users are generally logged in a third party server. A malicious attacker can collect this location information and infer the personal attributes of a user including his whereabouts, life style, political and religious affiliations [3]. The high level of intrusion and privacy threats associated has created a reluctance among users to use location-based services [4].

There are two major forms of privacy associated with LBS [7], namely - Query privacy and Location privacy. Query privacy refers to users’ private information, such as their identity and lifestyle that can be exposed by looking at query attributes. Location privacy refers to users’ private information directly or indirectly exposed by their locations. These two notions of privacy are intertwined and loss of one type of privacy often leads to the loss of the other. Several of the privacy protection schemes focus on preserving query privacy. A class of these schemes degrade or transform location data in such a way that an adversary cannot infer the actual locations from the dataset. Anonymization, obfuscation, and adding random perturba-
tions are three such techniques. Gruteser et al. [8] introduced “cloaking” for query privacy and expanded the idea of k-anonymity, which was more commonly used in database security. Every user must be indistinguishable from k − 1 other users to satisfy the k-anonymity metric. The path confusion [9] approach proposed by Hoh and Gruteser focuses on protecting privacy in crowdsourcing applications. In their approach, the paths of the users are perturbed in order to ensure that an intelligent attacker using advanced tracking algorithms cannot track users. On the other hand, cryptographic schemes provide “perfect secrecy” but depend on pre-shared secrets and can be computationally expensive. Private information retrieval [11] is a cryptographic approach directed towards protecting query privacy.

Besides, there have also been a lot of privacy-protection schemes pertaining to our interest area of proximity detection services. The first techniques to enable private proximity detection used distance preserving transformations [12] to not reveal actual user locations. Most of the work in this domain assume the existence of pre-shared secrets or private shared functions. Secure computational geometry [13] used a more probabilistic approach. It introduced protocols to privately compute scalar products and test for vector dominance. These protocols could be extended to test for point-inclusion in a polygon. Zhong et al [14] discuss the shortcomings of this approach and introduce three protocols: Louis, Lester and Pierre for private proximity testing. These were based on public key encryption schemes and every pair of users have to establish a secure communication channel in order to test for proximity. The number of connections can quickly escalate in a large scale geo-social network. More recently, Arvind Narayanan et al. [15] propose the use of location tags for privacy protection. Location tags are unpredictable and reproducible signals that can be derived from the surrounding environment. SHARP [18] used location tags to establish proximity detection to strangers. During an initial phase, users embed a key in their location tag and send it to the server. Only users who share the same location tag can extract this key.

However, all of the above schemes treat GPS as a black box. In this work, we open the black box and leverage some of the “information” inside for privacy protection. We extract privacy-preserving location information directly from an intermediate step in GPS location estimation. We implement private proximity detection using anonymized pseudorange measurements. The anonymous range vectors of each user can also be interpreted as the location hash of an user. Further, we propose a simple, efficient and effective algorithm to detect whether two users are close. This algorithm is first validated using actual pseudorange measurements released by the IGS tracking network. Also, experiments using Android phones and u-blox LEA-6T receivers were performed to show the proof of concept on a smaller scale.

The rest of the paper is organized as follows. In section 2, we define the concept of a location hash and formulate a problem statement. Section 3 briefly describes the algorithms used for matching and introduce some performance measures. In section 4, we present experiments to validate our algorithms. In section 5, we summarize our work and discuss about avenues for future research.

2 PROBLEM STATEMENT

Our goal is to enable private proximity detection between users. In this section, we specify the adversary model in our framework. We consider a centralized social network where the proximity detection between all pairs of users is performed by a centralized server. A centralized network significantly decreases the number of connections required when there are a huge number of users. Also, note that there will be no user adversaries in a centralized social network as the server can easily hide information from other users. However, the server can itself be malicious and can try to infer the actual locations of the user from the information provided. Also, we assume that all communication channels are secure (via TLS, IPsec, etc.), i.e., there is no concern about eavesdropping or interception.

In order to achieve our goal, we seek a location hash function \( h(x_k) \) of the location \( x_k \) of a user \( k \). The hash function should not be easily invertible in order to prevent an attacker from inferring the actual locations from the hash. More formally, it should be computationally intensive to find a solution to the equation \( h(x) = h(x_k) \). Besides, there could be multiple locations that map to the same value of \( h(x) \), thus creating ambiguity about the actual user location. Also, it should be feasible to test for proximity detection on the location hashes. Let \( t \) be the threshold distance we intend to use for testing proximity between the users. There must exist a detection statistic \( T(f(h(x_1), h(x_2), \tau)) \) such that we can conclude that two users are close with high probability whenever \( T \) exceeds an adjustable threshold \( \tau \). In this paper, we use the anonymous range vector of an user as the location hash and derive the detection statistic \( T \) using a matching algorithm.

3 PRIVATE PROXIMITY DETECTION USING ANONYMOUS RANGE MEASUREMENTS

In this work, we use a set of range measurements to GPS satellites (without PRN designated) as the location hash of the user. Without the PRN’s designated to the range measurements, it is impractical for an adversary to compute the actual location of the user. We propose a scheme for proximity detection using anonymous range measurements. Let us denote the range vector of the \( i^{th} \) user as:

\[
\rho_i = \left[ \rho_i^{(1)}, \rho_i^{(2)}, \ldots, \rho_i^{(K_i)} \right]^T,
\]

where \( K_i \) is the number of satellites visible to the user. The range measurements \( \rho_i^{(l)} \) do not include the user clock...
bias which is removed beforehand. Further the range vector \( \rho_i \) is sorted beforehand in ascending order.

Consider two users at locations \( x_i \) and \( x_j \). Let the range vectors to the set of visible satellites of the two users be denoted by \( \rho_i \) and \( \rho_j \) respectively. Suppose there is a satellite visible to both users with corresponding range measurements \( \rho_{i1} \) and \( \rho_{j1} \), with elevation and azimuth \( \alpha \) and \( \beta \). We know that

\[
|\rho_{i1} - \rho_{j1}| \approx \|x_i - x_j\|_2 \cdot |\cos(\alpha) \cos(\beta)|. 
\]

(2)

Thus for any threshold distance \( t \),

\[
|\rho_{i1} - \rho_{j1}| \leq \|x_i - x_j\|_2 \leq t. 
\]

(3)

Thus, if two users at a distance \( t \) see a common satellite with ranges \( \rho_{i1} \) and \( \rho_{j1} \), it is a necessary condition that \( |\rho_{i1} - \rho_{j1}| < t \).

However, in our scheme, the server has to do blind matching of range measurements. In other words, the users do not designate PRN numbers to each range measurement in order to protect their privacy. We formulate this problem of “blind matching” in an optimization framework and propose an algorithm to do the same.

### 3.1 Blind matching of range vectors

Let \( \subset \) denote the subset relation between two sorted vectors. We write \( x \subset y \) if each element in \( x \) also belongs to \( y \). Let \( card(x) \) denote the cardinality, i.e., number of elements, of a vector (or a set) \( x \). The proximity detection problem can be formulated as the following optimization problem:

\[
\text{maximize} \quad c, \\
\text{subject to} \quad c = card(\phi_1) = card(\phi_2), \\
\phi_1 \subset \rho_1, \\
\phi_2 \subset \rho_2, \\
\|\phi_1 - \phi_2\|_\infty \leq t, 
\]

(4)

where the infinity norm \( \|u_1, \ldots, u_n\|_\infty = \max(|u_1|, \ldots, |u_n|) \). Further, we define the proximity detection statistic \( T \) as the number of mismatches,

\[
T = \min\{card(\rho_1), card(\rho_2)\} - c.
\]

Two users are said to be close if the proximity detection statistic \( T \) is less than a threshold \( \tau \). The decision threshold \( \tau \) is selected to adjust the false alarms and missed detections that can occur.

The optimization problem in gather (4) is similar to the longest common sub-sequence problem in the literature. Let \( \rho[n] \) denote the \( n \)th element in the sorted vector \( \rho \), and let \( \rho[n] \) denote the vector of the first \( n \) elements, i.e., \( \rho[n] = [\rho[1], \rho[2], \ldots, \rho[n]]^T \). We have the following recursive equation:

\[
c(\rho_{[k_1]}, \rho_{[k_2]}) =
\begin{cases}
0 & \text{if } k_1 = 0 \text{ or } k_2 = 0; \\
c(\rho_{[k_1-1]}, \rho_{[k_2-1]}) + 1 & \text{if } |\rho_{[k_1]} - \rho_{[k_2]}| \leq t; \\
c(\rho_{[k_1-1]}, \rho_{[k_2]}) & \text{if } \rho_{[k_1]} > \rho_{[k_2]} + t; \\
c(\rho_{[k_1]}, \rho_{[k_2-1]}) & \text{if } \rho_{[k_1]} < \rho_{[k_2]} - t.
\end{cases}
\]

(5)

where we use \( c(\ldots) \) as a metric of matching between the two vectors. Assume that the vectors \( \rho_1 \) and \( \rho_2 \) are sorted. An efficient dynamic programming algorithm to find \( c \) is presented below.

**Require:** two sorted vectors \( \rho_1 = [\rho_1^{(1)}, \rho_1^{(2)}, \ldots, \rho_1^{(K)}]^T \), \( i \in [1, 2] \)

**Require:** distance tolerance \( t > 0 \)

\( k_i \leftarrow 1, i \in [1, 2] \)

\( c \leftarrow 0 \)

**while** \( k_1 \leq card(\rho_1) \text{ and } k_2 \leq card(\rho_2) \) **do**

**if** \( |\rho_1^{(k_1)} - \rho_2^{(k_2)}| < t \) **then**

\( c \leftarrow c + 1 \)

\( k_1 \leftarrow k_1 + 1 \)

\( k_2 \leftarrow k_2 + 1 \)

**else if** \( \rho_1^{(k_1)} < \rho_2^{(k_2)} \) **then**

\( k_1 \leftarrow k_1 + 1 \)

**else**

\( k_2 \leftarrow k_2 + 1 \)

**end if**

**end while**

**return** \( c \)

Since the lengths of the two range vectors \( \rho_1 \) and \( \rho_2 \) are finite in practice, we can consider the time complexity of the algorithm to be \( O(1) \). Therefore, our proximity detection algorithm is very efficient.

As a statistical hypothesis test, private proximity detection has a probability of making two types of errors: false alarm and missed detection. Suppose there are \( N \) users and let \( S \) denote the set of all pairs of users, \( card(S) = \binom{N}{2} \). Let \( X \) be the set of pairs of users who are within a threshold distance \( t \). Let \( Y \) be the set of pairs of users who are detected to be close to each other. We define the following performance measures:

- **Probability of false alarm**

\[
P_{FA} = \frac{card(Y \setminus X)}{card(S \setminus X)}. 
\]

(6)

where the set difference \( Y \setminus X = \{ e \in Y | e \notin X \} \).
• Probability of missed detection

\[ P_{MD} = \frac{\text{card}(X \setminus Y)}{\text{card}(X)}. \]  

(7)

• Probability of detection error

\[ P_{DE} = \frac{\text{card}(Y \setminus X) + \text{card}(X \setminus Y)}{\text{card}(S)}. \]  

(8)

4 EVALUATION AND EXPERIMENTS

4.1 Global Evaluation

We first evaluate our proposed private proximity detection scheme in a global setting by using the IGS network. We treat the stations around the world as nodes to test for proximity using the scheme outlined in section. These stations are usually very far apart so we need to choose our threshold distances for proximity appropriately. Further, the pseudoranges recorded by these stations still have the user clock bias in them. The pseudorange measurement \( \tilde{\rho}^{(k)}(t) \) to the \( k \)th satellite at epoch \( t \) can be modeled as

\[ \tilde{\rho}^{(k)}(t) = r^{(k)}(t) + c(\delta t_u - \delta t^{(k)}(t)) + f^{(k)}(t) + T^{(k)}(t) + \epsilon^{(k)}(t) \]  

(9)

where \( r^{(k)}(t) \) is the actual distance to the satellite, \( \delta t_u \) and \( \delta t^{(k)}(t) \) are the user and satellite clock offsets respectively, \( f^{(k)}(t) \) and \( T^{(k)}(t) \) are the ionospheric and tropospheric propagation delays and \( \epsilon^{(k)}(t) \) include the other sources of error such as multipath and measurement noise. Most receivers use a Newton-Raphson based technique to solve for the user position and user clock bias, given the pseudoranges and the ephemeris. We use a similar such technique and extract the clock bias out of every station’s pseudorange measurements. We then find

\[ \rho^{(k)}(t) = \tilde{\rho}^{(k)}(t) - c(\delta t_u) \]  

(10)

\[ = r^{(k)} - c(\delta t^{(k)}(t)) + f^{(k)}(t) + T^{(k)}(t) \]  

(11)

and use \( \rho^{(k)}(t) \) to form the range vector. Note that the range measurements will still have the errors due to atmospheric terms. However, this is actually a benefit for privacy protection. It is an additional source of randomness that is correlated for nearby users.

The algorithm was validated using the IGS data recorded on January 10, 2014. The pseudorange measurements released by 1171 stations around the world at the start of the day at one time epoch was used to aggregate the statistics for validation. For each pair of stations, the number of matches in the pseudorange vector and the proximity detection statistic \( \zeta \) was computed. Figure 1 shows the frequency of the number of visible satellites from the IGS stations at the chosen time epoch. The number of visible satellites is around 9 on the average. In Figure 2, we plot the proximity detection statistic as a function of distance for a threshold of \( \tau = 5000 \) m. We can clearly see that the number of mismatches is higher for stations which are farther than the threshold distance. Figure 3 shows the variation of \( P_{FA} \) with the threshold distance. The variation is as expected from the theoretical results. From figure 4, we see that the missed detections go to 0 for \( \tau \geq 1 \). However, the false alarms ratio is obviously higher for higher values of \( \tau \). Thus, the experimental results clearly illustrate the viability of our scheme for efficient private proximity detection.

4.2 Local Evaluation

With the evaluation using the IGS tracking network, the "user" locations were fixed. Further, most pairs of stations...
were very far apart and the locations of the stations in the tracking network do not model the distribution of mobile-phone users very well. Thus, we perform some local experiments to further validate the algorithm. An android app was written to continually log range data. Upon post-processing, we evaluate the utility of our scheme. However, working with the Android API presents a fresh set of challenges. An additional evaluation using u-blox receivers is presented to further strengthen the proof of concept.

4.2.1 Android application data

The goal of our scheme is to enable privacy protection in mobile devices. Thus, a prototypical android app was developed to test the scheme with the GPS engine in Android.

Figure 3. Probability of false alarm versus threshold distance $t$ for the IGS evaluation.

Figure 4. Probability of missed detection versus threshold distance $t$ for the IGS evaluation.

In an ideal scenario, an implementation of a proximity-based service using an android app would just have the android app interacting with a friend-finder server as shown in figure 5. However, there were several challenges while working with the Android location API which is the only mode of accessing the underlying GPS engine. We provide a description of the challenges and the suitable modifications required below.

- Pseudorange measurements not available from the API

  An app developer can only access $(x, y, z)$ information of an user. Thus, we had to set up an external server to continually download high rate ephemeris from nearby IGS stations. The high rate ephemeris from the stations NIST and GODS were downloaded and hosted on a server. The android app downloaded and updated the ephemeris from the server periodically. Further, android provides a GPSSatellite class which reports the satellites currently in view along with the elevation and azimuth. Using the last known location, the satellites in view and the downloaded ephemeris, we find the range measurements and form the anonymous range vector for proximity detection. In an ideal scenario, we would expect the chipset to provide pseudorange measurements as well in the NMEA messages.

- Unknown clock bias and time sync issues

  Another major issue is that the android API does not report accurate GPS time. It only reports the UTC time at fix and the clock error can be as large as a few minutes. In order to circumvent this problem, the app was forced to compute the satellite positions and range measurements at fixed time epochs. Initially, we manually configured each of the devices used for testing to not have time lag more than 5-10 seconds. However, the range measurements vary rapidly as the satellites are continually moving. In order to ensure
Figure 6. IGS stations used for downloading ephemeris (marked in red).

time sync between users, the app was forced to report range measurements at fixed time epochs, i.e., integer multiples of 20 seconds.

- Fluctuating satellite set

As stated earlier, the GPS Satellite class reports the satellites currently in view along with the elevation and azimuth. However, the satellites in view were very fluctuant and the 4-5 satellites reported changed very rapidly (on the order of seconds). Thus, we aggregated the reports over 10 seconds to find all the satellites in view.

- Synchronous vs Asynchronous implementation

In an asynchronous implementation, the user notifies the LBS server when he/she is looking for friends nearby. The LBS server can then notify the user’s friends and obtain their location hash to test for proximity. In a synchronous implementation, the users report their location hashes periodically at fixed time epochs. The former implementation is obviously better in terms of privacy because the users give away less information. Also, there is the extra communication overhead of continuously reporting data in the synchronous implementation. However, we resort to a synchronous implementation in this work for simplicity. Further, we do not implement a friend-finder server as we are simply recording data for research purposes. The android app just logs all the pseudorange data in a text file. We processed the files from all users after the recording to evaluate our algorithms.

Figure 7 encapsulates the final structure of the android application to overcome the challenges involved. Figure 8 shows a screenshot of the android application with all the logged data.

The app was distributed to 6 graduate students who lived and worked in Urbana-Champaign, IL. They primarily worked indoors and were close to each other for most part of the data collection. We present the results from this data collection in figures 9, 10, and 11. In figure 9, we plot the ratio of correct decisions, false alarms, and missed detections. Unlike the IGS stations, most users in this experiment were close to each other for the most period. Hence, we see a relatively higher ratio of false alarms as the denominator in equation (6) is considerably smaller. Figures 10 and 11 shown an interesting comparison of the number of mismatches between the anonymous range vectors of two users using two different threshold distance parameters $t$. We see that the results are consistent with what we expect.

4.2.2 u-blox LEA-6T receiver

In the previous section, we elaborated on the challenges and our approach to android data collection. We had to construct our own range measurements from the satellite ephemeris and the last known position. We thus decided to perform another experiment with u-blox LEA-6T receivers. It is well known that these receivers give out RXM-RAW messages with pseudorange measurements. Four graduate students from our lab took part in this experiment. Two of them drove in opposite directions from Urbana-Champaign, IL for a few kilometers. Two others were walking around near
Figure 9. Performance of the proximity detection algorithm with the data logged through Android phones.

Figure 10. Variation of number of mismatches with distance for two users over a day with $t = 250$ m.

Figure 11. Variation of number of mismatches with distance for two users over a day with $t = 1500$ m.

Figure 12. Paths of the four users during data collection.

Figure 13. Variation of number of mismatches with distance for first pair of users over a day with $t = 750$ m.

Figure 14. Variation of number of mismatches with distance for second pair of users over a day with $t = 750$ m.

Figure 15. Variation of number of mismatches with distance for third pair of users over a day with $t = 750$ m.

Talbot lab in Urbana, IL. The paths of these users are presented in figure 12. The results from these experiments are presented in figures 13, 14 and 15 respectively. From figure 13, we can see that there is a sharp jump in the number of mismatches when the distance between the users increases more than the threshold of $t = 5000$ m. This demonstrates the robustness of the algorithm. For the same pair of users, we see in figure 14 that, a lesser threshold of $t = 750$ m gives appropriate results in terms of the number of mismatches. We further reiterate these ideas by comparing the proximity detection statistic $T$ on a different pair of users in figure 15.

With the different experimental evaluations, we have covered a wide range of test cases. In the IGS evaluation, the stations were very far apart. However, this evaluation helped understand the statistics of the false alarms and missed detections. The android app based evaluation was useful in understanding the challenges that one might face while trying to implement proximity detection using anonymous range measurements in the real world. The u-blox receiver evaluation clearly demonstrated the efficiency and robustness of this scheme for performing proximity detection.

5 CONCLUSION AND FUTURE WORK

With the widespread use of location-based services in mobile devices, efficient schemes for privacy protection are essential. Here, we introduced a scheme for private proximity detection using pseudorange/range measurements associated with GPS satellites. We hide the PRN numbers asso-
associated with the range measurements, and present a scheme for proximity detection using the range vectors. The proof of concept of this scheme was presented using experiments with android phones and u-blox receivers. The implementation of the scheme with android phones was severely constrained by the limited information available from the Android API. However, the results prove the viability of the scheme for use in mobile phones. The experiment with u-blox receivers gave very promising results with actual pseudorange measurements. Integrating this scheme with those existing in the security literature can possibly strengthen privacy, thus opening up a new avenue for GPS privacy research.

REFERENCES


