Location Privacy for Users of Wireless Devices through Cloaking

Calvert L. Bowen, III, David R. Raymond, and Thomas L. Martin
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University, Blacksburg, Virginia
{triiip, raymondd, tlmartin}@vt.edu

Abstract

The continued growth in online services that provide users with content based on location presents a unique privacy concern for the user. Since the user cannot control the use of the data included in his network transactions once they leave the device, nor can he control the response from the location-based system (LBS), he must assume that information is available to an unknown observer who could use the information to estimate the user's location. This paper presents a cloaking system that preserves a user's location privacy by submitting multiple requests from disparate false locations to the LBS in rapid succession in order to confuse the observer and meet the user's pre-determined location privacy threshold.

1. Introduction

The growth of location-based systems (LBS), which provide the user with information based on his location, continues as the number of location-aware devices available to the public at reasonable prices increases dramatically. In [1], data is presented from a 2005 in-Stat market survey estimating the number of Global Positioning System (GPS) devices and IEEE 802.11 (WiFi) devices in the United States in 2005 to be approximately 133 million and 120 million, respectively. The report also estimated market penetration would increase to approximately 137 million by 2006 for GPS and 430 million by 2009 for WiFi.

Many of today's handheld devices are capable of using both GPS and WiFi. There are several sources for location information including devices and applications that use various wireless signals to calculate an estimate of the user's location. Examples include GPS, Navizon [2] and Place Lab [3]. Navizon and Place Lab both use multiple inputs, if available, including GPS and WiFi, to generate estimates of the user's current location. Regardless of the source or type of location information, it is possible to derive a more accurate location estimate for a user by aggregating all available information.

Typical uses of LBSs involve Internet search queries for information based on the user's location. These queries and the associated responses can be used to enhance the estimate of the user's location, and anyone with access to the information can use the same algorithm to estimate the user's location. The user has no way to manage the use of his location information once it has been sent to the LBS; therefore, he must assume that an outside observer may gain access to this information and use it to estimate his location. This creates a unique issue of location privacy that must be addressed.

Since the user has no control over his location information once it leaves his device, it is both reasonable and prudent to assume that an observer exists who has access to the user's information. This research is concerned with the observer types presented in [4]. There are two different strengths of observers – strong and weak. A strong attacker (observer) is able to eavesdrop anywhere in the network, linking the sender to the recipient. Weak attackers can only use information that leaks from the content of the online transactions. In the case of Internet queries, strong observers are capable of gaining access to the user's information. Several attacks on user location privacy are available to the observer as presented in [5], [6], [7], and [8]. Each of these attacks assumes a strong adversary. The intersection attack [8] is of great interest because it correlates several pieces of data that are common in Internet queries and responses. In this research, we place the following limitation on the observer: he does not have the ability to collect the RF signals between device and wireless access points (AP) because this would disclose the user's location to within ~300m – the transmission range of an AP.

The system presented here applies a user's privacy threshold (described in Section 4) to help preserve the user's location privacy while he is wirelessly connected to the Internet. The system provides a boundary between the application and the Internet that

1530-1605/08 $25.00 © 2008 IEEE

assesses the information in the transaction to determine the amount and accuracy of location information being released. Artificial information is added to the transaction to confuse or overwhelm any observer.

The remainder of the paper is organized as follows. Section 2 presents other approaches and related work. Section 3 presents the mathematical foundation for both combining information and cloaking the user's location. Section 4 describes the cloaking system. Section 5 describes the tests and results used to verify that the system helps to preserve a user's location privacy. Section 6 discusses several resource concerns associated with the cloaking system. Section 7 provides potential areas for continued research and conclusions are drawn in Section 8.

2. Previous work

To date, the approaches to protecting a user's location privacy has been to attain anonymity of the user by hiding within a group of other users across a period of time. However, a solitary user cannot use these techniques to protect his location.

Gruteser and Grunwald [9] use *k-anonymity* to provide both location privacy in both space and time. The location information is presented in a tuple with intervals for the *x* and *y* coordinates as well as the time that the user was in the location. The location intervals represent a rectangle which includes the user's position and the time interval indicates that the user was at that location at some point during the period. The algorithm calculates the size of all three intervals based on the user-defined value for k_{min}, which identifies the set size for acceptable anonymity. The algorithm sets the location intervals to be large enough that *k-1* other users are in the region. To establish the time interval, the algorithm tracks the number of users that are in the region until k_{min} users are present, and then opens the interval. At this point, a random cloaking factor is added to the start time and the interval is closed. If the solution set exceeds k_{min} then the algorithm continues to iterate, reducing the intervals as necessary until k_{min} is achieved.

Beresford and Stajano [10] combine the use of *pseudonyms* and *mix zones* to provide location privacy. Pseudonyms provide some level of anonymity as the user does not provide his true identification to the LBS. In order to be effective, the pseudonym must change during the time period of observation. Failure to do so would allow an observer to merely track the pseudonym to determine the location of the user. The periodicity of changing pseudonyms is not addressed specifically, but it must be short enough that multiple changes are achieved in the amount of time it takes the user to move through a specified area. The use of pseudonyms alone does not prevent an adversary from tracking a user. The mix zone is used to provide an area in which the user may cross paths with other users, in effect identifying an anonymity set similar to the one required for k-anonymity. The size of the mix zone is based on how far a user can move within one location update period because a zone that is larger than this distance may result in incomplete mixing.

Kido, et al. [11] create and send false position data ("dummies") to LBSs to anonymize the user's location. Their work uses an anonymity set construct to determine how to distribute the locations for the dummies. Three factors – ubiquity, congestion, and uniformity – are quantified based on the number and distribution of users in a region and used to calculate the anonymity set. Two algorithms are presented which generate the dummies' current and next locations. The Moving in Neighborhood (MN) algorithm randomly generates the next position based on the current position. The Moving in a Limited Neighborhood algorithm extends the MN algorithm by assessing the density of users within the region.

In each of these works, there is a requirement for there to be at least two, and usually several, users in the region during a calculated period of time. None of them addresses the case where the user is operating alone in a region. If there are no other users in the region, then these techniques cannot provide location privacy for the user. Our cloaking system benefits from the presence of additional users, but does not require them. The presence of additional users forces the observer to attempt to combine more pieces of location information from multiple users. This may be particularly effective if an identification cloaking scheme, such as pseudonyms, is also in place as the observer would also have to determine which data were associated with his user of interest.

3. Mathematical foundation

Today, there are several devices in the public market which have multiple types of sensors on-device. Personal Digital Assistants (PDAs) like the Dell Axim series include both WiFi and Bluetooth radios as well as a card slot which can be used to add a third party GPS. Each of these sources can be used by an application (e.g. Navizon or Place Lab) to produce a multiple-input location estimate. In [12], it is shown that the dependency relationships of each estimate source must be considered prior to making the combination and producing the new estimate.

Since the system is measuring signals from various sources, each source should be treated as a separate variable with a distinct variance and standard deviation. For example, a series of readings for source

x_1 would consist of a position estimate for each trial in a time period, t. The distance between the estimated position and the true position can be calculated for each individual estimate. Collectively, a distribution of these differences can be formulated and the standard deviation and variance can be calculated. This process is repeated for each of n sources in the environment resulting in n separate variables. Based on a units assessment, the use of standard deviation, s, is most appropriate because it is measured in meters whereas the variance, s^2, is measured in meters squared.

Given a heterogeneous environment and a system that can process multiple types of signals, the data sets can be considered as independent variables. For example, GPS and WiFi signals measured at the same location at the same time would not interact with each other, and could be considered independent. As such, there is no correlation between the data sets.

A common method for combining location estimates is to take the mean of the estimates to produce the new combined estimate as in [13, 14]. Equation (1) provides the variance for two variables that are combined using the average as the combining function. The result of this equation relates directly to the value of the correlation coefficient, ρ. Independence implies that ρ=0. Solving (1) with ρ=0 produces the standard deviation in (2).

$$s^2_{\frac{x_1+x_2}{2}} = \frac{s^2_{x_1}}{4} + 2\rho\frac{s_{x_1}}{4}\frac{s_{x_2}}{4} + \frac{s^2_{x_2}}{4} \quad (1)$$

$$s_{\frac{x_1+x_2}{2}} = \frac{1}{2}\sqrt{s^2_{x_1} + s^2_{x_2}} \quad (2)$$

Parratt [15] presents the propagation of error through calculations when combining multiple independent random variables in a function, $u(x_1,x_2,\ldots,x_j)$, where x_j is the jth random variable. The standard deviation of u, s_u, is given by (3). Notice that this calculation is general in nature and does not specify how the variables are to be combined. Rather, this calculation is based on the function u used to combine the variables.

$$s_u = \left[\sum_{j=1}^{J}\left(\frac{\delta u}{\delta x_j}\right)^2 {s_{x_j}}^2\right]^{1/2} \quad (3)$$

Applying (3) to the average of two sources, x_1 and x_2, produces the same results as (2). However, Parratt's equation allows for various functions to be considered as the combining function $u(x_1, x_2)$ as long as all variables are independent. The cloaking system uses two functions – average and weighted average – to verify that the results meet the user's location privacy threshold introduced in the next section. Equations (4) and (5) are used to calculate these standard deviations, respectively.

$$s_{avg} = \frac{1}{j}\sqrt{\sum_{i=1}^{j}\sigma^2_{x_i}} \quad (4)$$

$$s_{wt\_avg} = \frac{\sqrt{\sum_i^j n}}{\sum_i^j n}\sigma_{x_1} \quad (5)$$

4. Cloaking system

Preserving a user's location privacy requires the use of several factors including a *location privacy threshold* (LPT) as described in [16] which has two components, the *distance threshold* (T_D) and the *probability threshold* (T_P). Establishment of the LPT may occur in one of two ways: default value or user-determined value. Default values are established for T_D and T_P, but can be overridden if the user knows the values that he wants to use for both (or either) thresholds. We have not found any research that specifies any metrics for a user's comfort level for location privacy. Therefore, it is assumed that a user would accept a distance of 1,000m as T_D because it is more than three times the estimated range for a wireless AP. The default value for T_P is 0.2. In this case, the assumption is that a user would be comfortable with the observer being able to blindly find the user in 20 out of 100 possible locations. Additional investigation into a user's comfort level is outside the scope of this research, but could be used to adjust the default values once available.

With the thresholds in place, several randomized actions take place to generate false locations based on the user's true location. Pseudocode of the cloaking system flow is depicted in Algorithm 1 with a full description of the system flow available in [16]. As conditions are met, the system continues towards providing the user the results to his web query. The first action taken is to identify a *seed* location which is based on the user's true location. Once the seed has been generated, the same process is repeated to generate a list of random false locations, called *bogeys*, using the seed as the reference point instead of the user's true location. These bogeys are sent to the LBS for processing once T_D has been initially satisfied. Bogeys are generated with random numbers that use the device's system clock as the seed value. As a result, each value will be different. While this makes experimental replication a challenge, it works in our favor as we do not want a predicable set of bogeys each time the set is developed. The number of bogeys to be generated is an option for the user to set. The number selected by the user is the minimum number of bogeys that will be generated. The notation

Algorithm 1: User location cloaking.

```
Inputs: user_location, min_number_of_bogeys (user selected, default =
12), T_D, T_P

threshold_area = π*T_D^2;
generate random_seed;
loop
   generate bogeys;
   if(index > min_number_of_bogeys)
   {
      generate bogey_box;
      if(bogey_box_area > threshold_area)
      {
         if(user_location inside bogey_box)
         {
            generate and execute queries;
            cluster responses based on T_D;
            calculate min_response_area;
            if(min_response_area > threshold_area)   //T_D is satisfied
            {
               calculate p(select);
               if(p(select) < T_P)    //T_P is satisfied
               else add bogey (return to loop);
            }
            else add bogey (return to loop)
         }
         else add bogey (return to loop);
      }
      else add bogey (return to loop);
      if (T_D and T_P are both satisfied)
      {
         generate standard deviations;
         if(s_avg and s_weighted_avg > T_D)
         {
            show user results;
            end;
         }
      }
   }
   else add bogey (return to loop);
end loop
```

min(bogeys) = *x* is used henceforth to indicate which set of runs is being discussed. Possible values of *x* are 5, 10, 12, 15, and 20. The actual number of bogeys generated for each run is determined during execution as explained in the next paragraph. The default value for the number of bogeys to be generated is 12 because the *min(bogeys)=5* and *min(bogeys)=10* runs were meeting the threshold area constraint (also explained below) as the 12th bogey was generated.

Once the user-determined number of bogeys are generated, a preliminary analysis of the privacy threshold distance, T_D, must be completed. First, the list of bogeys is used to build a *bogey box* which includes both the user's true location and the seed. It is imperative to include the user's true location in the bogey box in order to ensure that the results from the queries provide information that is useful to the user. The area of the bogey box must also be greater than the threshold area of $\pi*T_D^2$. If the bogey box does not include the user's true location or is not larger than the threshold area, additional bogeys are added one at a time until the bogey box includes the user's location.

Each bogey in the list provides a "location" for a set of queries. To assist the user, queries are predefined for several services: restaurants, gas stations, and hotels. The user selects the query type and clicks a button on the user interface to begin the query generation process. The format for each query is the same except for identifying which service the user desires. The number of queries generated is based on the number of bogeys in the list. Each bogey is used to generate ten queries. An example query for hotels in the area would be:

```
"http://www.google.com/maps?hl=en&q=hotels&nea
r=" + bogeys.locLat[bogeysIndex] + "%2C" +
bogeys.locLon[bogeysIndex];
```

where `locLat` and `locLon` represent the latitude and longitude for each bogey.

It is assumed that the observer would use all of the queries and response data to identify those areas which have a relationship between them. It is likely that services will be located in groups throughout the area. For example, downtown, main street, or high-density population areas are potential regions for these groups. These regions can be graphed using a minimum spanning tree (MST) using Kruskal's MST algorithm [17]. The MST is then pruned based on the value of T_D by cutting those edges whose weight is greater than T_D. Figure 1 depicts the set of clusters produced after a sample query where the original MST has had several edges cut based on the default value of 1,000m for T_D. These cuts result in the identification of five clusters. In order to satisfy the user's T_D, polygons are created for each cluster and the area is calculated for each polygon. The sum of these areas must be greater than the threshold area.

Once the distance threshold has been met, the probability threshold, T_P, must be assessed. Each location identified as the seed, bogeys, or in the responses includes values for latitude and longitude. These values are used to determine the percentage of locations that are within T_D meters of the user's location. This value is considered as the probability that an observer could randomly select a location inside the distance threshold, *p(select)*. In order to satisfy the user's probability threshold, the following relationship must hold: $p(select) \leq T_P$. When this is the case, the both components of the user's location privacy threshold have been met, and no additional bogeys or queries need to be generated.

Finally, the standard deviations are calculated using both average and weighted average as combining functions. For each location (seed, bogey, response), the standard deviation is calculated for the latitude and

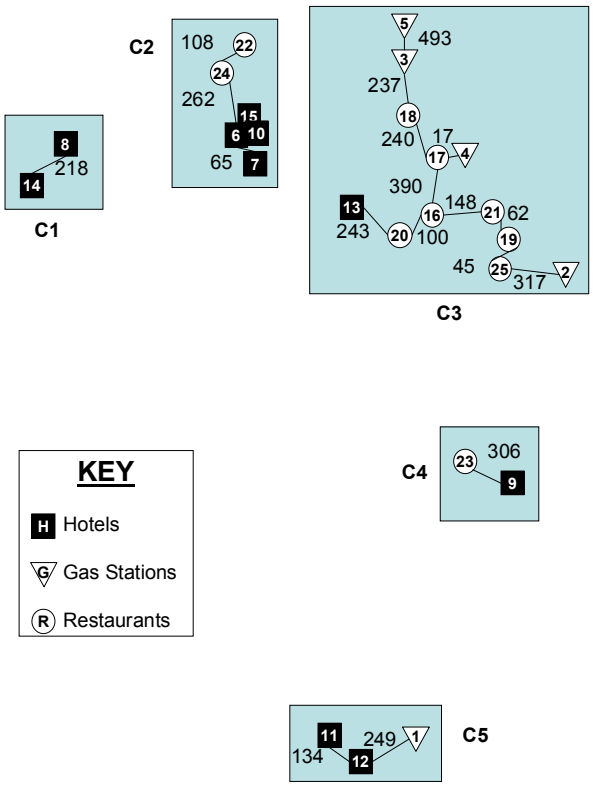

Figure 1: Pruned MST with T_D= 1,000m.

longitude individually. Further analysis is conducted by looking up the latitude and longitude of the zip code that is listed in each response. The standard deviation of these values is also calculated. These standard deviations are combined using both equations (4) and (5). If both are greater than T_D, then the system presents the user with the results of his request. However, if this is not the case, additional bogeys are generated until all conditions are met.

5. Cloaking system analysis

In this section, we analyze the results of our cloaking system experiments to show that the system meets the user's LPT. The system was deployed on six different devices: one iPAQ 4150, one iPAQ 4155, two Dell Axim X30s, and two Dell Axim X51s. These devices were selected primarily based on availability. However, the systems could be deployed to any device runs a Windows mobile operating system as far back as Windows Mobile 2003 Second Edition (SE). Four of the tested devices run Windows Mobile 2003, SE – iPAQ 4150, iPAQ 4155, and Axim X30s. The Axim 51s run Windows Mobile 5.

The cloaking system analysis was conducted using randomly generated user locations based on two environments – rural (Blacksburg, VA, population

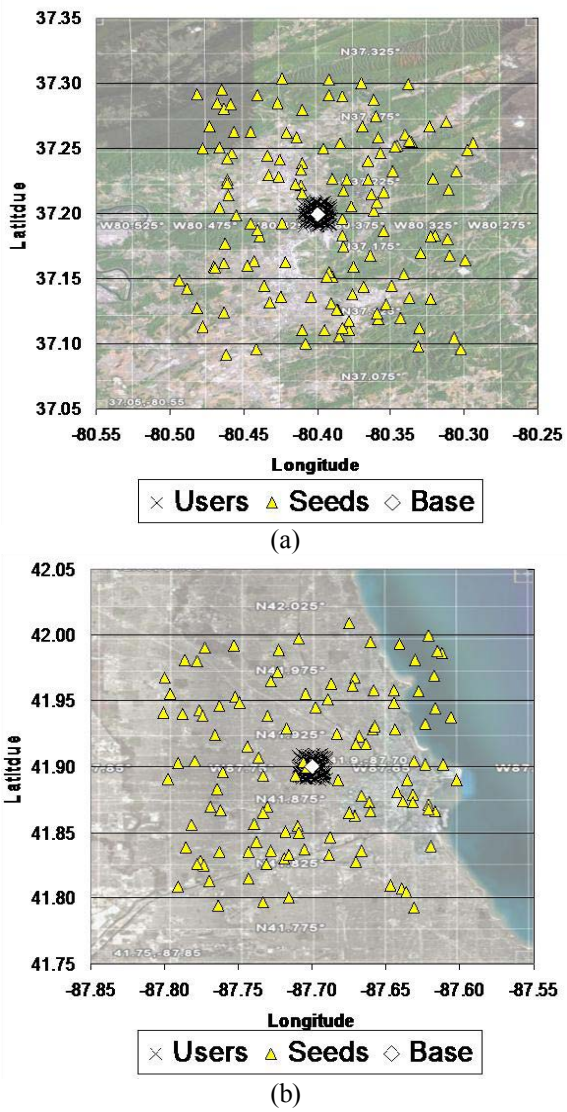

Figure 2: Dispersion of users and seeds for Blacksburg (a) and Chicago (b).

~80,000) and urban (Chicago, IL, population ~2.5 million). These two environments were selected because the density of services and population is drastically different and the system needs to be able to operate effectively in either case. Partial sets of the user locations and seeds for both environments are shown in Figure 2.

5.1. Repeated responses

In both environments, it is likely that some of the responses will be repeated in any single run of the cloaking system because the randomly generated bogeys may be in close proximity to each other. The response data was analyzed for both environments and show that despite the repetition which occurs, there are

enough of the lodging, gas, and restaurant services in both environments to allow the system to meet the user's LPT. Using 12 bogeys, in the rural region, 54 services of 120 listed in the results (45%) were unique. Despite the low percentage of uniqueness, these services were geographically separate enough that only 2.9% of the 54 entities were within the default of 1,000m for T_D. Similar analysis for the urban region provided dramatically different results: 116 of 120 responses (96.7%) were unique. In this case, 0.78% of the entities were located within 1,000m of the user's location. These percentage values are both significantly lower than our default value of 20% for T_P.

5.2. Probability of selection – *p(select)*

Test results shown in Figure 3 demonstrate that the average values for *p(select)* are well below the default value for T_P of 0.2 (dashed line). However, the maximum values observed for *p(select)* are near or above 0.1 for all four sets of tests. This result supports using 0.2 as the default value.

5.3. Standard deviations

Analysis of the various measurements of standard deviation shows that the cloaking system was able to meet the requirements for T_D. Figure 4 shows the average standard deviations calculated for each component as well as for the combining functions for all runs on the iPAQ 4155. All values are greater than T_D (dashed line).

6. Device and network resources

Because this system is deployed on PDAs, resource consumption is a primary concern. This concern is especially important because the use of several bogus locations to cloak the user's true location will impact resources both on- and off-device. The two critical resources that this research is focused on are power and network bandwidth. The third critical resource is the user's wait time while the system is working. This section presents the testing procedures and results for determining the impact of the cloaking approach on the limited resources available on all six devices.

6.1. Testing configurations

There are different configurations and applications used to complete the tests for resource consumption. Some were on-device while others required connection to a computer running a Windows operating system

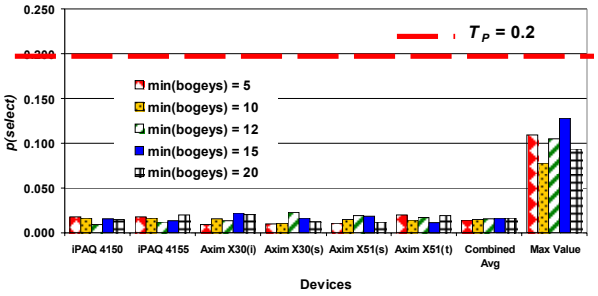

Figure 3. Average *p(select)*.

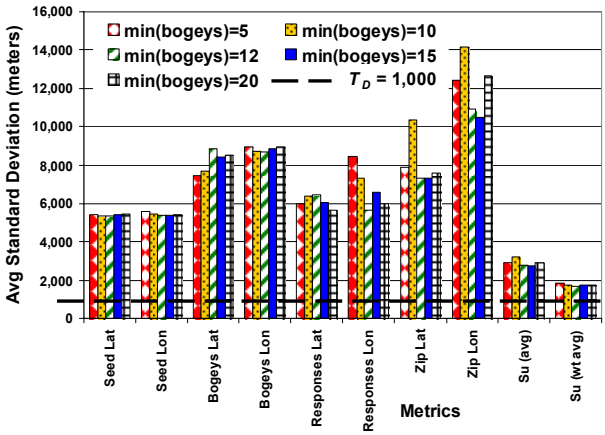

Figure 4: Average standard deviations for the iPAQ 4155 (meters).

with the Performance Monitor application built in. Windows XP (home edition) was used for these tests.

6.1.1 Smart battery. Data from the smart battery is used to calculate energy consumption both with and without the cloaking mechanism in place. The Smart Battery Data Specification [18] allows sampling at various rates but discourages high sampling rates due to System Management Bus (SMBus) constraints in both bandwidth and availability of the SMBus itself. The specification also recommends that these values should be updated in memory "within five (5) seconds of the associated parameter change" and read from the memory location at least once every ten seconds. All of the devices used in this research operate within these recommendations. Samples were acquired from the smart battery driver using the SYSTEM_POWER_STATUS_EX2 [19] class which is part of the Microsoft Compact Framework standard library.

It should be noted that each battery and device configuration varies between manufacturers. Results from this research show that even devices from the same manufacturer implement the configuration differently. For example, the timing of the updates by the battery driver varies between devices. The Axim

X30s update approximately every two seconds while the Axim X51s and the iPAQ 4150/55s update at an interval of approximately nine seconds. While it should be assumed that the current and voltage fluctuate more often than every nine seconds, these changes are not available to be read more frequently because of the specific implementation for each device-battery pairing. To gain a more accurate estimate of the changes in current and voltage, the device and battery could be connected to an oscilloscope. However, for this research, the oscilloscope is not necessary because the smart battery provides a first-order approximation of the power consumed by the device while executing different applications.

The period of time for updating the values for both current and voltage will decrease as smart battery technology advances. According to Jacoby [20], the industry is working towards a smart battery that "can report battery information at over 18,600 samples per second and is capable of taking current readings in time increments as low as 3.5 microseconds". Once this rate is achieved, one should expect an update rate of less than one second to be routine. Today, there is not a compelling need for these faster update rates because the SMBus data being presented is used to estimate the amount of time remaining for the battery charge which does not require a reporting rate of less than one second. Recent research, including [20] and [21], is taking advantage of the ability to sample the smart battery at a faster rate for other purposes. The two-second update rate exhibited by the Axim X30s is indicative of results observed in that work.

6.1.2. .NET CF performance monitor. Tests were conducted using the .NET Compact Framework (CF) 2's Remote Performance Monitor (RPM) calls available in service pack 1 as presented in [22]. These calls measure several performance metrics including the number of socket bytes sent and received across the network. This configuration requires the device to be connected to a computer through Microsoft's Active Sync application. The device application is started using an interface on the computer and the metrics are then reported to the Performance Monitor on the computer for collection in a log and post processing. The polling period can be changed during setup and was set to a one-second interval.

6.2. Power

Power is a leading concern of user of mobile devices. Any application used must limit its impact on charge lifetime of the device's battery. This section introduces several power saving mechanisms available on PDAs, discusses the development of a power baseline and the comparison of power consumption between that baseline and the cloaking system.

6.2.1. Power saving mechanisms. The devices used in this research have a charge lifetime that can range from hours to days depending on several settings and power management strategies that are built into the operating system. Each device is different, even if they are running the same operating system. Some devices have options to control the processor speed which allow the user to determine whether the device should determine the best operating speed for the work cycle, force the processor to work at the maximum speed, or use a proprietary power-saving algorithm to manage the processor speed. The iPAQs have a single setting for processor speed which cannot be reconfigured by the user. The Axim X30s have three user-configurable speeds: Power Save (208 MHz), Maximum (624 MHz) and Automatic which allows the device to manage the processor speed as necessary. The Axim X51s have four processor speeds: Power Save (208 MHz), Normal (416 MHz), Maximum (520 MHz), and Automatic.

Each device has a set of options for conserving power in addition to changing the processor speed. All have the ability to adjust the backlight brightness, set timing options for dimming the screen, set shutdown options for a lack of use, and set options to turn off WiFi as the battery charge wanes below a factory-set threshold. This last option is based on the 802.11 card in the device and is therefore, proprietary to the manufacturer of the device. The use of these options would extend the charge life for the devices and is recommended during normal use of the cloaking system. All of these power-saving mechanisms were disabled and the backlight was set to medium brightness for all tests. The processor speed was changed for tests as described in the sections below.

6.2.2. Establishing a power baseline. A baseline of power consumption was established for each device by running a "Battery Sampler" program which collected the instantaneous current and voltage from the smart battery. Each device was tested for each possible processor speed. The baseline was set by stopping all programs running in memory – other than File Explorer which was used to start the Battery Sampler – and enabling WiFi networking and disabling Bluetooth networking. The program ran for 300 seconds, producing 600 samples of instantaneous current and voltage from the smart battery in each device as the sample rate was hard-coded at 500 msec. The program was run on each device 10 times, resulting in 6,000 samples for each device. Instantaneous power was then calculated by taking the product of these two values. Since the cloaking system will be used with the WiFi

radio on, the values from the baseline (BL) will be used to determine how much additional power consumption is attributable to cloaking.

6.2.3. Comparing power consumption. In order to determine the impact of the cloaking system on the power consumption of each device, 20 tests were run with different values for *min(bogeys)=x* (where x = 5, 10, 12, 15, or 20). Each test was a single run of the cloaking system which accessed the Internet through a 5Mbps fiber optic connection. For the Axim X30s and X51s, the processors were set to "Maximum" in order to identify a potential upper bound on the amount of power consumed for each device.

The results of the 5-bogey tests presented in Figure 5 indicate that the increase from BL to the cloaking system is significant. However, on each device, they are all less than the highest instantaneous power value determined during the establishment of the BL. For example, the Axim X30(i)'s maximum power value during baseline testing was 1.528W which is 41% higher than the average value of 1.086W while using the cloaking system on that device.

Additionally, the cloaking system is not expected to be used continuously, but rather only when the user needs to access an LBS. Therefore, the execution time for the system is important to analyze. The execution times presented in Figure 6 show that the application is running, on average, for less than two minutes for *min(bogeys)=20*. The spike shown in Figure 6 for the iPAQ4155 was caused by a single run that lasted over eight minutes. Without that run, the average for the 4155 with *min(bogeys)=20* would be 56 sec instead of 1:59.

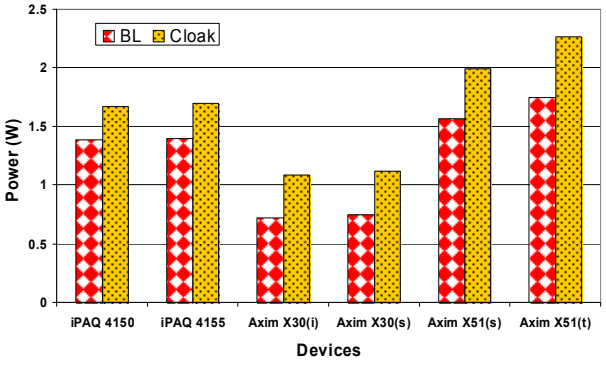

Figure 5. Comparison of power between BL and cloaking system with *min(bogeys)=5*.

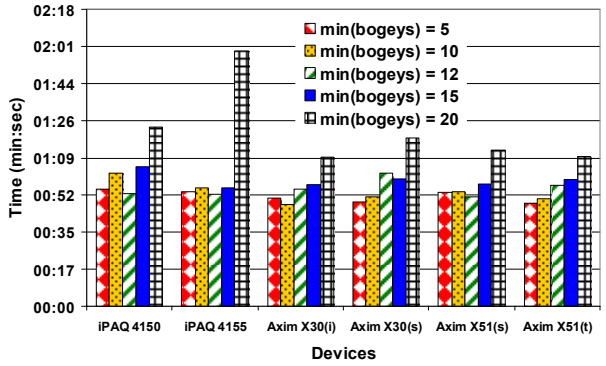

Figure 6. Average execution times.

The average power values for BL ($Power_{bty\_samp}$) and the average power values for the cloaking system runs ($Power_{cloak}$) can be used to determine how much impact the use of the cloaking system has on the device's operational time per charge. The amount of available energy per charge (E_{charge}) is calculated by calculating the product of the voltage and capacity ratings for the battery in each device as in equation (6). All of the batteries in the devices used in this research have a rated voltage of 3.7V. The capacity ratings are different for each type of battery. These values and the corresponding values for E_{charge} are shown in Table 1.

$$E_{charge} = V_{rated} * C_{rated} \quad (6)$$

Table 1: Available energy for a single charge.

Device	Battery Type	C_{rating} (mAH)	V_{rating} (V)	E_{charge} (J)
41xx series	PE2081AM	1,000	3.7	3,700
X30 series	X1111	950	3.7	3,515
X51series	T6476	1,100	3.7	4,070

In order to determine the true power consumption cost of using the cloaking system, a comparison can be made between the operational time per charge running the Battery Sampler ($OPtime_{bty\_samp}$) and the operational time per charge running the cloaking system ($OPtime_{cloak}$). Assuming that BL represents the device running no programs (other than sampling the battery), $OPtime_{bty\_samp}$ can be estimated using the by calculating the ratio between E_{charge} and $Power_{bty\_samp}$ per Equation (7). Similarly, $OPtime_{cloak}$ is calculated using E_{charge} and $Power_{cloak}$ per equation (8). Since both $OPtime_{bty\_samp}$ and $OPtime_{cloak}$ are calculated using E_{charge}, the reduction in charge lifetime ($\Delta OPtime$) can be calculated as a proportion between the two using equation (9) with the result expressed as a percentage. The results of the calculations for the *min(bogeys)=5* tests are shown in Table 2 and are spread across a large range which is attributable to the cloaking overhead power costs presented above. Similar results occurred in the 10-, 12-, 15-, and 20-bogey tests. The largest reduction in charge lifetime was 37.1% on the X30(s) for *min(bogeys) = 15*. It should be noted that these are worst-case power consumption overheads since they assume that the cloaking mechanism is running constantly.

$$OPtime_{bty\_samp} = E_{charge} / Power_{bty\_samp} \quad (7)$$

$$OPtime_{cloak} = E_{charge} / Power_{cloak} \quad (8)$$

$$\Delta OPtime = -1 * \frac{(OPtime_{bty\_samp} - OPtime_{cloak})}{OPtime_{bty\_samp}} \quad (9)$$

Table 2. Reduction in charge lifetime due to continuous cloaking with *min(bogeys) = 5*.

Device	E_{charge} (J)	$OPtime_{bty\_samp}$ (sec)	$OPtime_{cloak}$ (sec)	$\Delta OPtime$
4150	3,700	2,671.48	2,218.88	-16.9%
4155		2,648.53	2,181.88	-17.6%
X30(i)	3,515	4,902.37	3,237.12	-34.0%
X30(s)		4,724.46	3,145.97	-33.4%
X51(s)	4,070	2,602.30	2,041.54	-21.5%
X51(t)		2,332.38	1,800.50	-22.8%

6.3. Bandwidth

The use of multiple queries in a rapid succession has a minimal impact on bandwidth consumption. The tests included different numbers of queries to determine whether or not an increase in queries had an impact on the bandwidth consumed. The consolidated results are shown in Table 3. On average, each query took 3.8 seconds to complete. This is measured by elapsed time between start of each query and the end of the response for the last query and dividing that time by the number of queries sent. The average numbers of Mbps sent and received were 0.000234 and 0.203942, respectively.

One way to determine whether or not a device or application is consuming a large part of the available bandwidth is to conduct a review of the firewall logs of the supporting infrastructure. This review did not indicate any sort of disruption or flooding on the Internet connection. No alert was sent to the security administrators on the network, thereby indicating that the cloaking application did not meet their criteria for a flooding attack.

6.4. User wait time

While the time per query results remained within a one second range regardless of the number of queries, the true timing issue is the user's patience. It took up to 6 min and 17 sec (377 sec) to complete 100 queries. Elapsed times for 30, 20, and 10 queries were 121, 75, and 39 sec, respectively. These values are more user-friendly given a user who is traveling by vehicle. If the user is traveling by foot, he may be willing to wait for the 100 query response if that is necessary. A visual inspection of the percentage of time used for different subtasks (Figure 7) indicates that almost 75% of the execution time is spent between waiting on Internet connection and LBS to process the request (getHTTP) and parsing the response (Process Responses).

Table 3: Network load.

# Queries	Time/Query (sec)	Avg Mbps Sent	Avg Mbps Received
5	3.4	0.000259	0.217858
10	3.9	0.000224	0.203004
15	3.6	0.000245	0.202330
20	3.8	0.000236	0.202404
25	4.0	0.000222	0.200049
30	4.0	0.000217	0.198006
Average	**3.8**	**0.000234**	**0.203942**

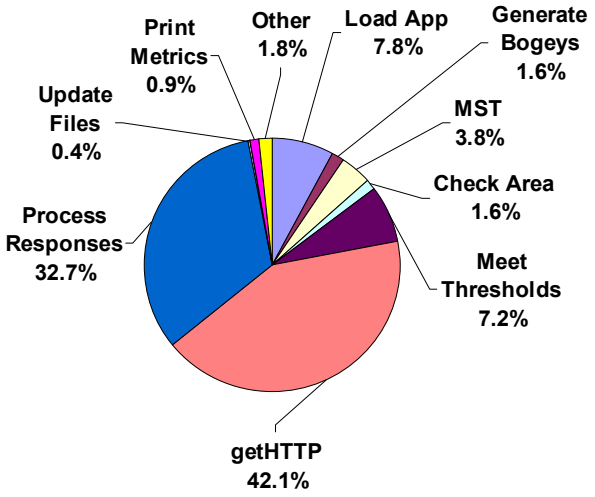

Figure 7. Processing time overview.

Overall, the results of our tests indicate that the traditional concerns of resource constraints in mobile computing are adequately addressed and the use of our application does not negatively impact the user's ability to continue using the device with respect to power consumption or battery charge lifetime. The issue of rapid multiple queries is also only a point of concern based on the user's tolerance for waiting for the response to his true information request. With average execution times between one and two minutes across all devices, this is deemed as acceptable for user wait time whether driving or walking.

7. Future work

There are several opportunities for future work associated with this cloaking system. First, there are efficiencies in the implementing C# code which would likely reduce the user wait time. Specifically, the use of parallel queries would address the process that takes the longest amount of time to execute. That said, there is some concern that parallel queries could indicate a flooding attack on the local network. This concern is based on the results of tests conducted on several devices simultaneously all through the same AP. Although there was not flooding indication on the local

network, the Google Maps server sent responses to the queries indicating that the queries appeared to be similar to know virus and worm attacks. After sending that response, the server denied those devices access for a limited amount of time. Second, additional research into determining a user's true comfort levels for his location privacy would provide data to refine the default values for T_D and T_P. Third, a context-aware system could use the cloaking system as one module in providing service to the user and protecting his privacy.

8. Conclusion

Anyone using a mobile device to connect to the Internet continuously puts their location privacy at risk. Users cannot secure the transaction once it leaves the device or the response when it comes in from the LBS and an outside observer could use this information to estimate the user's location. The cloaking system presented here allows the user to make his request to an LB and cloak his location by presenting multiple locations for consideration by this observer. The use of thresholds allows the user to determine his comfort level with regard to his location privacy.

9. References

[1] Kolodziej, K., "Advances in GPS: NAVIZON," in *LBSZone.com*, 2006.
[2] Mexens, "Navizon," New York, 2005, www.navizon.com.
[3] Intel, "Place Lab," Seattle: Intel Corporation, 2005, www.placelab.org.
[4] Choi, S., K. Kim and B. Kim, "Practical solution for location privacy in mobile IPv6," in *WISA 2003 (Revised Papers)*, Jeju Island, South Korea, 2004.
[5] Kong, J., X. Hong and M. Gerla, "A New Set of Passive Routing Attacks in Mobile Ad-hoc Networks," in *MILCOM 2003*, Boston, Massachusetts, 2003.
[6] Deng, J., R. Han and S. Mishra, "Countermeasures Against Traffic Analysis in Wireless Sensor Networks," in *SecureComm 05*, Athens, Greece, 2005.
[7] Cheng, R., Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving User Location Privacy in Mobile Data Management Infrastructures," in *PET 2006*, Cambridge, United Kingdom, 2006.
[8] Cvrcek, D. and M. Vaclav, "On the Role of Contextual Information for Privacy Attacks and Classification," in *Workshop on Privacy and Security Aspects of Data Mining*, Brighton, United Kingdom, 2004.
[9] Gruteser, M. and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *MobiSys 2003*, San Francisco, California, 2003.
[10] Beresford, A. R. and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing,* vol. 2, no. 1.
[11] Kido, H., Y. Yanagisawa and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *IEEE ICPS'05*, Santorini, Greece, 2005.
[12] Bowen, C. L. and T. L. Martin, "Combining Position Estimates to Enhance User Localization," in *WPMC 06*, San Diego, California, 2006, pp.648-652.
[13] Pandya, D., R. Jain and E. Lupu, "Indoor Location Estimation Using Multiple Wireless Technologies," in *IEEE PIMRC 2003*, Beijing, China, 2003.
[14] Gwon, Y., R. Jain and T. Kawamura, "Robust Indoor Location Estimation of Stationary and Mobile Users," in *INFOCOM 2004*, Hong Kong, China, 2004.
[15] Parratt, L. G., *Probability and Experimental Errors in Science; an Elementary Survey*. New York: Dover Publications, Inc., 1971.
[16] Bowen, C. L. and T. L. Martin, "Preserving User Location Privacy Based on Web Queries and LBS Responses," in *the 8th Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2007*, West Point, New York, 2007.
[17] Joseph B. Kruskal, J., "On the shortest spanning subtrees of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society,* vol. 7, no. 1, February, 1956.
[18] Friel, D., "4.4.5 Data Polling and Update Requirements," in *Smart Battery Data Specification (v.1.1)*: SBS-Implementers Forum, 1998.
[19] MSDN Library, "SYSTEM_POWER_STATUS_ EX2," Microsoft Developer Network, http://msdn2.microsoft.com/en-us/library/ms940385.aspx.
[20] Jacoby, G. A., "Battery-Based Intrusion Detection," in *Bradley Department of Electrical and Computer Engineering*. Doctor of Philosophy Blacksburg: Virginia Polytechnic Institute and State University, 2005.
[21] Buennemeyer, T. K., M. Gora, R. C. Marchany, and J. G. Tront, "Battery exhaustion attack detection with small handheld mobile computers," in *Portable 2007*, Orlando, Florida, 2007.
[22] Pratschner, S., "Analyzing device application performance with the .NET Compact Framework Remote Performance Monitor.", 2006, http://blogs.msdn.com/stevenpr/archive/2006/04/17/577636.aspx.